DEGREE PROJECT, IN SYSTEMS ENGINEERING, SECOND LEVEL
STOCKHOLM, SWEDEN 2015

Modelling and Control of a Fixed-wing UAV for Landings on Mobile Landing Platforms

GEORG ROBERT BALMER

KTH ROYAL INSTITUTE OF TECHNOLOGY

SCI SCHOOL OF ENGINEERING SCIENCES

Modelling and Control of a Fixed-wing UAV for Landings on Mobile Landing Platforms

HANNAH LINDBERG

Master's Thesis in Systems Engineering (30 ECTS credits)
Master Programme in Aerospace Engineering (120 credits)
Royal Institute of Technology year 2015
Supervisor at DLR, Germany: Tim Muskardin
Supervisor at KTH was Per Engvist
Examiner was Per Engvist

TRITA-MAT-E 2015:77
ISRN-KTH/MAT/E--15/77--SE

Royal Institute of Technology
SCI School of Engineering Sciences

KTH SCI
SE-100 44 Stockholm, Sweden

URL: www.kth.se/sci

Abstract

Landing on mobile landing platforms could eliminate the need for landing gear. This would particularly benefit high altitude solar UAV, which typically have a very limited payload. Such landings would however require a precise and decoupled control of the UAV's altitude and speed. In this thesis, a small UAV is modelled, and a flight control system suitable for such landings is developed.

The aerodynamic properties of the UAV were estimated using the vortex lattice method. Propeller performance data was obtained from the manufacturer and used in the propulsion model. The complete UAV model was validated using data from test flights. A comparison of period and damping of the dynamic modes showed a good agreement ($<10\%$ error) with the flight data, except for the phugoid damping, which was too low in the model.

The model was used to design two flight control systems, one consisting of three SISO loops for altitude, airspeed and course; and another using a TECS-based controller for airspeed and altitude. Extensive testing in simulation and flight revealed a superior performance of the TECS-based controller, especially in the ability to decouple altitude and airspeed responses.

Sammanfattning

Landningar på mobila plattformar skulle kunna avlägsna behovet av landställ. Detta skulle vara särskilt gynnsamt för högflygande solkraftsdrivna obemannade flygplan, som oftast har en mycket begränsad lastmöjlighet. En sådan landning skulle kräva en mycket noggrann och frikopplad reglering av planets höjd och hastighet. I detta examensarbete har ett litet obemannat flygplan modellerats och ett reglersystem passande för denna typ av landning har utvecklats.

De aerodynamiska egenskaperna hos det obemannade flygplanet har uppskattats genom VLM (Vortex Lattice Method). Prestandadata för propellrarna givet från tillverkaren har använts i modelleringen av framdrivningssystemet. Modellen för flygplanet har validerats med data från flygtester, och en jämförelse av perioder och dämpning av de dynamiska moderna visar på en bra överensstämmelse med flygdatan (<10% fel), utom för phugoiddämpningen som var för liten i modellen jämfört med flygtesterna.

Modellen har använts för att skapa två reglersystem; ett betsående av tre SISO-kretsar för höjd, hastighet och kurs, och en TECS-baserad regulator för hastighet och höjd. Omfattande simulerings- och flygtester visar att den TECS-baserade regulatorns prestation överträffar den SISO-kretsbaserade, framför allt i förmågan att frikopplade höjd- och hastighetssvar.

Acknowledgements

I would like to express my deepest gratitude to Tin Muskardin, my supervisor, for his great support during my stay at DLR. From solving administrative matters to discussing complex technical details, he was always willing to help, and I have very much appreciated that.

I would also like to thank my examiner, Dr. Per Enqvist, associate professor at KTH, for his continuous support despite the distance, and for his understanding whenever problems occurred.

Many thanks to my colleagues at the Flying Robots Group, led by Dr. habil. Konstantin Kondak, for accepting me as a peer and for their valuable feedback during many discussions.

Tack så mycket to Linnea Persson, who was so kind as to translate the title and abstract into Swedish.

I also thank the fellow students at the Institute of Robotics and Mechatronics, for the many enjoyable moments together, from coffee breaks in the sun to board game nights and weekend activities.

I owe my immense gratitude to everybody who gave me moral support throughout this thesis work. First and foremost, my girlfriend Netta Ussyshkin, whose love and support was invaluable; my brother Frédéric Balmer and the rest of my family. Also Sven Wlach, Matteo Abächerli, and other friends who encouraged me when I faced obstacles.

Finally, I would like to thank the many who are not listed here, but gave me their time, advice, encouragement or some other form of support at some point. Your contribution, however small it may be, is greatly appreciated!

Contents

List of Figures

List of Tables

List of Symbols

C_D	Drag coefficient [-]	m	Aircraft mass [kg]
C_L	Lift coefficient [-]	n	Propeller rotational speed [1/s]
C_Y	Side force coefficient [-]	p	Rolling rate [rad/s]
C_l	Rolling moment coefficient [-]	q	Pitching rate [rad/s]
C_m	Pitching moment coefficient [-]	r	Yawing rate [rad/s]
C_n	Yawing moment coefficient [-]	$\bar{q}$	Dynamic pressure [N/m^2]
C_P	Power coefficient [-]	α	Angle of attack [rad]
C_T	Thrust coefficient [-]	β	Sideslip angle [rad]
J	Propeller advance ratio [-]	γ	Flight path angle [rad]
P	Motor power [W]	δ_a	Aileron deflection [rad]
S	Reference wing area [m^2]	δ_e	Elevator deflection [rad]
T	Propeller thrust [N]	δ_f	Flap deflection [rad]
V_a	True airspeed [m/s]	δ_r	Rudder deflection [rad]
V_k	Ground speed [m/s]	δ_t	Throttle command [-]
W	Wind speed [m/s]	θ	Pitch angle [rad]
b	Reference wingspan [m]	ρ	Air density [kg/m^3]
c	Mean aerodynamic chord [m]	ϕ	Bank angle [rad]
d	Propeller diameter [m]	χ	Course [rad]
g	Gravitational acceleration [m/s^2]	ψ	Yaw angle (heading) [rad]
h	Altitude above ground level [m]	ω	Propeller rotational speed [rad/s]

1. Introduction

This final report is based on the author's work at the German Aerospace Center (DLR), as a part of DLR's ongoing research in the area of autonomous high-altitude aerial vehicles. In the frame of the EC-SAFEMOBIL project[1], a research group at DLR's Institute of Robotics and Mechatronics is working on safe and reliable landing technologies for such aircraft. This introductory chapter illustrates the context of this thesis, defines its goals and introduces the system composed of an aerial vehicle and a ground vehicle.

1.1. Background and Motivation

1.1.1. High Altitude Unmanned Aerial Vehicles

In the recent years, Unmanned Aerial Vehicles (UAV) have gained widespread interest, and they are now used in many areas. While UAV are still mostly known for their military applications, small UAV are also used in civil, both private and commercial contexts, mostly for aerial photography. High altitude, long endurance (HALE) UAV are promising in terms of potential applications; however, the development has not yet reached the stage where a commercial use is viable.

Intended for use in the stratosphere, at altitudes in the order of 20 km, HALE UAV typically have large wingspans and very lightweight structures. Equipped with solar cells and sufficient battery capacity, such airplanes can stay airborne day and night, which gives them theoretically unlimited endurance. This opens up possibilities for various applications, including earth observation, atmospheric science and communication networks. Some of these tasks are now performed by satellites, which are very expensive to build, launch and operate. Apart from saving the cost of a rocket launch, HALE UAV have more advantages: they are independent of an orbit and can operate anywhere, either staying at a given location, patrolling over a large area, or relocating as necessary, e.g. in response to a natural disaster. Unlike satellites, aerial vehicles are able to return to the earth surface for maintenance or updates.

One of the main limitations of HALE platforms is their typically low payload capacity. In order to fly at high altitudes, a low wing loading is necessary, which

[1]EC-SAFEMOBIL project website: `http://www.ec-safemobil-project.eu/`

translates to a large wing area and a low gross weight. However, the batteries required for overnight flight are heavy, and further reduction of the structural mass is very difficult, given the already extreme lightweight design. New approaches are required to lower the empty weight and in turn increase the payload.

1.1.2. Landings on Mobile Landing Platforms

The approach suggested in this project is to remove the landing gear. With mission durations of several months, the landing gear, which is only used for take-off and landing, is merely dead weight for most of the time. And it can be significant: in DLR's former HALE platform, Elhaspa, with a total mass of around 100 kg, the landing gear was in the order of 3 kg – almost half as much as the payload of 7 kg. Without aircraft-mounted landing gear, new solutions need to be found for take-off and landing. Take-offs could be realized using some sort of temporary gear that is dropped once the aircraft is airborne. Landings however are a bigger challenge, and this is the focus of a research project at DLR's Institute of Robotics and Mechatronics.

The core idea is to have a landing platform mounted onto a ground vehicle, which is able to accelerate to the UAV's landing speed and to cooperatively with the UAV control position and velocity, so that the UAV can touch down on the landing platform without a horizontal velocity component relative to the platform. This would not only eliminate the need for landing gear, but also facilitate landings in crosswind situations, as the aircraft does not need to align with the runway direction, it can land on the platform facing any direction. HALE UAV are typically very sensitive to wind, due to their size and low weight, and this proposed landing procedure could greatly improve their operational availability.

The overall goal of the DLR research project is to experimentally demonstrate the feasibility of such a landing. For this purpose, a small UAV is used together with a landing platform mounted on a car with a human driver. The landing manoeuvre is divided into six phases, illustrated in Figure 1.1, namely:

1. Initial approach
2. High precision approach
3. Guided descent
4. Final descent and touchdown
5. Deceleration to full stop
6. Ground lock

The main challenge associated with this landing procedure is the cooperative control of the vehicles – both of which are complex dynamic systems – ensuring a safe and efficient execution of the landing manoeuvre. On the aircraft side,

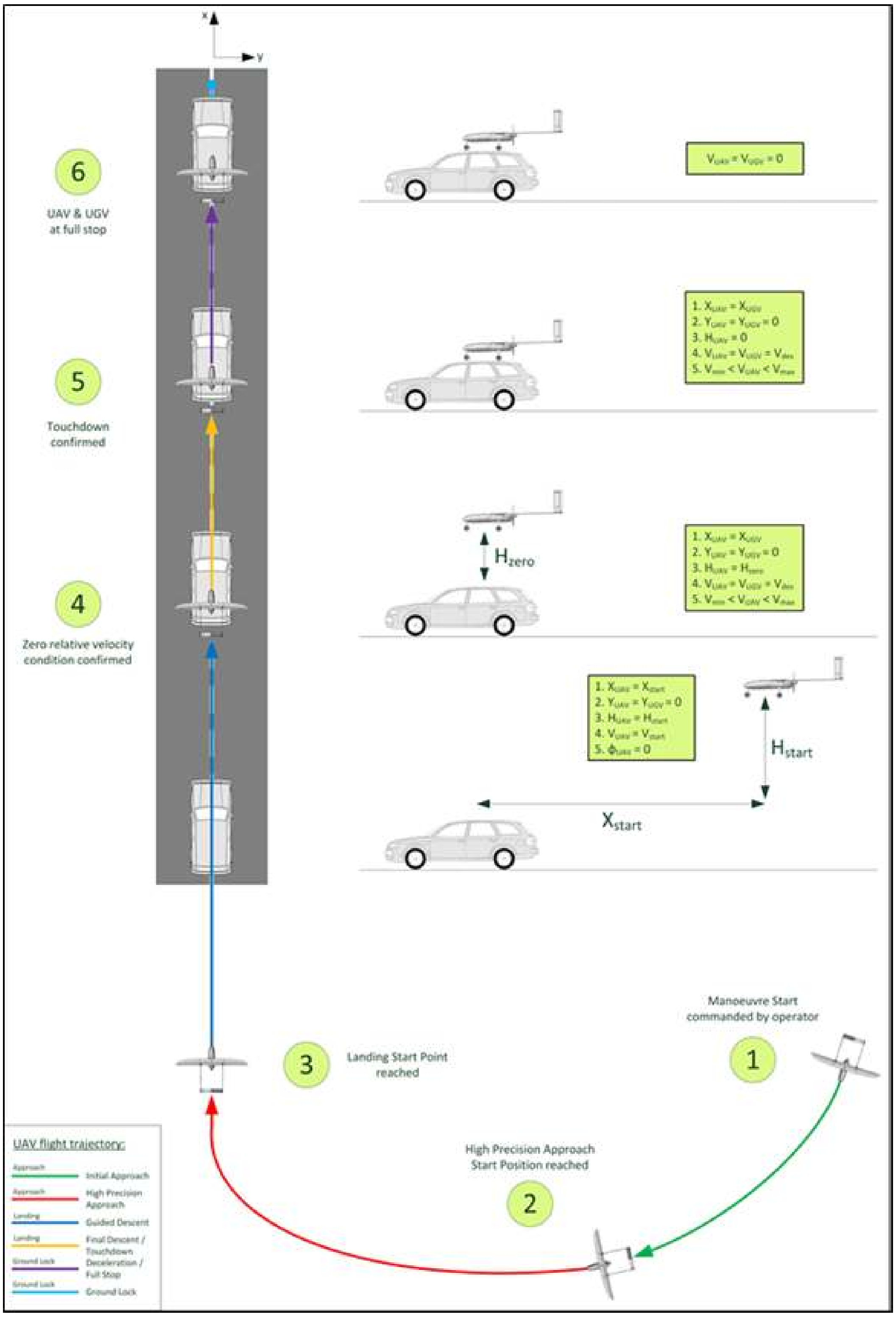

Figure 1.1.: The landing procedure, divided into 6 phases. (Figure reprinted from [1])

particularly phases 2 to 4 require precise and independent control of all degrees of freedom.

1.2. Scope and Objectives

The main goal of this thesis is to develop a flight control system that is suitable for landings on mobile platforms as envisioned by the DLR project. The DLR Flying Robots Group has previously used cascaded SISO control systems on their fixed-wing aircraft, but this approach was found to be insufficiently precise for this landing application, mainly due to the coupling of altitude and airspeed, which makes the independent control of airspeed and glide path difficult. This independence is required, because the airspeed is used to control the aircraft's position relative to the ground vehicle, but control action in the airspeed shall not interfere with the altitude tracking, as this would result in an unsteady approach and possibly an unsafe landing. Also the opposite case, where the altitude control influences the airspeed, and thus the relative position, is to be avoided.

In order to assist the development of the flight control system, a simulation model of the UAV is to be created. The model shall be reasonably accurate to allow initial testing and tuning of the controller before the flight experiments. It may also be used for further simulations later in the project, e.g. for testing the cooperative control in landing simulations together with a model of the ground vehicle.

1.3. System Description

1.3.1. The Penguin UAV

The aircraft used in this project is the Penguin BE, an electric UAV produced by UAV Factory Ltd[2]. It is a high-wing aircraft with a negative-V-tail and it is propelled by a geared brushless DC motor and a 19x11 inch propeller in push configuration. It has a wing span of 3.3 m and a take-off mass of up to 21.5 kg. A photo of the aircraft is shown in Figure 1.2. More specifications and some performance data is provided in Appendix A.

The aircraft was equipped by DLR with the necessary sensors, including differential GPS, an inertial measurement unit (IMU), and an air data probe, which provides airspeed and barometric altitude. Wireless LAN is used for communication with the ground station as well as the ground vehicle. A real-time computer

[2]Company website: `http://www.uavfactory.com/`

is used for navigation and control; with the control software being automatically generated from Matlab/Simulink models.

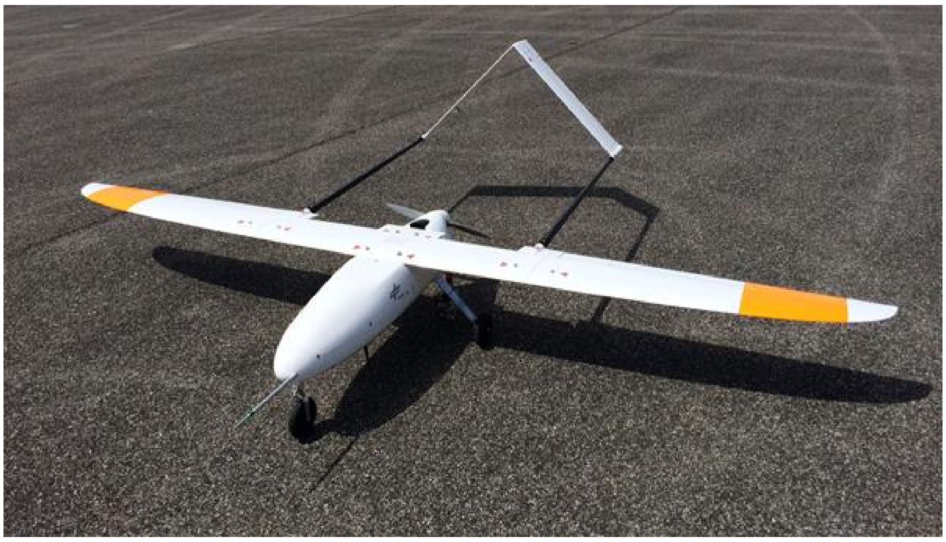

Figure 1.2.: Photo of the Penguin UAV, taken during the flight tests.

1.3.2. The Ground Vehicle

The ground vehicle, at this stage, consists of a car with a human driver and a top-mounted landing platform. The landing platform is 5 m wide and 4 m long and consists of a frame holding a net under tension. The car is equipped with a differential GPS and with a communication link to the UAV. A real-time computer processes the position and velocity data obtained from the GPS and the state vector of the UAV to generate a command for the ground vehicle, which is displayed to the driver in a graphical user interface. The driver is regarded as an actuator in this system.

This setup has been tested, including landings with a small and inexpensive remotely piloted aircraft. The platform was stable even at high speeds, and landings were possible despite the car's influence on the airflow. CFD simulations were performed to determine the airflow around the car, and this data is used to enhance the MATLAB simulations of the autonomous landings.

In a future stage, the ground vehicle is envisioned to be autonomous as well, potentially based on DLR's RoboMobil (ROMO)[3]. The landing platform may be actuated in pitch and/or roll to allow for landings with various attitudes, e.g. in strong crosswind.

[3]ROMO website: `http://www.dlr.de/rmc/rm/desktopdefault.aspx/tabid-8039`

2. System Modelling

This chapter is divided into three sections, which correspond to the three main steps taken in the modelling process: flight dynamics modelling, implementation and validation; as illustrated in Figure 2.1.

Modeling of flight dynamics
Implementation in Matlab/Simulink
Validation with flight experiments

Figure 2.1.: The three steps of the modelling process.

The first section focuses on the mathematical modelling of the aircraft dynamics, which is the core of the model. This includes modelling the aerodynamic forces and moments, the propulsion system and the aircraft's inertia properties. The second section describes the practical implementation of the mathematical model in Matlab/Simulink as well as a number of additional elements to model the environment, including the atmosphere and earth. In the third section, the complete simulation model is compared to data from test flights to validate the model and get a qualitative understanding of its accuracy and potential deficiencies.

2.1. Flight Dynamics Modelling

2.1.1. Aerodynamics Model

Determining the aerodynamic properties of aircraft is a task that has been studied extensively since the advent of aviation. High-fidelity models are produced even before the first prototype leaves the production halls. However, the creation of such models typically involves expensive wind tunnel experiments and intensive numerical calculations. The required human, technical and financial resources greatly exceed the typically available means for projects involving small UAV. A relatively simple approach leading to results with reasonable accuracy has been presented in [2]. The aerodynamics model of the Penguin UAV was created based on said approach.

Method

A freely available potential flow solver named Athena Vortex Lattice (AVL) [3], was used to model the aerodynamics. As the name suggests, it is an implementation of the vortex lattice method, which is unable to model viscous effects. The airfoil's drag polars were calculated separately in XFOIL [4] and integrated into the AVL calculation to account for the viscous drag.

The only input required for this method is the aircraft geometry, which was kindly provided as a CAD file by the aircraft manufacturer[1]. The geometry is then modelled in AVL as a combination of thin airfoils for all lifting surfaces and slender bodies for the fuselage. Control surfaces are defined as well and will be included in the calculations.

As the aerodynamics model is intended to be used in real-time simulations, where it will not be possible to run the AVL solver in every time step, the AVL routine is used to generate a simpler model, where the aerodynamic coefficients are expressed as polynomials in the state variables $\alpha, \beta, \hat{p}, \hat{q}, \hat{r}$ and the input variables $\delta_a, \delta_e, \delta_r, \delta_f$. α and β denote the angle of attack and the sideslip angle, $\hat{p}, \hat{q}, \hat{r}$ are the dimensionless angular rates around the aircraft axes, and $\delta_a, \delta_e, \delta_r, \delta_f$ denote the control surface deflection of ailerons, elevator, rudder and flaps.

A set of values in those 9 variables defines an input for an AVL calculation. 10 000 such inputs were generated randomly, with each variable being bounded as defined in Table 2.1, and with a cosine-like probability distribution of the values, to increase the number of samples near the normal flight regime. The AVL model was then evaluated at all inputs, computing the aerodynamic coefficients $C_D, C_Y, C_L, C_l, C_m, C_n$ for every input. These results were then fitted in a least-square sense with multi-dimensional polynomials in the 9 input variables. Figure 2.2 shows a simplified example of this method.

Table 2.1.: The value ranges of the AVL input variables.

Angles	Rates	Controls
$\alpha \in [-10°, 15°]$	$\hat{p} \in [-0.10, 0.10]$	$\delta_a, \delta_e, \delta_r \in [-20°, 20°]$
$\beta \in [-20°, 20°]$	$\hat{q} \in [-0.03, 0.03]$	$\delta_f \in [\ \ 0°, 30°]$
	$\hat{r} \in [-0.25, 0.25]$	

Monomial Selection

The selection of the relevant monomials is important to ensure a good fit of the polynomials. An iterative process was used to discover the relevant monomials.

[1]UAV Factory Ltd., Jelgava, Latvia, `http://www.uavfactory.com/`

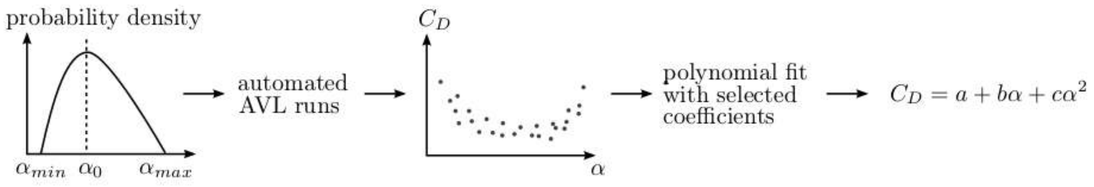

Figure 2.2.: The process of the geometry-based aerodynamics estimation. (Figure reprinted from [2])

In a first step, the results of the AVL computations are plotted against the input variables, to find the most important dependencies. An example is shown in Figure 2.3, where the linear dependency of C_L on α is clearly visible, as opposed to Figure 2.4, where no obvious dependency of C_L on δ_a can be detected. After establishing these strong dependencies, the resulting polynomials are calculated. For each subsequent step, the residuals of the polynomial fit are plotted against the input variables, which brings out the weaker dependencies. Colouring is useful to detect coupled effects in two or more variables.

As an example, in Figure 2.5 the residual of the lift coefficient approximation after one iteration ($C_{L_{AVL}} - C_{L_{poly-it1}}$) is plotted against the aileron deflection (δ_a), and unlike in Figure 2.4, a pattern is now visible, and it seems that a large aileron deflection can influence the lift coefficient. Whether the lift is increased or decreased depends on a second variable, $\hat{r}$ as is shown with the colouring in Figure 2.6. Therefore the monomial $\hat{r}\delta_a$ is added to the polynomial for C_L in the next iteration of the polynomial fitting. Figure 2.7 shows another example, where the colouring reveals a quadratic dependency of C_L on β coupled linearly with α.

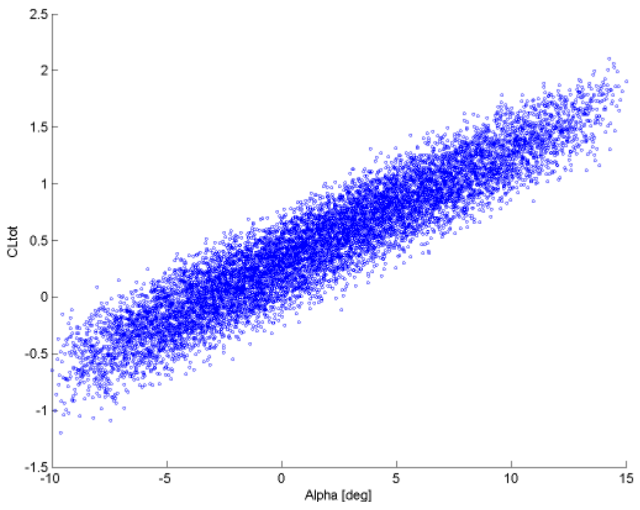

Figure 2.3.: C_L versus α. The strong, linear dependency of the lift coefficient on the angle of attack is clearly visible, and a term $C_{L_\alpha}\alpha$ will be included in the polynomial approximation of C_L.

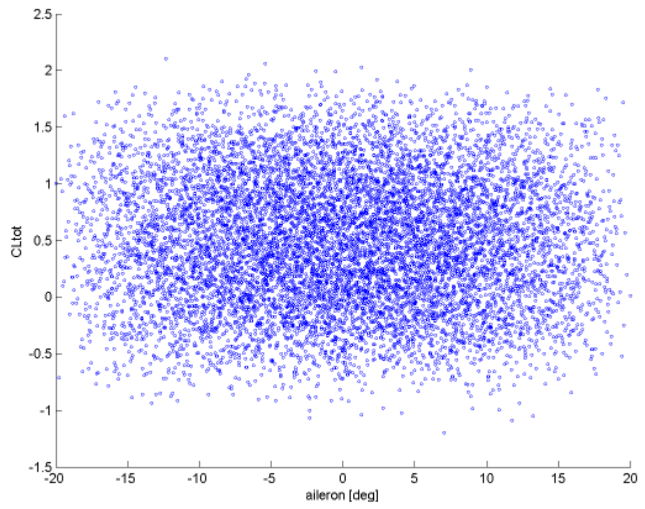

Figure 2.4.: C_L versus δ_a. As expected, the lift coefficient does not seem to depend on the aileron deflection.

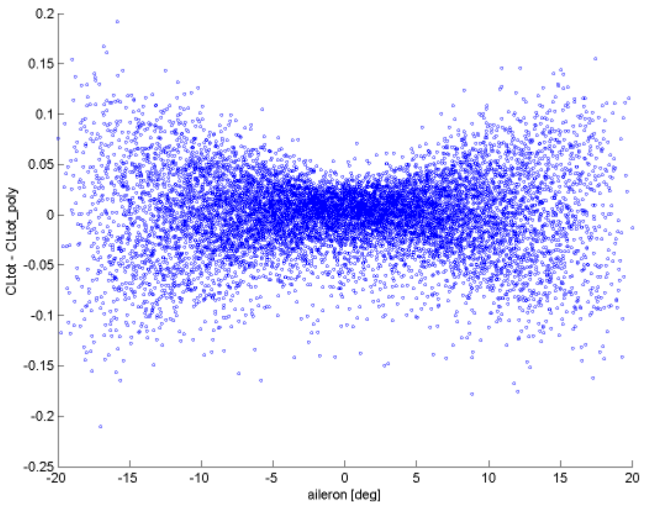

Figure 2.5.: $(C_L - C_{L_{itr1}})$ versus δ_a. Plotting the residuals after a first polynomial approximation shows previously undetected dependencies. While the aileron deflection seemed to have no influence on the lift coefficient in Figure 2.4, this plot seems to indicate some relation. However, the relation is not quite clear until adding a second variable in Figure 2.6.

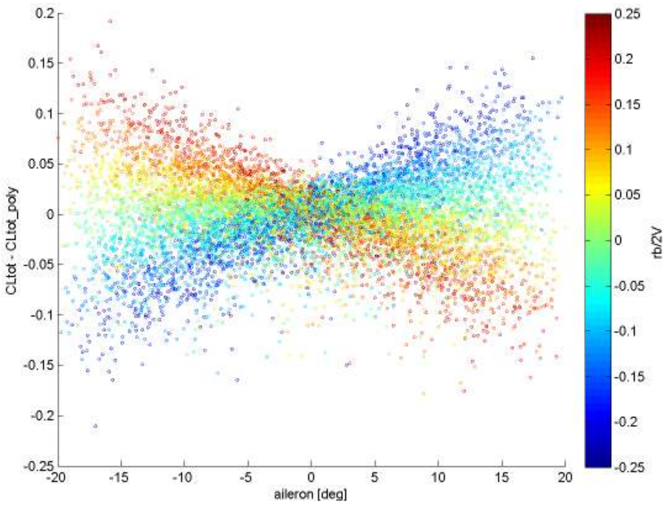

Figure 2.6.: $(C_L - C_{L_{itr1}})$ versus δ_a, coloured with $\hat{r}$. Adding colour to Figure 2.5 reveals a mixed dependency of C_L on δ_a and $\hat{r}$, and a term $C_{L_{\delta_a \hat{r}}} \delta_a \hat{r}$ needs to be added to the polynomial approximation of C_L to account for this.

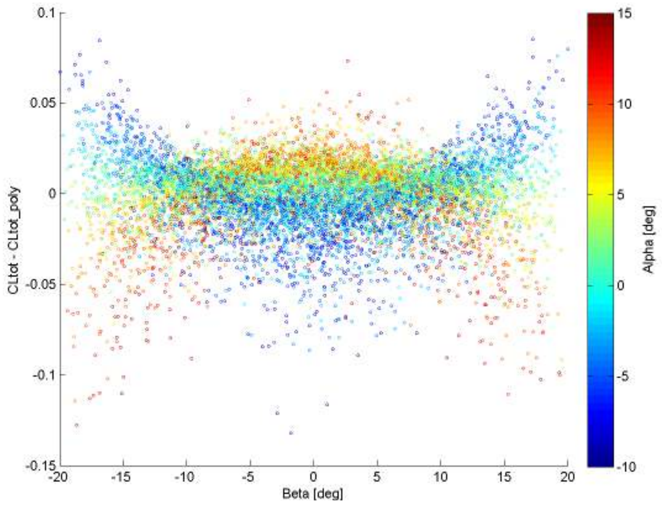

Figure 2.7.: $(C_L - C_{L_{itr2}})$ versus β, coloured with α. The colouring reveals the parabolic shape, indicating a dependency on β^2, with α determining the curvature of the parabola. Therefore, a term $C_{L_{\alpha\beta^2}}\alpha\beta^2$ is added to the polynomial approximation of the lift coefficient.

Result

After six iterations, the resulting polynomials were deemed sufficiently accurate. Figure 2.8 shows how the fit of the polynomials improved with every iteration. After the last iteration, the normalized root-mean-square deviations were around 10% or less, as listed in Table 2.2. The monomials included in the final version of the polynomial approximations are listed in Table 2.3.

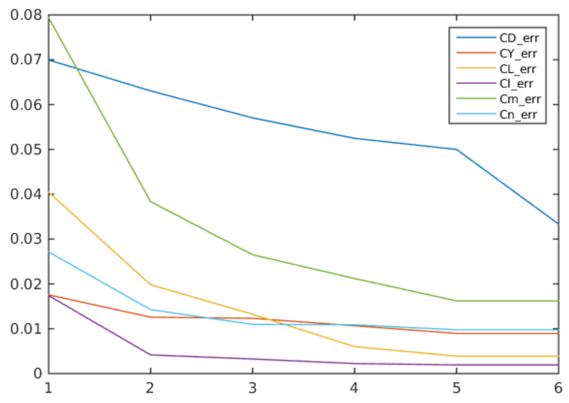

Figure 2.8.: The root-mean-square deviations of the polynomials approximating the aerodynamic coefficients. Each subsequent iteration adds a few monomials to the polynomial, thus improving the fit to the data. See also Table 2.2.

The model has to be used with caution, as it does not account for unsteady flow or turbulent flow. Therefore it does not accurately represent the real flight dynamics in highly dynamic manoeuvres (aerobatics) and at high angles of attack (stall). Compressive flow is also not considered, however that is not expected to have a significant effect due to the Penguin's low speed ($M < 0.15$). For the purpose of this thesis, namely the design and initial tuning of a flight control system for automatic landings, the model is considered to be sufficiently accurate, as the expected flight conditions are well within these boundaries.

Table 2.2.: The root-mean-square deviations, both absolute (RMSD) and normalized with the value range (NRMSD), of the polynomial approximations of the aerodynamic coefficients after 6 iterations. See also Figure 2.8.

Value	Range	RMSD	NRMSD
C_D	$-0.03 \dots 0.68$	0.0333	4.7%
C_Y	$-0.27 \dots 0.26$	0.0089	6.3%
C_L	$-0.78 \dots 1.79$	0.0039	1.3%
C_l	$-0.17 \dots 0.18$	0.0019	9.5%
C_m	$-1.42 \dots 1.80$	0.0162	1.0%
C_n	$-0.15 \dots 0.17$	0.0098	10.3%

Table 2.3.: The dependencies listed here were considered in the final version of the polynomial approximations of the aerodynamic coefficients.

$$\begin{aligned}
C_D &= f\Big(1, \alpha, \hat{q}, \delta_f, \alpha^2, \beta^2, \hat{q}^2, \delta_a^2, \delta_e^2, \delta_r^2, \delta_f^2, \alpha\hat{q}, \beta\hat{r}, \hat{p}\delta_a, \hat{q}\delta_e, \alpha\delta_f, \\
&\qquad \alpha^3, \alpha\delta_a^2, \alpha^2\delta_f, \alpha\hat{r}\delta_a, \alpha^4, \alpha^2\delta_a^2, \alpha^3\hat{r}\delta_a\Big) \\
C_Y &= f\Big(\beta, \hat{p}, \hat{r}, \delta_a, \delta_r, \alpha\beta, \alpha\hat{p}, \beta\hat{q}, \hat{q}\hat{r}, \hat{r}\delta_e, \hat{q}\delta_r, \beta^3, \alpha^2\beta, \beta\delta_a^2, \alpha\beta^3, \alpha^2\beta^3\Big) \\
C_L &= f\Big(1, \alpha, \hat{q}, \delta_e, \delta_f, \alpha^2, \beta^2, \hat{q}^2, \hat{r}^2, \alpha\hat{q}, \hat{p}\hat{r}, \alpha\delta_e, \hat{q}\delta_e, \hat{r}\delta_a, \hat{r}\delta_r, \alpha^3, \alpha\beta^2, \alpha\hat{r}^2, \beta^2\delta_e, \beta^2\delta_f\Big) \\
C_l &= f\Big(\beta, \hat{p}, \hat{r}, \delta_a, \delta_r, \alpha\beta, \alpha\hat{r}, \alpha\delta_a, \hat{q}\delta_a, \alpha\delta_r, \beta^2\delta_a\Big) \\
C_m &= f\Big(1, \alpha, \hat{q}, \delta_e, \delta_f, \beta^2, \hat{q}^2, \delta_a^2, \delta_e^2, \alpha\hat{q}, \beta\hat{r}, \hat{r}\delta_a, \hat{q}\delta_e, \beta\delta_r, \hat{r}\delta_r, \alpha\beta^2, \beta^2\hat{q}, \beta^2\delta_e, \hat{q}^2\delta_e\Big) \\
C_n &= f\Big(\beta, \hat{p}, \hat{r}, \delta_a, \delta_r, \alpha\hat{p}, \alpha r, \alpha\delta_a, \hat{q}\delta_a, \hat{r}^3, \delta_a^3, \alpha^2\hat{r}, \alpha^2\delta_a, \hat{r}\delta_a^2, \\
&\qquad \alpha\hat{r}^3, \alpha^3\delta_a, \alpha\delta_a^3, \alpha^2\hat{r}^3, \hat{r}^3\delta_a^2, \alpha^2\delta_a^3, \alpha^3\delta_a^3\Big)
\end{aligned}$$

2.1.2. Propulsion Model

Propeller

Propeller aerodynamics are typically described using a thrust coefficient C_T and a power coefficient C_P, which are defined as

$$C_T = \frac{T}{\rho\, n^2 d^4} \qquad \text{and} \qquad C_P = \frac{P}{\rho\, n^3 d^5} \tag{2.1}$$

where ρ is the atmospheric density, n is the rotational speed, d the propeller diameter and T and P the thrust and power, respectively. The coefficients C_T and C_P are in general functions of the propeller geometry, the advance ratio $J = \frac{V}{nd}$, the rotational speed and the Reynolds number, however the influence of the latter two is often neglected. The propeller efficiency can be expressed in terms of C_T, C_P and J as follows:

$$\eta = \frac{TV}{P} = \frac{C_T J}{C_P} \tag{2.2}$$

In practice, the values for C_T and C_P can be determined experimentally, e.g. as described in [5], or they can be approximated based on the propeller geometry, in particular the pitch/diameter ratio, as described in [2]. The Penguin UAV is equipped with a APC 19x11 propeller. Its performance data was kindly provided by the propeller manufacturer[2] as tables of $C_T(J)$ and $C_P(J)$ for different rotational speeds. Figure 2.9 gives an overview of the propeller performance data.

The existing fixed-pitch propeller block in the AeroSim library was modified to include 2-dimensional look-up tables for C_T and C_P with the data provided by APC. The propeller block, shown in Figure 2.10, calculates the propeller thrust and torque in function of airspeed, air density and propeller rotational speed.

Motor

As no reliable technical details of the motor were available, the modelling of the motor relied on test data. The power consumption for a range of throttle settings was calculated based on voltage and current measurements from static thrust tests. A look-up table with these power values was created, assuming that the mechanical power output of the motor is equal to its electrical power input, i.e. 100% efficiency. As modern brushless DC motors typically have efficiencies of over 90%, this assumption is considered reasonable. The output torque is calculated as $\tau = \frac{P}{\omega}$. Figure 2.11 shows the Simulink implementation of the motor model.

2.1.3. Mass and Inertia

Compared to a fuel-consuming aircraft, the mass and inertia of an electric aircraft are relatively easy to model, as these properties are constant throughout the flight. Mass and centre of gravity can quite simply be measured; the moments of inertia however are more difficult to determine. Measuring them is a rather complicated endeavour and beyond the scope of this thesis. Instead, the moments of inertia were estimated using a method suggested in [2]. The masses of all components are measured, and the structures are approximated as hollow hulls (fuselage) or flat plates (wings and tail) with constant mass per surface. The other components are modelled as point masses in the right locations. This results in a mass distribution, from which the moments of inertia can easily be calculated.

[2]Landing Products Inc., Woodland, CA, USA, `http://www.apcprop.com/`

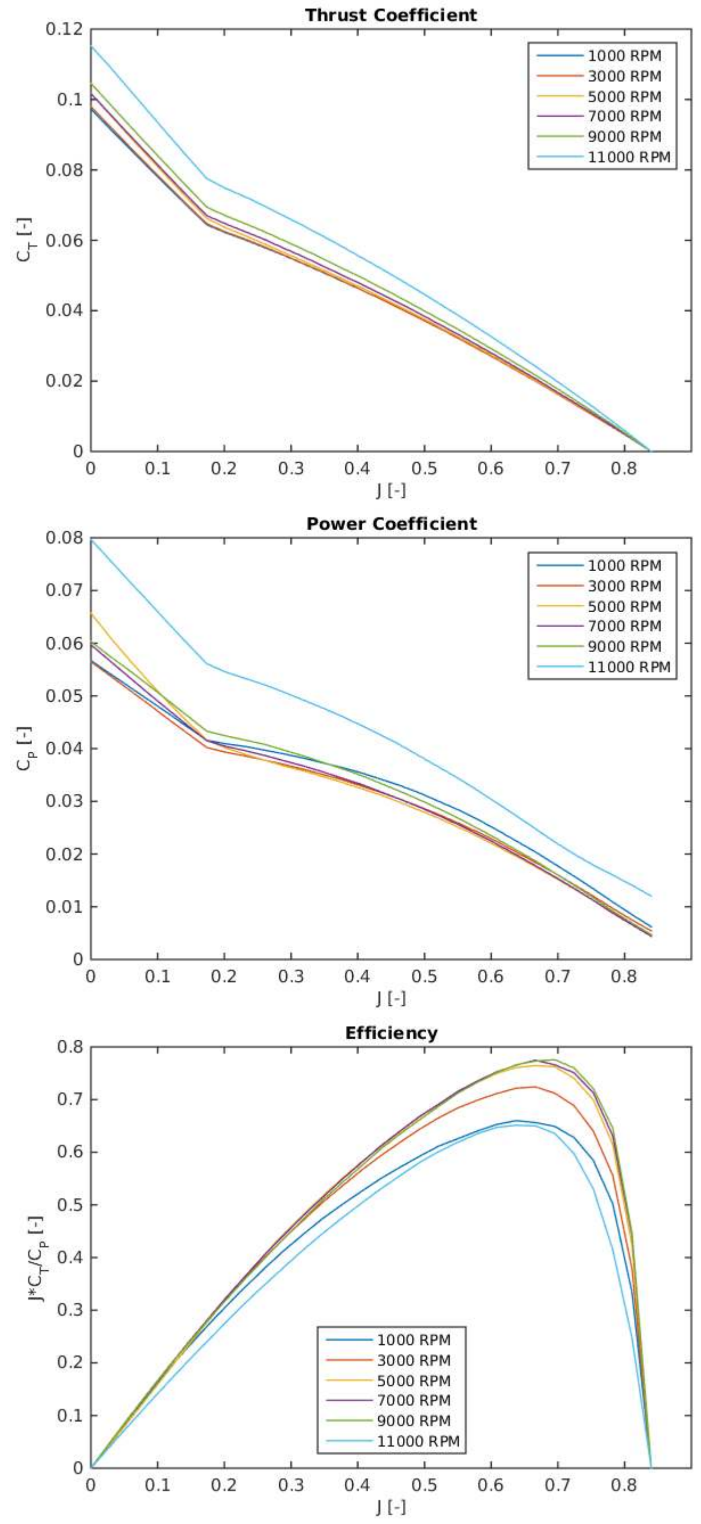

Figure 2.9.: The performance data of the APC 19x11 propeller as provided by the manufacturer. See Equations 2.1 and 2.2 for more details.

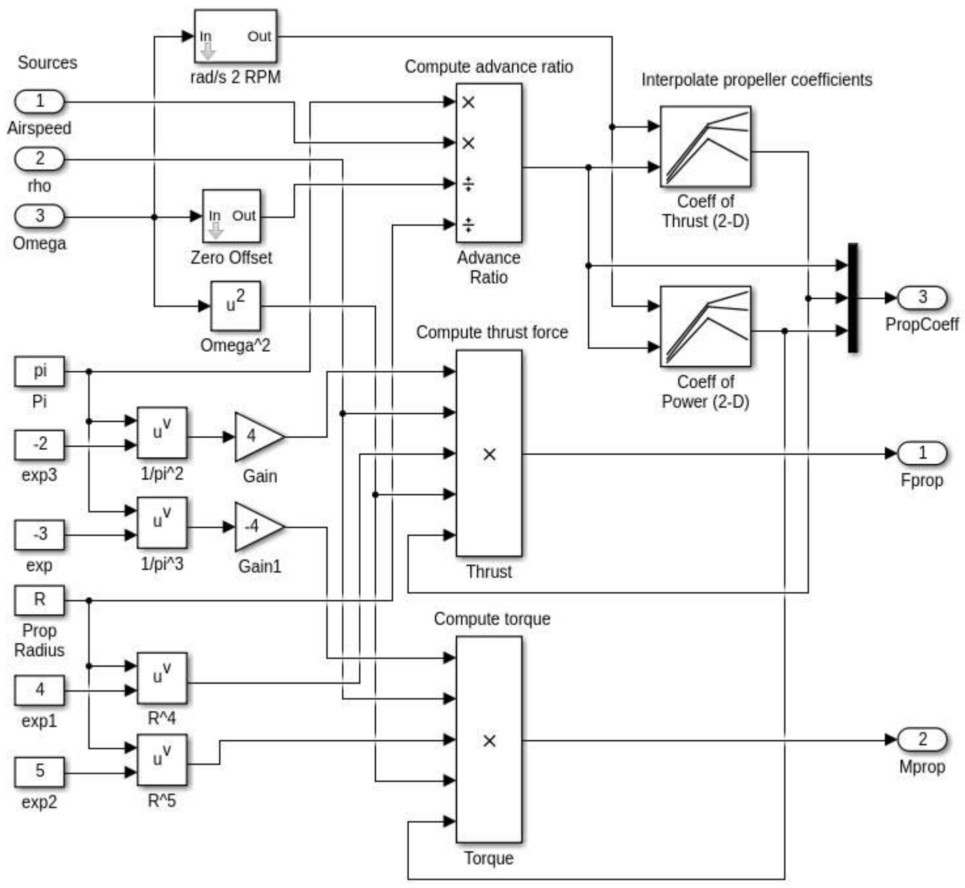

Figure 2.10.: The *Fixed-pitch Propeller* model as implemented in Simulink. Propeller thrust and torque are calculated based on propeller coefficients, rotational speed and air density according to Equation 2.1. The propeller coefficients are provided by look-up tables.

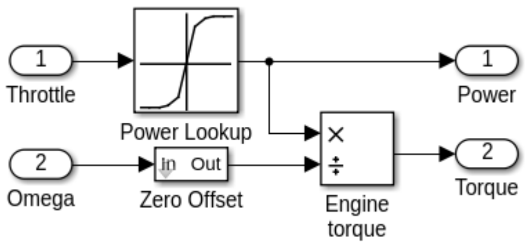

Figure 2.11.: The motor model in Simulink, with a look-up table for the power, which is based on test data.

2.2. Matlab/Simulink Model

2.2.1. General Model Description

The Matlab/Simulink model was built using the AeroSim Blockset [6], which is a Simulink block library providing tools for the development of nonlinear 6-DOF aircraft models. It provides earth and environment models, such as atmosphere, wind, gravity and magnetic field, which were mostly used without changes, and flight dynamics blocks which were modified or replaced in order to properly integrate the nonlinear aerodynamics model from Section 2.1.1 and the electric propulsion model from Section 2.1.2, as the original AeroSim blocks are designed for piston engines and linearized aerodynamics only.

A top-level overview of the Simulink model is given in Figure 2.12. The following sections will give details about the different parts of the model, in particular the implementation of the flight dynamics model from Section 2.1, but also a description of the earth and atmosphere models.

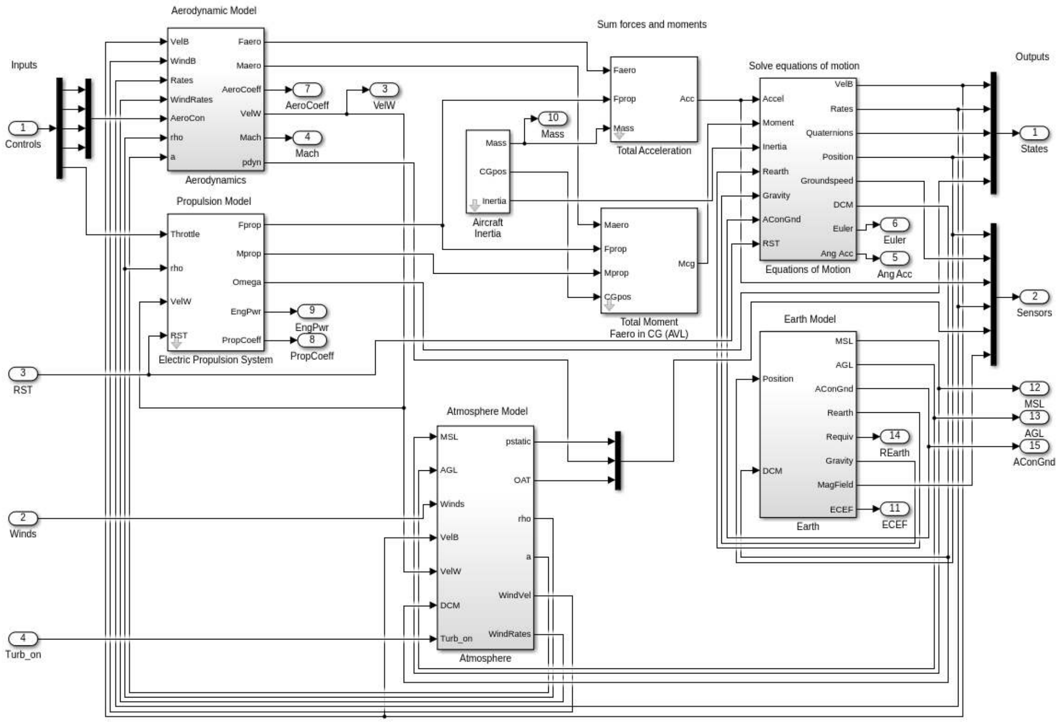

Figure 2.12.: Top-level overview of the Penguin Simulink model. In the upper left corner are the aircraft specific flight dynamics blocks, on the upper right are the general equations of motion. The earth and atmosphere models are in the lower half.

2.2.2. Aerodynamics Model

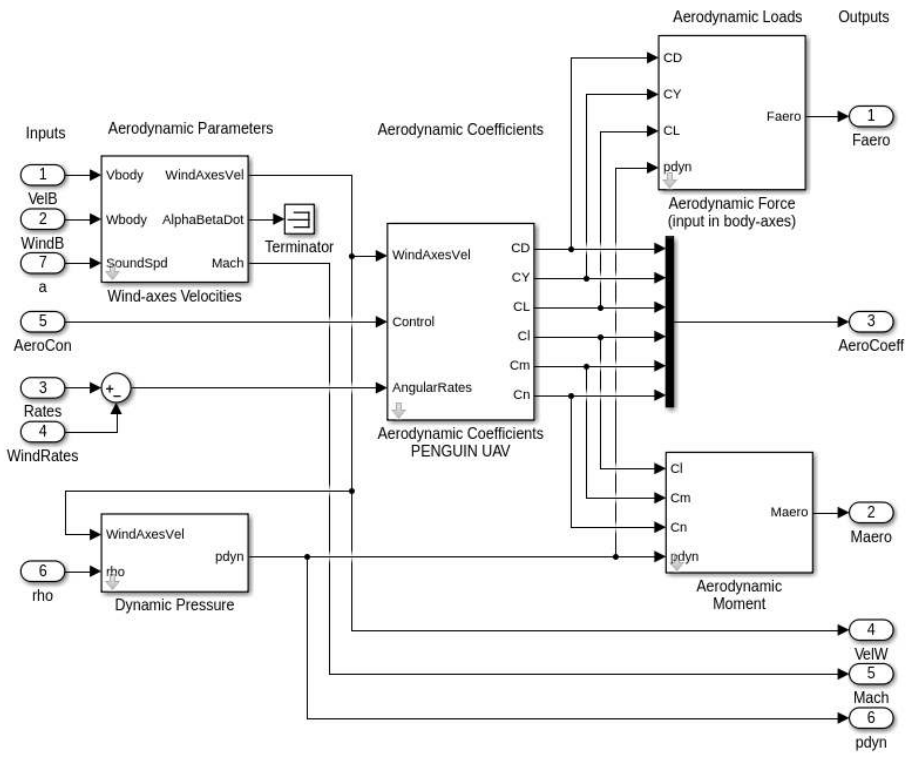

Figure 2.13.: The *Aerodynamics* block, with the aerodynamic coefficients calculated in the centre and converted into forces and moments on the right.

The *Aerodynamics* block is shown in Figure 2.13. Its core is formed by the polynomial equations for the aerodynamic coefficients, which are implemented as an 'Embedded Matlab' block. The input to this block consists of the 9 variables of the polynomials, i.e. the wind angles α, β; the angular rates $\hat{p}, \hat{q}, \hat{r}$ and the aerodynamic controls $\delta_a, \delta_e, \delta_r, \delta_f$. α and β are calculated in the *Wind-axes Velocities* block according to

$$\alpha = \arctan\left(\frac{V_z - W_z}{V_x - W_x}\right) \qquad \beta = \arcsin\left(\frac{V_y - W_y}{\|V - W\|}\right) \tag{2.3}$$

where V is the aircraft's velocity and W is the wind velocity, or in other words, $V - W$ is the aircraft's velocity relative to the surrounding air. Similarly, the angular rates are calculated as the difference of the aircraft rates and the wind rates. The transformation to non-dimensional rates is included in the *Aerodynamic Coefficients* block.

From the aerodynamic coefficients, the aerodynamic forces and moments are calculated according to

$$F_x = -C_D\bar{q}S \qquad F_y = C_Y\bar{q}S \qquad F_z = -C_L\bar{q}S \tag{2.4}$$

and

$$M_x = C_l\bar{q}Sb \qquad M_y = C_m\bar{q}Sc \qquad M_z = C_n\bar{q}Sb \tag{2.5}$$

where $\bar{q}$ denotes the dynamic pressure ($\bar{q} = \frac{1}{2}\rho(V-W)^2$), S the reference wing area, b the wing span and c the mean aerodynamic chord. The resulting forces and moments, expressed in body axes, are the main output of the *Aerodynamics* block.

2.2.3. Propulsion Model

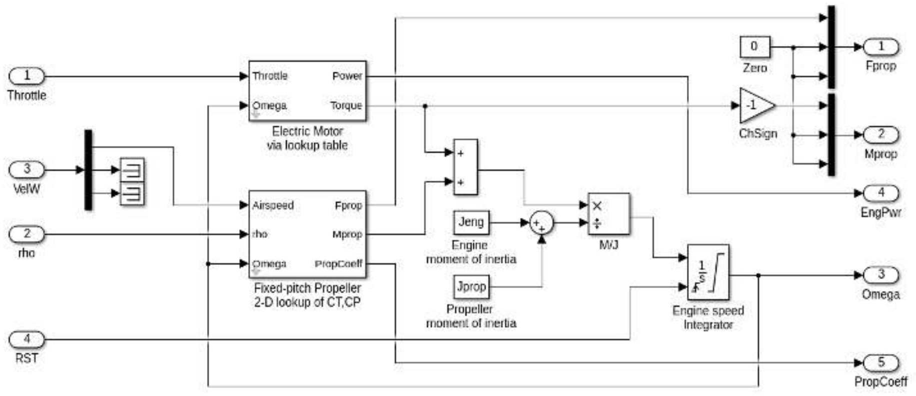

Figure 2.14.: The *Electric Propulsion System* block, with a dynamic model of the propeller rotation and the motor and propeller blocks described earlier.

The *Electric Propulsion System* block, shown in Figure 2.14, includes the propeller and motor models described in Section 2.1.2 and a dynamic model of the propeller rotation, which is described by the following differential equation:

$$(J_{eng} + J_{prop})\,\dot{\omega} = \tau_{eng} - \tau_{prop} \tag{2.6}$$

The motor's moment of inertia J_{eng} is neglected as it significantly smaller than the propeller's moment of inertia J_{prop}. The torques τ_{eng} and τ_{prop} are calculated in the motor and propeller block, respectively, and of course depend on the propeller's rotational speed ω, thus forming a feedback loop. The propeller's thrust force is calculated in the *Fixed-pitch Propeller* block, and together with the motor torque it forms the output that is passed on to the equations of motion.

2.2.4. Equations of Motion

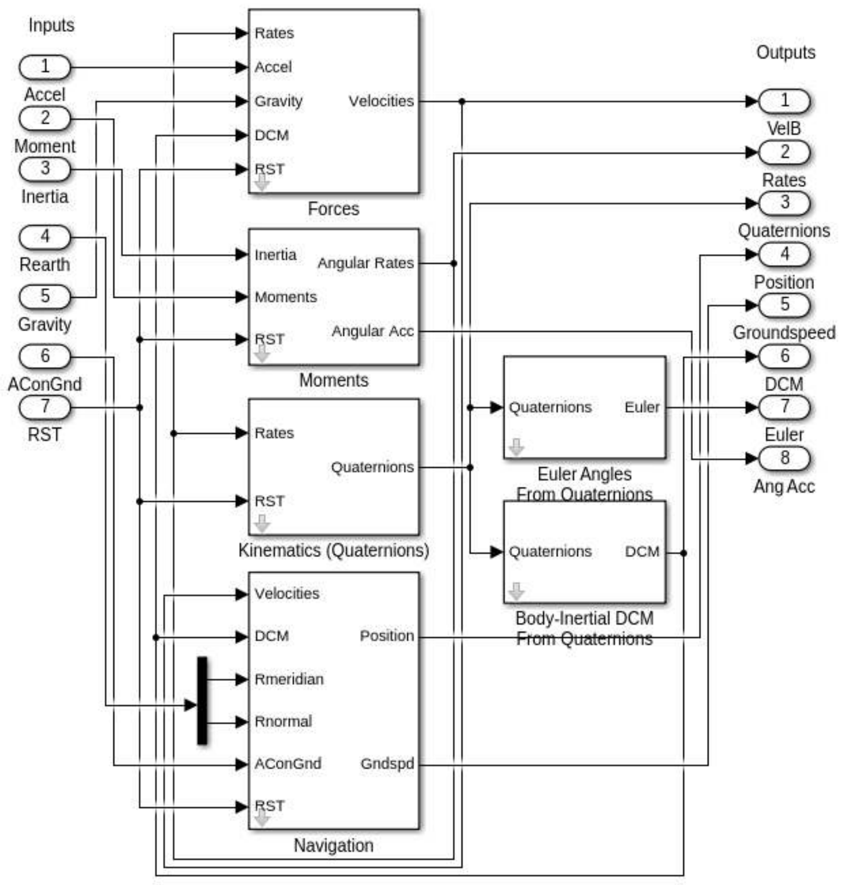

Figure 2.15.: The *Equations of Motion* block. The four main blocks seen on the left each contain a set of differential equations: one block each for velocities, angular rates, quaternions (attitude) and position.

The *Equations of Motion* block groups the differential equations describing the aircraft motion together. These are general, aircraft independent equations; the aircraft specific flight dynamics have been described earlier and are inputs to this block in the form of total force, total moment and moments of inertia.

The AeroSim Blockset provides different versions of the equations of motion, with the main difference being the coordinate system used. The version selected for the Penguin model calculate velocities and angular rates in an aircraft-fixed coordinate system. The aircraft attitude is calculated in quaternions, and the position is calculated in the WGS 84 coordinate system (spheroidal earth).

2.2.5. Earth Model

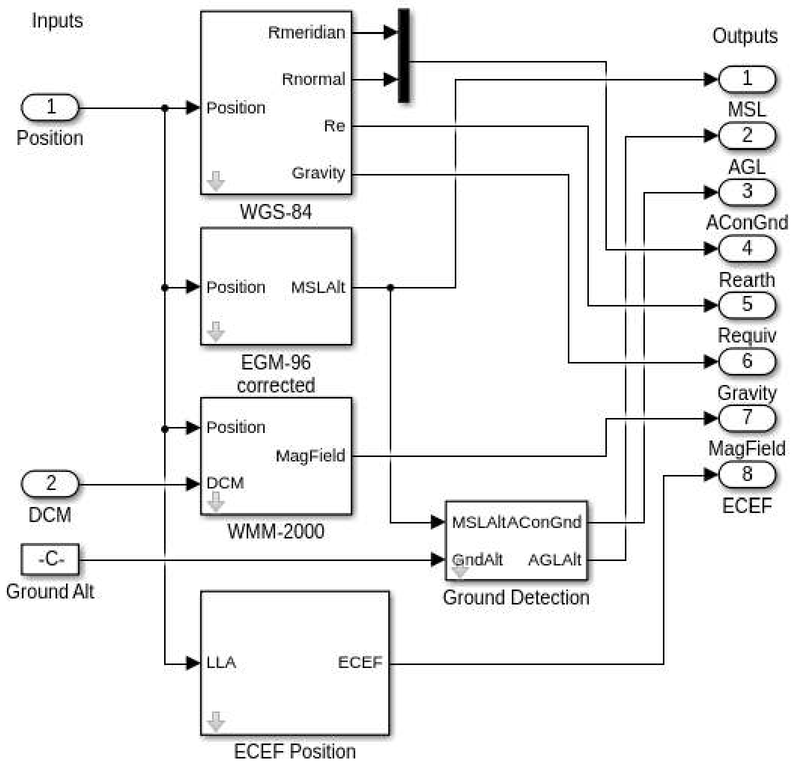

Figure 2.16.: The *Earth* block, consisting of the WGS 84 World Geodetic System, the EGM 96 Earth Gravitational Model the WMM World Magnetic model, a block to convert the position to a earth centred, earth fixed coordinate system, and a block to detect collision with the ground.

The main purpose of the *Earth* block, shown in Figure 2.16, is to calculate the local gravitational acceleration and the local earth radii used for transformations from local to global coordinates and vice versa, particularly in the *Navigation* block inside *Equations of Motion*. These calculations are based on the World Geodetic System 1984 (WGS84), defined in [7].

Furthermore, the altitude above mean sea level according to the Earth Gravitational Model 1996 (EGM96, also defined in [7]) is also calculated here, and compared to a user-specified ground altitude to detect collisions with the ground. Lastly, the *Earth* block provides the local magnetic field according to the World Magnetic Model [8], and the position in an earth-centred, earth-fixed reference frame.

2.2.6. Atmosphere Model

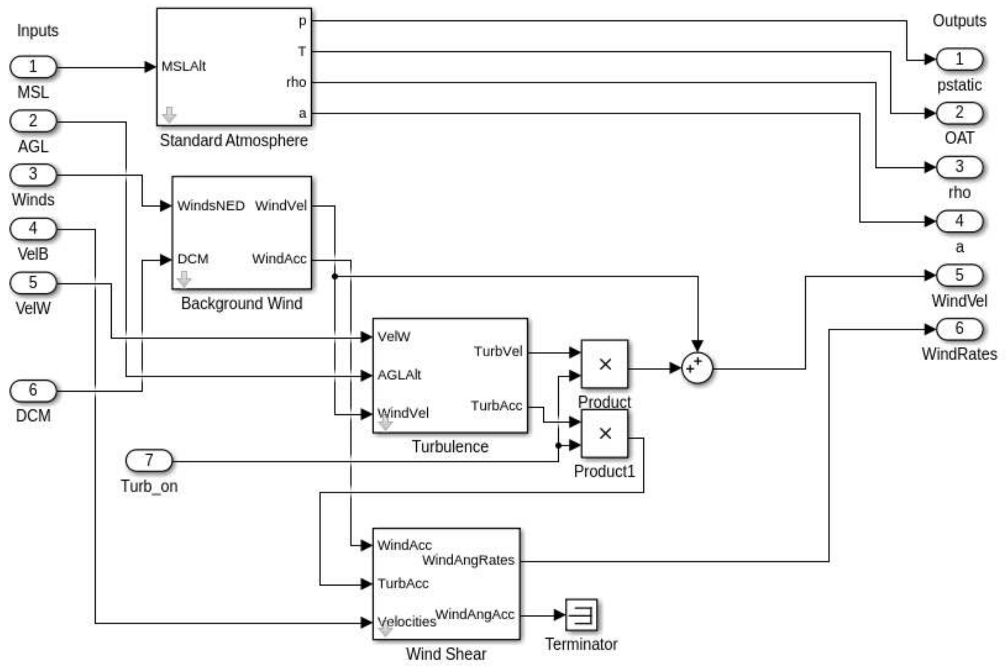

Figure 2.17.: The *Atmosphere* model, consisting of a block providing standard atmosphere data and three blocks modelling background wind, turbulence and wind shear, respectively.

The *Atmosphere* block models the properties of the atmosphere at different altitudes as well as wind, wind shear and turbulence. It consists of four parts as shown in Figure 2.17. The *Standard Atmosphere* block provides pressure, temperature, density and speed of sound according to the 1976 U.S. Standard Atmosphere [9] in the form of look-up tables with a vertical resolution of 100 m. *Background Wind* transforms the wind vector to body axes and calculates the wind acceleration, which is used for the *Wind Shear* block. The *Turbulence* block implements a von Kármán model of continuous gusts, which is commonly used in analysis and simulation of manned aircraft. *Wind Shear* calculates the angular rate effects of changes in background wind or turbulence.

When used in Simulation, the background wind is specified as an input to the aircraft model, and can be defined in any suitable way, e.g. as a function of time or aircraft position. Turbulence effects can be turned on and off by means of a switch, in order to be able to investigate the effects of continuous wind and turbulent wind separately.

2.3. Model Validation with Flight Data

The methods presented in this chapter are based on approximations and assumptions. To increase confidence in the model and get a qualitative and quantitative indication of its accuracy, simulation results were compared to log data from previously performed flight tests that were flown manually by a pilot.

In those flight tests, various manoeuvres were flown to excite the different dynamic modes of the aircraft, and some manoeuvres to test the aircraft performance. Four manoeuvres were selected and recreated in the simulation. These manoeuvres will be discussed in the following sections.

2.3.1. Phugoid Mode

The phugoid is one of the two longitudinal dynamic modes of fixed-wing aircraft. It is characterized by large oscillations in airspeed, pitch angle and altitude at a nearly constant angle of attack. It is in fact a repeated transformation of kinetic to potential energy and vice versa. The phugoid is generally poorly damped, but slow enough that it can easily be controlled by a pilot.

Figure 2.18 shows a comparison of simulation and log data of the phugoid oscillation. In both cases, the phugoid was initiated from a trimmed horizontal flight by a nose-up full elevator input of roughly 0.8 s duration. Oscillations are seen in airspeed, altitude, pitch and propeller speed. While the amplitudes do not match, this is not considered to be a problem, as the main relevant parameters are period and damping. The difference in amplitude may be caused by a slightly different excitation, for example.

Period and damping of the phugoid are shown in Table 2.4. The period matches very well, while the damping is somewhat too low in the simulation. One reason for this may be that the fuselage was not included in the aerodynamic model. The air resistance of the fuselage during pitch oscillations would certainly increase the damping of the phugoid.

Table 2.4.: Period and damping of the phugoid mode.

	Simulation	Flight	Error
Period	12.42 s	12.92 s	3.9%
Damping	0.0587	0.0933	37.1%

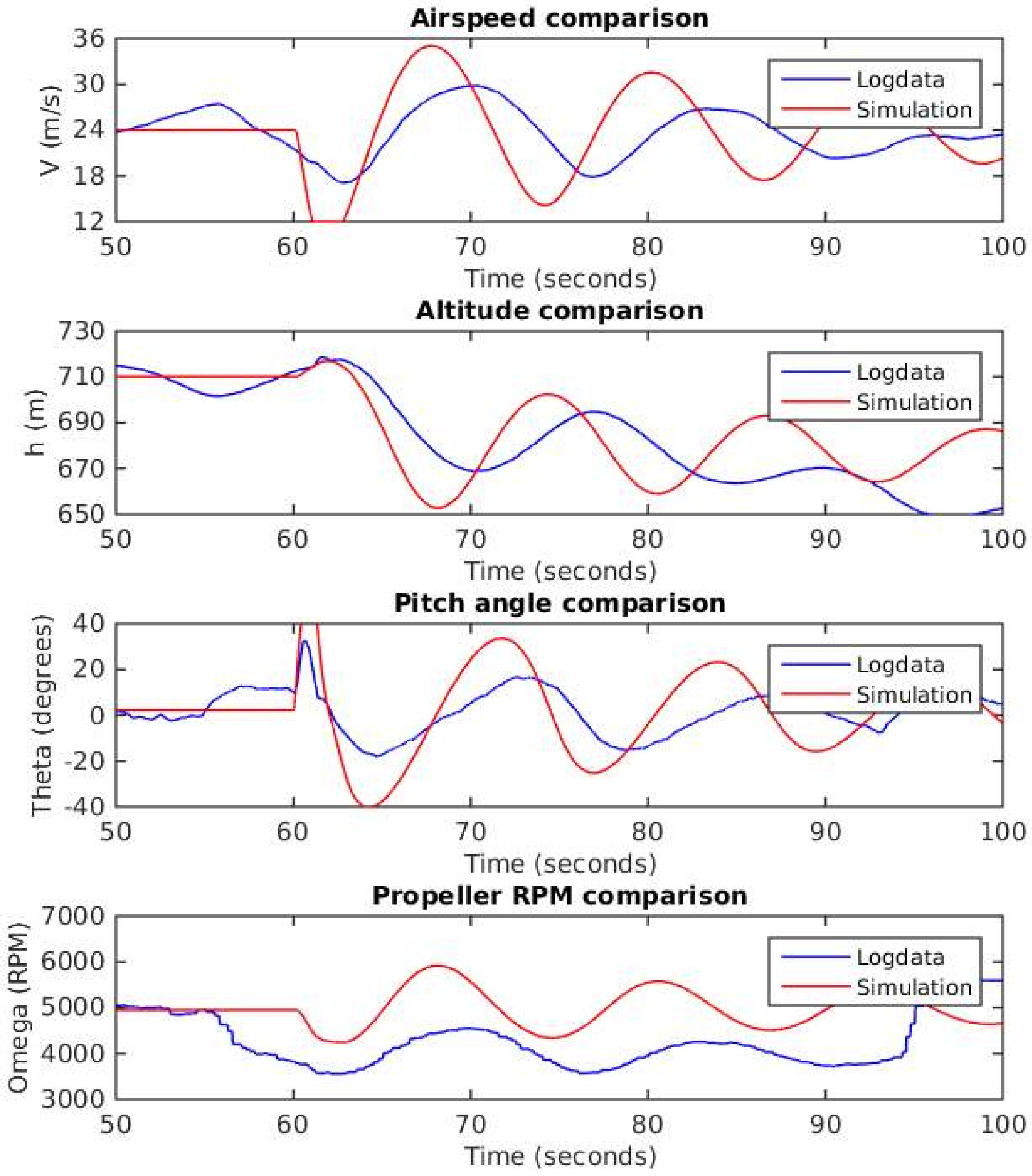

Figure 2.18.: Comparison of simulation and log data of the phugoid mode. In the flight test, the aircraft was not in an ideally trimmed state, and the throttle reduction by the pilot, seen at $t = 56\,\text{s}$, leads to a loss of altitude during the phugoid manoeuvre.

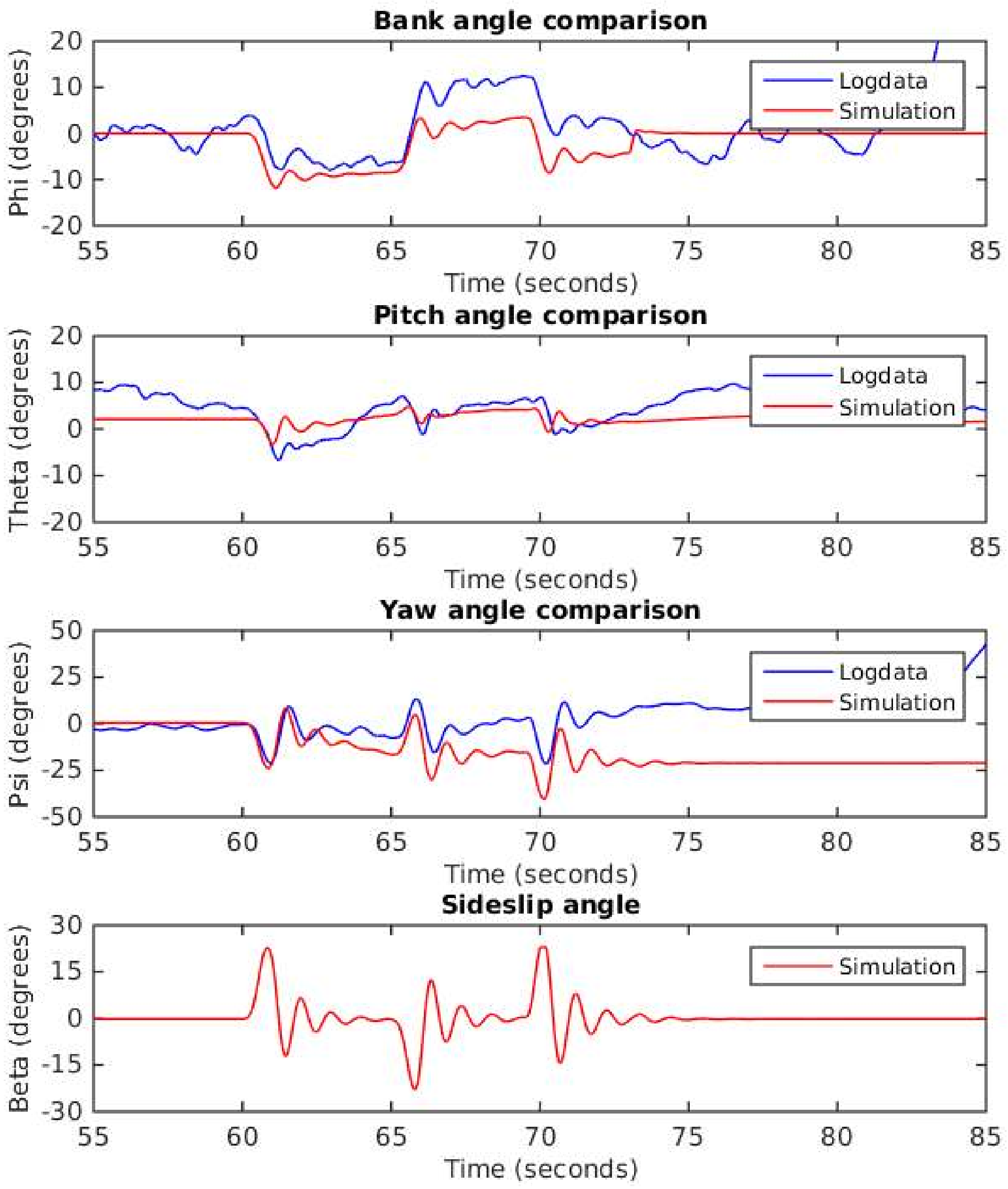

Figure 2.19.: Comparison of simulation and log data of the dutch roll mode. The aircraft is slightly unstable around the roll axis, which leads to the discrepancies in bank and yaw angle. The sideslip angle is unfortunately not measured on the Penguin UAV, hence no log data is available.

2.3.2. Dutch Roll Mode

The dutch roll is the main lateral dynamic mode of fixed-wing aircraft. It is a coupled roll-yaw motion, usually stable and lightly damped. In passenger aircraft, the dutch roll is considered unpleasant and therefore a yaw damper is usually required. In light aircraft damping is usually better.

Figure 2.19 shows a comparison of simulation and log data of the dutch roll oscillation. Three short rudder deflections were given about 5 seconds apart, each causing oscillations for a couple of seconds. The oscillation is best visible in the yaw angle. The period and damping were determined, the values are given in Table 2.5. The simulation matches the log data well, with errors of under 10%.

Table 2.5.: Period and damping of the dutch roll mode.

	Simulation	Flight	Error
Period	1.13 s	1.22 s	7.4%
Damping	0.1440	0.1558	7.6%

2.3.3. Glide Performance

In the next manoeuvre, the pilot rapidly reduced the throttle to idle, which causes the aircraft to enter a glide, with some phugoid oscillations. The comparison of the sink rate of the simulation and log data of this glide gives an indication of the lift to drag ratio; a higher ratio means a lower sink rate at a given airspeed. The sink rates calculated from the data shown in Figure 2.20 is around 1.9 m/s for the simulation and 1.6 m/s for the real flight. However, it should be noted, that there is also a discrepancy in the airspeed: in the log data, the aircraft seems to have been trimmed for a slightly lower airspeed, which would explain at least part of the difference in sink rate.

2.3.4. Climb Performance

The last manoeuvre consists of a phase of full throttle input after a trimmed flight condition. This gives a qualitative indication of the accuracy of the propulsion model. In Figure 2.21, the reactions of simulated and real aircraft to this manoeuvre are plotted. The peak climb rates are very similar, around 11.1 m/s for the simulation and 11.7 m/s for the real flight.

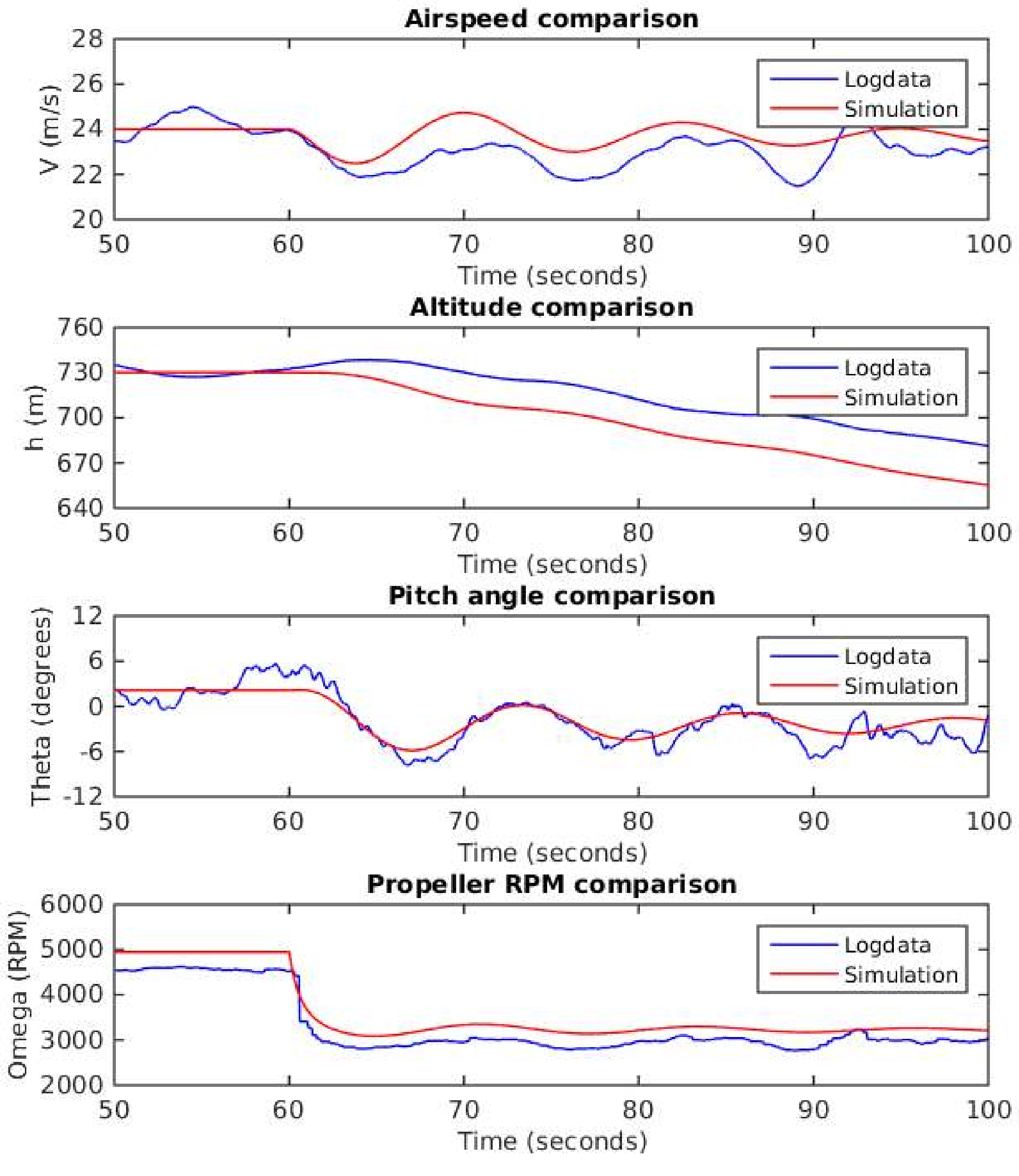

Figure 2.20.: Comparison of simulation and log data of the glide performance. At $t = 60\,\text{s}$, the throttle is reduced to zero, and the aircraft enters a glide, combined with a phugoid oscillation.

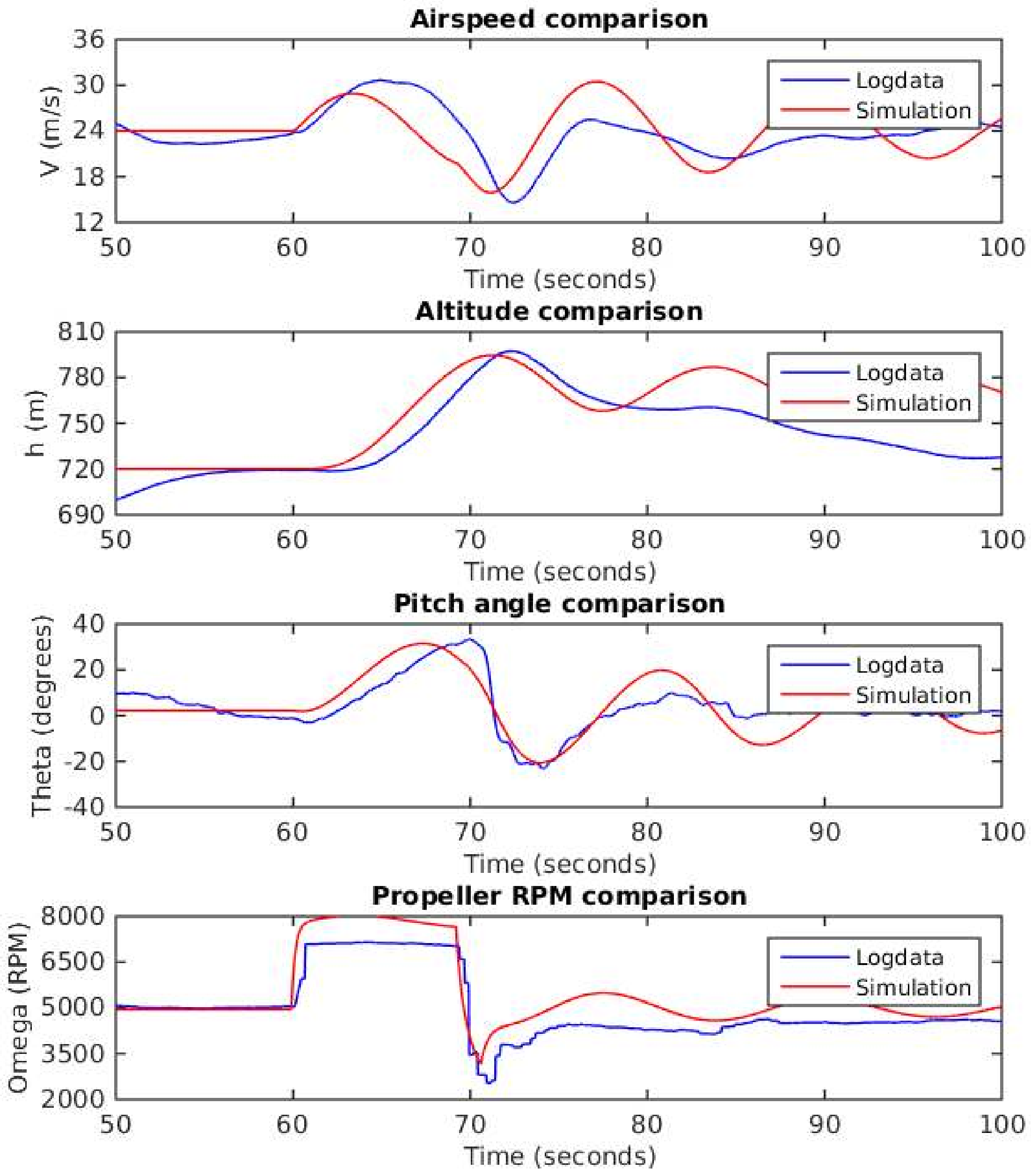

Figure 2.21.: Comparison of simulation and log data of the climb performance. At $t = 60\,\mathrm{s}$, the pilot applies full throttle, and around $t = 70\,\mathrm{s}$ idle and then again the trim setting. Again, a phugoid oscillation is visible.

3. Controller Design and Tuning

Two flight control systems were designed and tested: a classical controller consisting of three separate SISO loops for airspeed, altitude and directional control, and an advanced controller with integrated speed and altitude control based on energy principles. The former was built on an existing concept with some adjustments, and serves as a 'reference' control system to judge the performance of the new controller.

The design of a new control system was necessary due to the requirements of the intended application: landing on a mobile ground vehicle. The close cooperation of two vehicles requires that both vertical speed and forward speed of the UAV can be controlled precisely and independently, minimizing the inherent coupling in the aircraft's longitudinal motion.

The first section of this chapter gives an overview of the SISO controller, while the second section explains the theory as well as the implementation of the second, energy-based control design.

3.1. Classical SISO Control System

The control system described in this section has been used at DLR's Flying Robots Group for a variety of fixed-wing aircraft. It consists of three separate SISO controllers, one for directional control, another for altitude control, and the third for speed control. They are generally designed as cascaded SISO controllers.

This control system has been updated and adapted to the Penguin UAV.

3.1.1. Lateral Control

As can be seen in Figure 3.1, the lateral controller consists of three cascaded SISO loops. It controls the course χ using the ailerons. The course is measured as an angle relative to north, typically given in the interval $[0°, 360°]$. The discontinuity at $0° \,/\, 360°$ needs to be considered in the calculation of the course error, otherwise the aircraft may not turn into the desired direction. For example, if the desired course is 355°, and the actual course is 5°, the aircraft should make a left turn of $-10°$, not a right turn of 350°. Therefore, the course error is calculated according

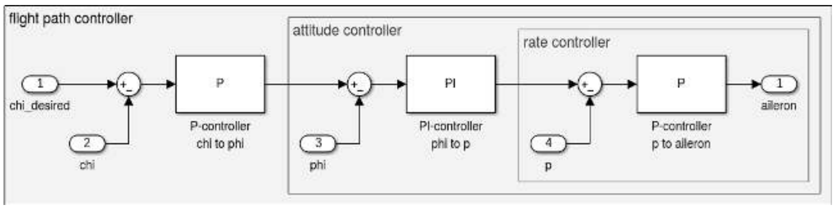

Figure 3.1.: Simplified diagram of the lateral/directional controller. It consists of three cascaded loops controlling course χ, bank angle ϕ and roll rate p, respectively, using P and PI controllers.

to the following equation, which ensures that it is always in the interval $[-\pi, \pi]$:

$$\chi_{err} = [(\chi_{des} - \chi + \pi) \bmod 2\pi] - \pi \tag{3.1}$$

This course error is then fed to a proportional controller, whose output is the desired course change rate $\dot{\chi}_{des}$. However, as the next loop controls the attitude (bank angle), it expects a desired bank angle ϕ_{des}, which is calculated algebraically from $\dot{\chi}_{des}$ based on the force equilibrium in a steady turn (see Figure 3.2).

$$\dot{\chi}_{des} = K_{\chi} \cdot \chi_{err} \tag{3.2}$$

$$\phi_{des} = \arctan\left(\frac{V_k \dot{\chi}_{des}}{g}\right) \tag{3.3}$$

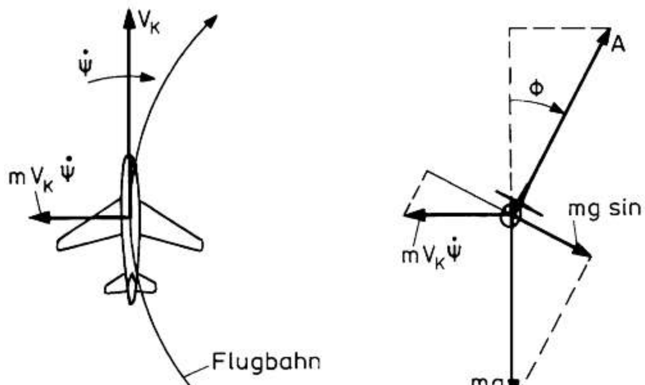

Figure 3.2.: The force equilibrium in a steady turn: the lift force A is tilted by the bank angle ϕ. Its vertical component $A\cos(\phi)$ is in equilibrium with the weight force mg, and the horizontal component $A\sin(\phi)$ is in equilibrium with the centrifugal force $mV_k\dot{\psi}$. With $\dot{\psi} \approx \dot{\chi}$ (equal if there is no wind) it follows that $\tan(\phi) = \frac{1}{g}V_k\dot{\chi}$. (Figure reprinted from [10])

The bank angle is controlled by a PI-controller, that is described by the equation below. ϕ_{des} is limited by a saturation block, with limits that can be adjusted for different phases of the flight, normally limited to 30° bank angle. The controller includes an anti-windup protection that turns the integrator action off ($K_{i\phi} = 0$) when the aileron deflection reaches a limit (±20°).

$$p_{des} = \left(K_\phi + \frac{1}{s}K_{i\phi}\right) \cdot (\phi_{des} - \phi) \tag{3.4}$$

Lastly, the roll rate p is controlled in the innermost loop, using a proportional controller. This part essentially acts as a damping term for the attitude (ϕ) controller.

$$\delta_a = K_p\,(p_{des} - p) \tag{3.5}$$

3.1.2. Altitude Control

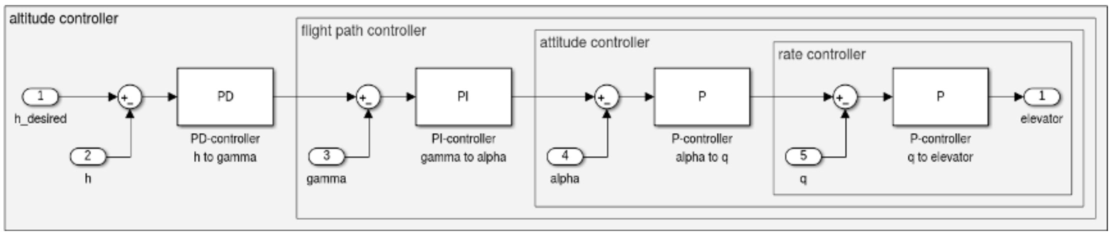

Figure 3.3.: Simplified diagram of the altitude controller. It consists of four cascaded loops controlling altitude h, flight path angle γ, angle of attack α and pitch rate q, respectively, using P, PI and PD controllers.

The altitude control is similar in structure to the lateral control, but it includes an additional top-level control loop, so it is a cascade of four SISO controllers. Figure 3.3 shows a simplified overview of the controller.

The top-level loop is the altitude controller: it produces a vertical speed command from the altitude error. Initially, a proportional controller was used, but after encountering oscillations in the simulation, a derivative term was added, using $V_k \sin(\gamma)$ instead of the numerical derivative of the altitude measurement.

$$\dot{h}_{des} = K_h \cdot (h_{des} - h) - K_{dh} \cdot V_k \sin(\gamma) \tag{3.6}$$

The desired vertical speed is then converted to a desired flight path angle, which is the input to the next loop, a proportional-integral controller. This γ_{des} is limited to account for the limits of the aircraft's performance. The integral part is added at this point in order to ensure that the flight path angle can be controlled without a steady state error. It includes an anti-windup protection that turns off the integral action whenever the elevator command reaches saturation.

$$\gamma_{des} = \arcsin\left(\frac{\dot{h}_{des}}{V_k}\right) \tag{3.7}$$

$$\alpha_{des} = \left(K_\gamma + \frac{1}{s}K_{i\gamma}\right) \cdot (\gamma_{des} - \gamma) \tag{3.8}$$

The last two loops are proportional controllers for angle of attack and pitching rate. However, as the angle of attack can not be measured with the sensors currently available on the Penguin UAV, it is approximated as $\alpha \approx \theta - \gamma$, which is exact in the case of straight flight in zero wind.

Furthermore, it has to be considered that during a steady turn, a pitching rate of $q_{turn} = \sin(\phi)\dot{\psi}$ is measured that comes from the heading rate and is not related to a change in θ or α. Therefore, it has to be added to the q_{des} value, otherwise the rate controller would give an undesired nose-down command during the turn. As seen in Figure 3.2, $\dot{\psi}$ can be written as $\frac{g\tan(\phi)}{V_k}$, therefore we get:

$$q_{des} = K_\alpha\left(\alpha_{des} - (\theta - \gamma)\right) + \sin(\phi)\frac{g\tan(\phi)}{V_k} \tag{3.9}$$

$$\delta_e = K_q\left(q_{des} - q\right) \tag{3.10}$$

3.1.3. Speed Control

The airspeed control consists of a single PI-loop with a feedforward term. The feedforward part is based on the throttle setting for trimmed straight and level flight at a constant airspeed. This value was determined in the simulation for a few different speeds, and linear interpolation is used in between. Anti-windup protection is included to turn off the integrating action when the throttle value reaches a limit (0 or 1).

$$\delta_t = \delta_{t_{FF}}\left(V_{des}\right) + \left(K_{V_a} + \frac{1}{s}K_{iV_a}\right) \cdot (V_{des} - V_a) \tag{3.11}$$

3.1.4. Limitations of the SISO Control System

The main limitations of a SISO control system for fixed-wing aircraft are the uncoordinated control of altitude and airspeed, and the comparatively poor reference tracking of the speed controller. The first issue comes from the inherent coupling of speed and altitude: any change of airspeed will influence the lift force since $F_L \sim V^2$, and any change of flight path angle γ will influence the airspeed due to the longitudinal component of the gravity acceleration ($g\sin(\gamma)$). The two separate controllers then act just like two pilots who don't know each other's intentions. For example, when a climb is commanded, the altitude controller will give a nose-up elevator command, while the speed controller waits until a speed decrease is detected, and only then commands an increased throttle.

The second issue is due to the motor being relatively slow, it is not useful for quick reactions, e.g. to gusts. Furthermore, the motor's control authority is rather limited, whereas the elevator has enough authority to change the flight path angle to values where the SISO speed controller is unable to maintain the desired airspeed. This leads to the additional challenge that a over- or underspeed (stall) protection can not be implemented in an effective way.

3.2. TECS-based Coupled Controller

The Total Energy Control System (TECS) is a way to address the shortcomings of SISO flight controllers. It was first presented by A. A. Lambregts in 1983 [11]. TECS was selected for its ability to decouple altitude and speed responses, while not requiring a very precise model of the aircraft. This decoupling is important for the landing manoeuvre, because the close proximity of aircraft and ground vehicle in the last phase of the landing requires the ability to control the longitudinal position with no or minimal influence on the altitude and vice versa.

This section will first introduce the principles behind total energy control, and then the implementation of TECS for the Penguin UAV.

3.2.1. TECS Core Algorithm

As mentioned in the previous section, the aircraft's responses in altitude and speed are coupled. An increase in thrust will generally increase both the airspeed and the altitude, while a nose-up elevator command will result in increasing altitude and decreasing speed. The control responses for elevator and thrust inputs are roughly orthogonal, as illustrated in Figure 3.4. Conventional SISO controllers don't account for this, and just assign one actuator to each control variable; typically using elevator for altitude control and thrust for speed control.

The TECS principle is based on the notion that an elevator deflection will not have a significant influence on the total (kinetic + potential) energy of the airplane, while the propulsion system quite obviously influences the total energy, with the power added being thrust times velocity, or $P_{thrust} = T \cdot V$. Note that at the same time, there is a energy loss from the drag force amounting to $P_{drag} = -D \cdot V$, so the total amount of energy change per time is $P = (T - D) \cdot V$. Lift and side force

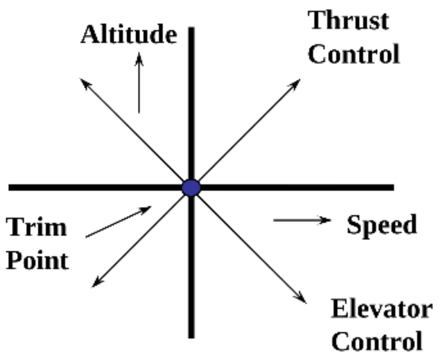

Figure 3.4.: Illustration of the coupling of control responses for elevator and thrust inputs. (Figure reprinted from [12])

are by definition perpendicular to the velocity vector, so they do not contribute to the energy, and the rotational kinetic energy is neglected.

These considerations make it a logical choice to use the engine to control the energy state of the aircraft. The total energy of the aircraft can be expressed as

$$E = \frac{1}{2}mV^2 + mg(h - h_0) \tag{3.12}$$

which makes the energy rate

$$\dot{E} = mV\dot{V} + mg\dot{h} \tag{3.13}$$

Since the energy rate is equal to the power applied, we get

$$mV\dot{V} + mg\dot{h} = (T - D) \cdot V \tag{3.14}$$

We divide by mgV to get the dimensionless equation

$$\frac{\dot{V}}{g} + \frac{\dot{h}}{V} = \frac{T - D}{mg} \tag{3.15}$$

With $\frac{\dot{h}}{V} = \sin(\gamma)$ and assuming small angles γ, we get

$$\frac{\dot{V}}{g} + \gamma = \frac{T - D}{mg} \tag{3.16}$$

The drag variation is generally slow, and assuming that it is compensated by the trim setting of the thrust, we can conclude that the short-term thrust requirement is proportional to the dimensionless energy rate:

$$\frac{\delta T}{mg} = \frac{\dot{V}}{g} + \gamma \tag{3.17}$$

Therefore, a TECS controller uses thrust to control the error in the dimensionless energy rate $\frac{\dot{V}}{g} + \gamma$.

The distribution of the energy rate between $\frac{\dot{V}}{g}$ and γ still has to be controlled by another actuator, and the elevator is ideally suited for that, since it does not influence the total energy rate. The elevator mainly influences the angle of attack, which in turn influences acceleration and γ in opposite directions, therefore in general $\frac{\dot{V}}{g} - \gamma$ is used as the control variable.

In practice, the TECS control algorithm is typically implemented as follows:

$$\frac{T_{des}}{mg} = \frac{K_{TI}}{s}\left(\gamma_{des} - \gamma + \frac{\dot{V}_{des}}{g} - \frac{\dot{V}}{g}\right) - K_{TP}\left(\gamma + \frac{\dot{V}}{g}\right) \tag{3.18}$$

$$\theta_{des} = \frac{K_{EI}}{s}\left(\gamma_{des} - \gamma - \frac{\dot{V}_{des}}{g} + \frac{\dot{V}}{g}\right) - K_{EP}\left(\gamma - \frac{\dot{V}}{g}\right) \tag{3.19}$$

Two things should be noted here: the proportional parts are implemented without reference to the desired values. This smoothens the transient response as step inputs are only processed through the integrals, and it eliminates response overshoots. Additionally, θ is used instead of α in the elevator control path, as it is easier to measure and has the same short term dynamics, because of $\theta \approx \alpha + \gamma$ and the slow dynamics of γ.

The Simulink implementation of the TECS core algorithm, shown in Figure 3.5, additionally includes anti-windup protection and a switching logic for limit thrust operations. The purpose of the latter is to define which variable (normally $\dot{V}$) shall be controlled by the elevator when the thrust is saturated and the control system is thus unable to control both variables. This ensures that, if an excessive γ is commanded, the speed command will be prioritized and dangerous over- or underspeed (stall) situations can be avoided. The opposite is also possible, for example in the last phase of landing (flare), where the flight path γ is typically prioritized over $\dot{V}$.

Figure 3.6 shows the complete TECS-based controller, with the TECS core block from Figure 3.5 in the middle. The other parts of the controller will be explained in the following sections.

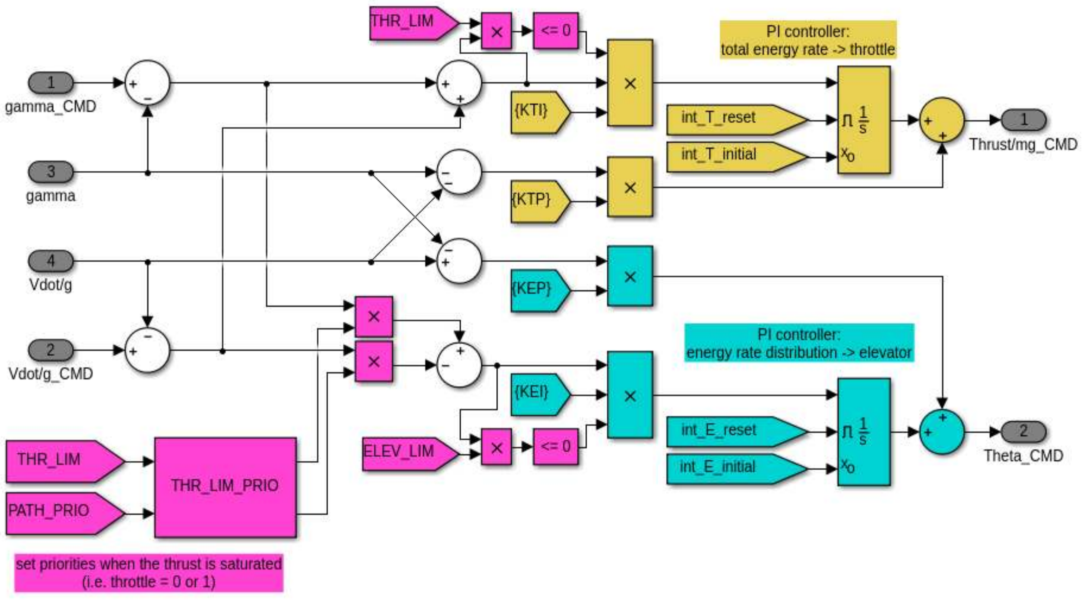

Figure 3.5.: The TECS core algorithm as implemented in Simulink. Note the *THR_LIM_PRIO* block, which decides which variable ($\dot{V}$ or γ) shall be controlled by the elevator when the thrust reaches saturation.

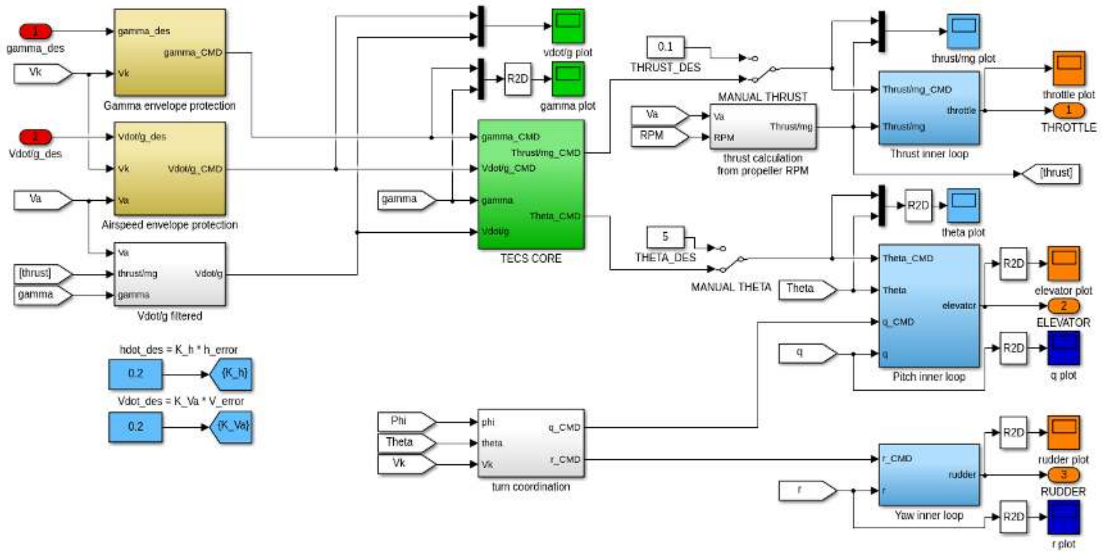

Figure 3.6.: The complete TECS controller, including envelope protection blocks, the TECS core, inner loop controllers as well as turn coordination and a yaw damper. All parts of the controller are explained in detail in the different parts of Section 3.2.

3.2.2. Inner Loop Controllers

As mentioned in the previous section, the output of the TECS core consists of a pitch angle θ_{des} and a non-dimensional thrust $\frac{T_{des}}{mg}$. Inner loop SISO controllers then process these commands into elevator and throttle settings.

The inner loop control for pitch can generally be implemented very similar to the inner loops of the altitude controller in Section 3.1.2. However, it was decided to have the PI controller for θ and the pitch rate damper in parallel. This makes it very similar to a PID controller, since $q \approx \dot{\theta}$ for small bank angles ϕ. The resulting controller is described by the following equation:

$$\delta_e = -\left(K_\theta + \frac{K_{i\theta}}{s}\right)(\theta_{des} - \theta) - K_q\left(q_{turn} - q\right) \tag{3.20}$$

Like in the SISO altitude controller described earlier, anti-windup protection is included, as well as a q_{turn} command for coordinated turns. The q_{turn} was implemented slightly differently, now including a $\cos(\theta)$ that was neglected before since θ is typically small.

$$q_{turn} = \sin(\phi)\cos(\theta)\tan(\phi)\frac{g}{V_k} \tag{3.21}$$

Also new is a gain scheduling function, that scales the control output with the inverse of the dynamic pressure, mostly intended for flight in high altitudes, where the air density is low. However, this feature has not been tested. The complete inner loop pitch controller is shown in Figure 3.7.

The inner loop controller for thrust is completely new, as the SISO airspeed controller consisted of a single loop directly controlling airspeed with throttle. Here, we need to control thrust, and therefore we need a measurement or estimation of thrust. A direct thrust measurement would be rather difficult to implement, but a the rotational speed of the electric motor, and thus the propeller, is easily measured. In fact it had been measured before, but it was not used for control purposes. This speed and the airspeed measurement are used together with the propeller model discussed in Section 2.1.2 to get an estimate of the current thrust.

The thrust controller is then implemented as a PI controller with a feedforward term. For the feedforward command, the desired thrust is divided by the estimated available maximum thrust, which gives a rough estimation of the required throttle setting. The controller can be described by the equation below. Its Simulink implementation is shown in Figure 3.8.

$$\delta_t = \left(K_T + \frac{K_{iT}}{s}\right)(T_{des} - T) + \delta_{tFF} \tag{3.22}$$

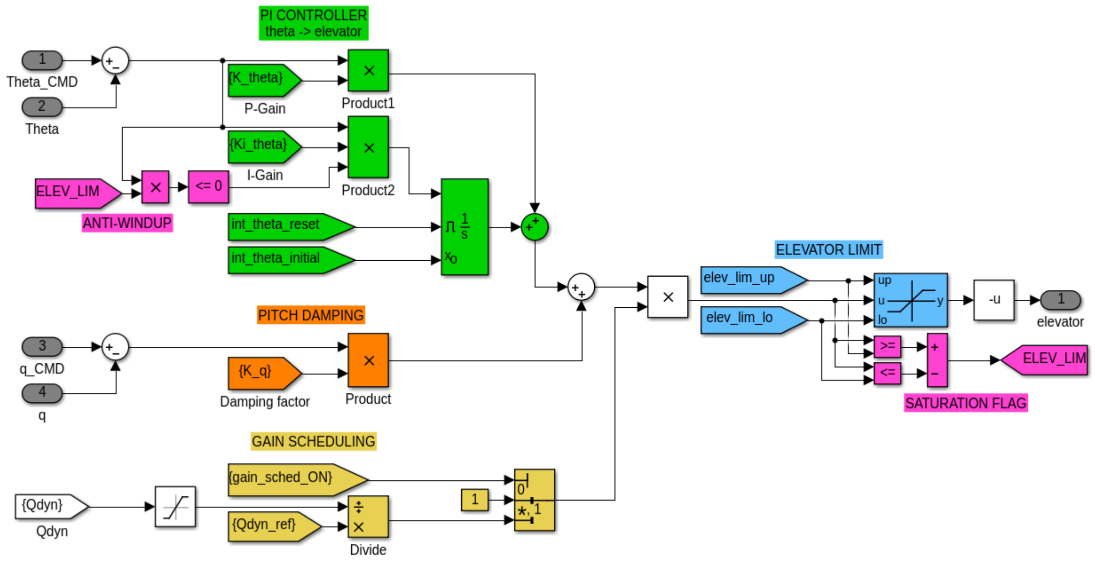

Figure 3.7.: The inner loop pitch controller. In green is the main PI controller for θ, in orange the pitch damper and in yellow the gain scheduling. Also included is a saturation block to limit the output and an anti-windup protection for the integrator.

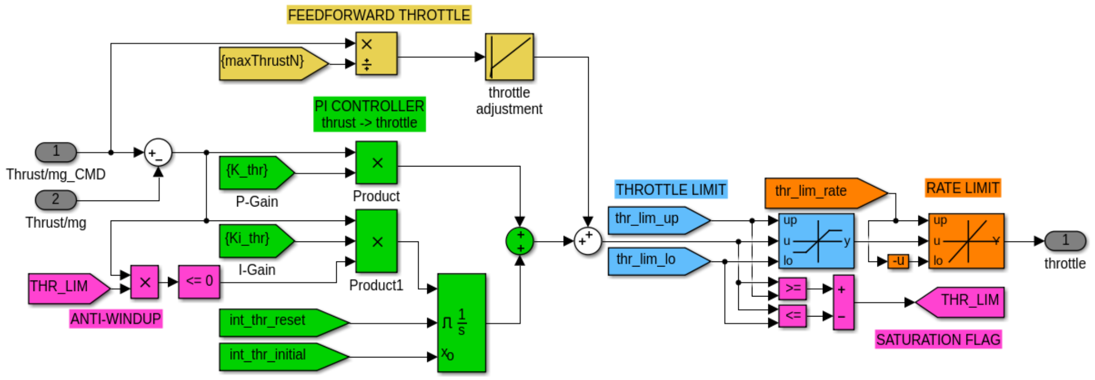

Figure 3.8.: The inner loop thrust controller. It is a PI controller with a feedforward term, anti-windup protection and a rate limiter to avoid excessive control activity.

3.2.3. Outer Loops and Envelope Protection Features

The controller inputs for TECS can either be given directly as γ and $\dot{V}$, or these inputs can be generated from outer control loops, which is typically done as proportional control:

$$\dot{V}_{des} = K_V \left(V_{des} - V_a\right) \tag{3.23}$$

$$\dot{h}_{des} = K_h \left(h_{des} - h\right) \tag{3.24}$$

$$\gamma_{des} = \frac{\dot{h}_{des}}{V_k} \tag{3.25}$$

These commands in γ and $\dot{V}$ should however be limited according to the aircraft performance in order to stay within the safe flight envelope. We have seen in equation 3.16, that

$$\frac{\dot{V}}{g} + \gamma = \frac{T - D}{mg} \tag{3.26}$$

Data from the simulation model is used to estimate the drag as well as the maximum available power as functions of airspeed, and these estimates are used to limit the control inputs as follows:

$$-\frac{D}{mg} < \frac{\dot{V}_{des}}{g}, \gamma_{des} < \frac{T_{max} - D}{mg} \tag{3.27}$$

The two control inputs are limited separately, therefore their sum could still be beyond the performance limit. However, in that case the priority logic inside the TECS core will decide which command shall be followed with higher priority.

The rates of the control inputs are also limited with

$$\frac{n_{min} g}{V_k} < \frac{\ddot{V}_{des}}{g}, \dot{\gamma}_{des} < \frac{n_{max} g}{V_k} \tag{3.28}$$

where $n_{min}g$ and $n_{max}g$ are the allowed limits of vertical acceleration ("g-force"). In a passenger airplane, these limits would have to be low for passenger comfort, e.g. around $\pm 0.1g$, but in an unmanned plane the main limit is the structure. For the Penguin, the limits were set to $\pm 1g$.

Lastly, $\dot{V}$ is limited again for stall and overspeed protection. This limit comes last, because it is critical for flight safety and shall not be constrained by other limits. It is defined as follows:

$$K_V \left(V_{min} - V_a\right) < \dot{V} < K_V \left(V_{max} - V_a\right) \tag{3.29}$$

where V_{min} and V_{max} are the smallest and largest allowed airspeeds. In this case they were set to $V_{min} = V_{stall} + 2\,\mathrm{m/s}$ and $V_{max} = 35\,\mathrm{m/s}$, with the values for V_{stall} depending on aircraft mass and bank angle. The stall speeds were taken from the Penguin's user manual.

3.2.4. Lateral Control

For the lateral control, the previous controller described in Section 3.1.1 was reused. A control system similar to TECS, called Total Heading Control System (THCS) exists, but it was not implemented because it does require a sideslip angle measurement, which is not available in the UAV used for this thesis.

However, a yaw damper as described by the equation below was implemented. It is a proportional control of the yaw rate r, where the desired value corresponds to the yaw rate in a coordinated turn.

$$\delta_r = -K_r\,(r_{turn} - r) \tag{3.30}$$

$$r_{turn} = \cos(\phi)\cos(\theta)\tan(\phi)\frac{g}{V_k} \tag{3.31}$$

This yaw damper would increase the damping of the dutch roll mode as well as enable coordinated turns using both ailerons and rudder. However, due to time constraints it has not been tested in flight. It should be noted that the dutch roll of the Penguin UAV is naturally well damped and a yaw damper is not essential to its safe operation.

3.3. Controller Tuning

The tuning of control parameters for small UAV is often done manually in flight, as in many cases, a mathematical model of the UAV is not available, so there is no other option. If a model is available, analytical methods than can be employed. For tuning of the inner loops, pole placement is a commonly used method. A tuning method for TECS is presented in [13].

However, in the interest of a rapid progression towards flight testing, the practical approach of manual tuning was chosen. The control parameters were first tuned by hand in the simulation, and subsequently re-tuned as necessary during flight tests, employing the Ziegler-Nichols tuning method. This led to a sufficient control performance with limited effort, while an elaborate optimization of control gains would not necessarily have much better results, given the expected inaccuracies of the model due to the limitations of the methods used in the creation of the flight dynamics model.

4. Controller Evaluation

Both TECS and the SISO control system were tested with a range of manoeuvres both in simulation and in flight. The goal was to compare the performance of both controllers and qualitatively assess the impact of using an energy-based MIMO controller.

In the simulation environment, step changes in altitude and airspeed were tested as well as combined manoeuvres, where both commands are changed simultaneously or shortly after each other. Due to the big effort involved with flight testing and the limited flight time, such combined manoeuvres could unfortunately not be included in the flight testing.

4.1. Simulation

For the simulation tests, both controllers were given the same set of manoeuvres, consisting of airspeed and altitude step commands. All simulations were performed at zero wind and with the flaps in the 12° position, as this is the usual configuration for the Penguin when it is flown manually.

4.1.1. Climb and Descent

In altitude change manoeuvres, a major deficiency of the SISO control becomes apparent: due to the limited control authority of the throttle, compared to the elevator, an altitude command may well drive the airspeed to or beyond the limits of safe flight. In the SISO controller as implemented here, a limit is placed on the commanded flight path angle γ to avoid unsafe situations. However, this limit does not account for airspeed, wind, air density, flaps position or other factors that all influence the maximum γ at which it is still possible to maintain the airspeed. Therefore, this static limit may in some situation place an unduly strict limit on the commanded γ, while in other situations not preventing a dangerous speed deviation. In the TECS controller, the 'speed priority' logic ensures that the speed is always controlled by letting the elevator control only the airspeed whenever the throttle command reaches a limit. This implicitly places a limit on γ whenever needed.

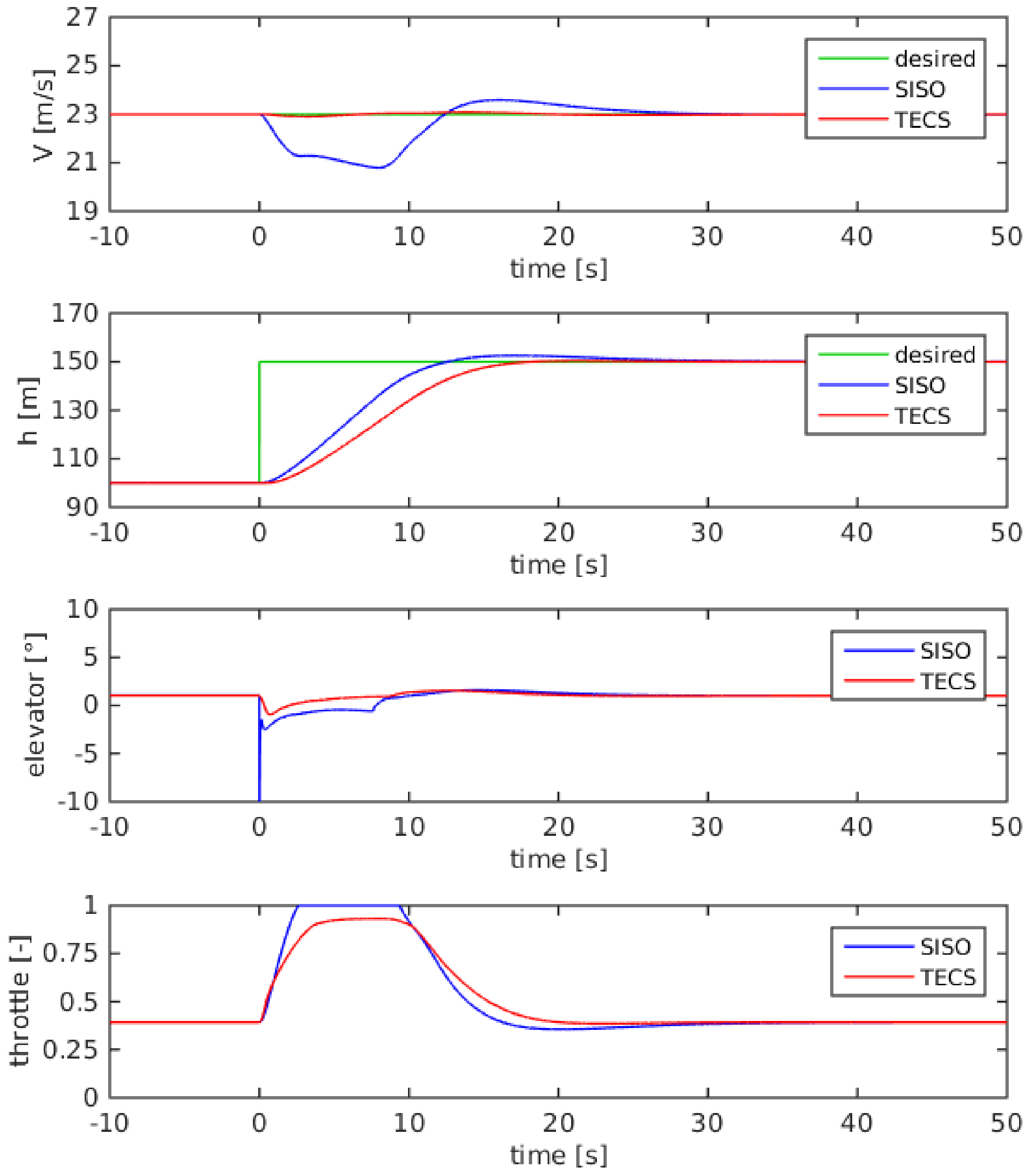

Figure 4.1.: Simulation of a 50 m climb manoeuvre. The SISO controller is not able to maintain the airspeed, despite a full throttle command. The TECS logic however reduces the climb angle as necessary to maintain the airspeed.

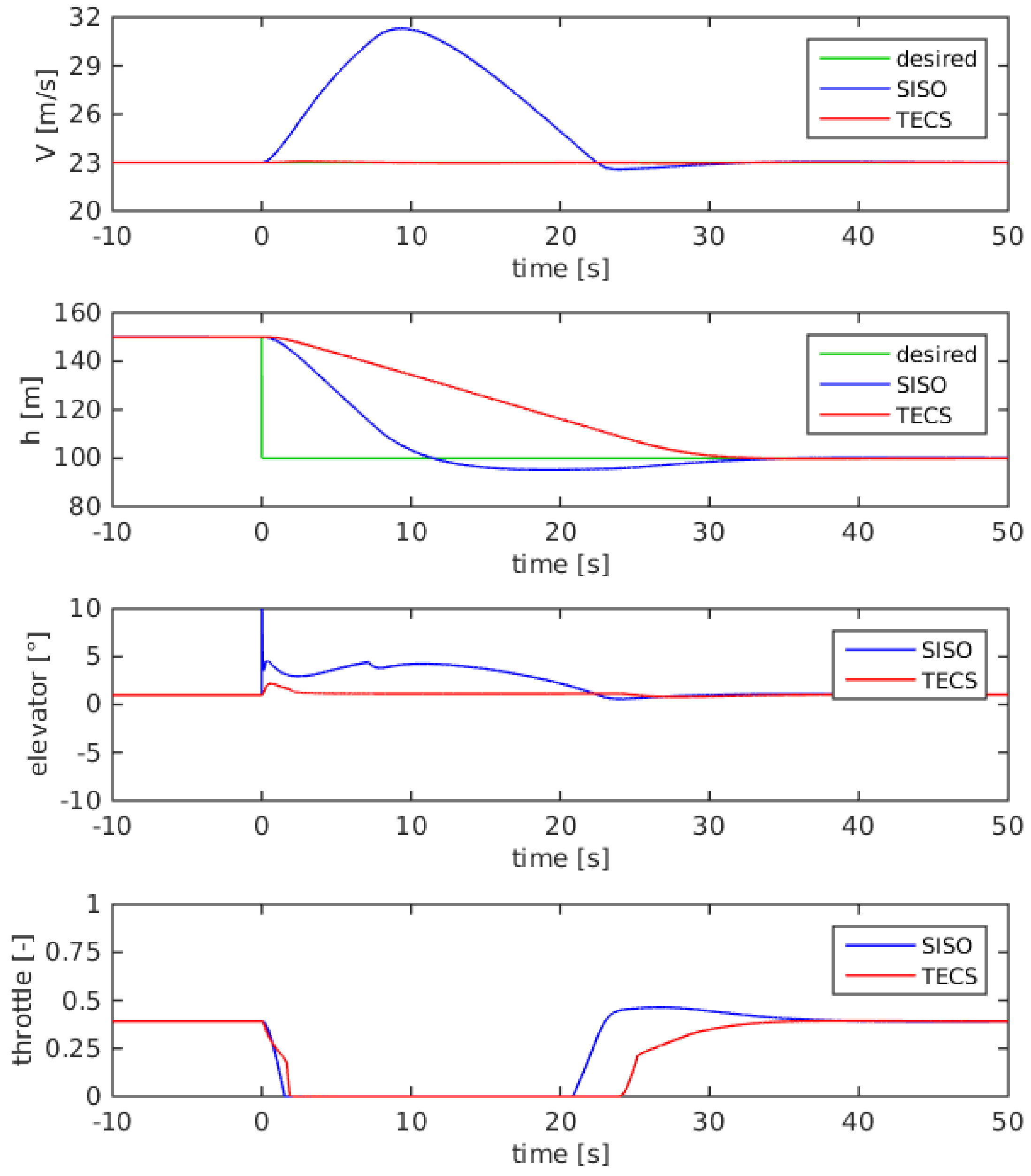

Figure 4.2.: Simulation of a 50 m descent manoeuvre. As before in the climb, the SISO controller is unable to maintain the airspeed. The increased airspeed now also leads to an increased sink rate and a bigger overshoot.

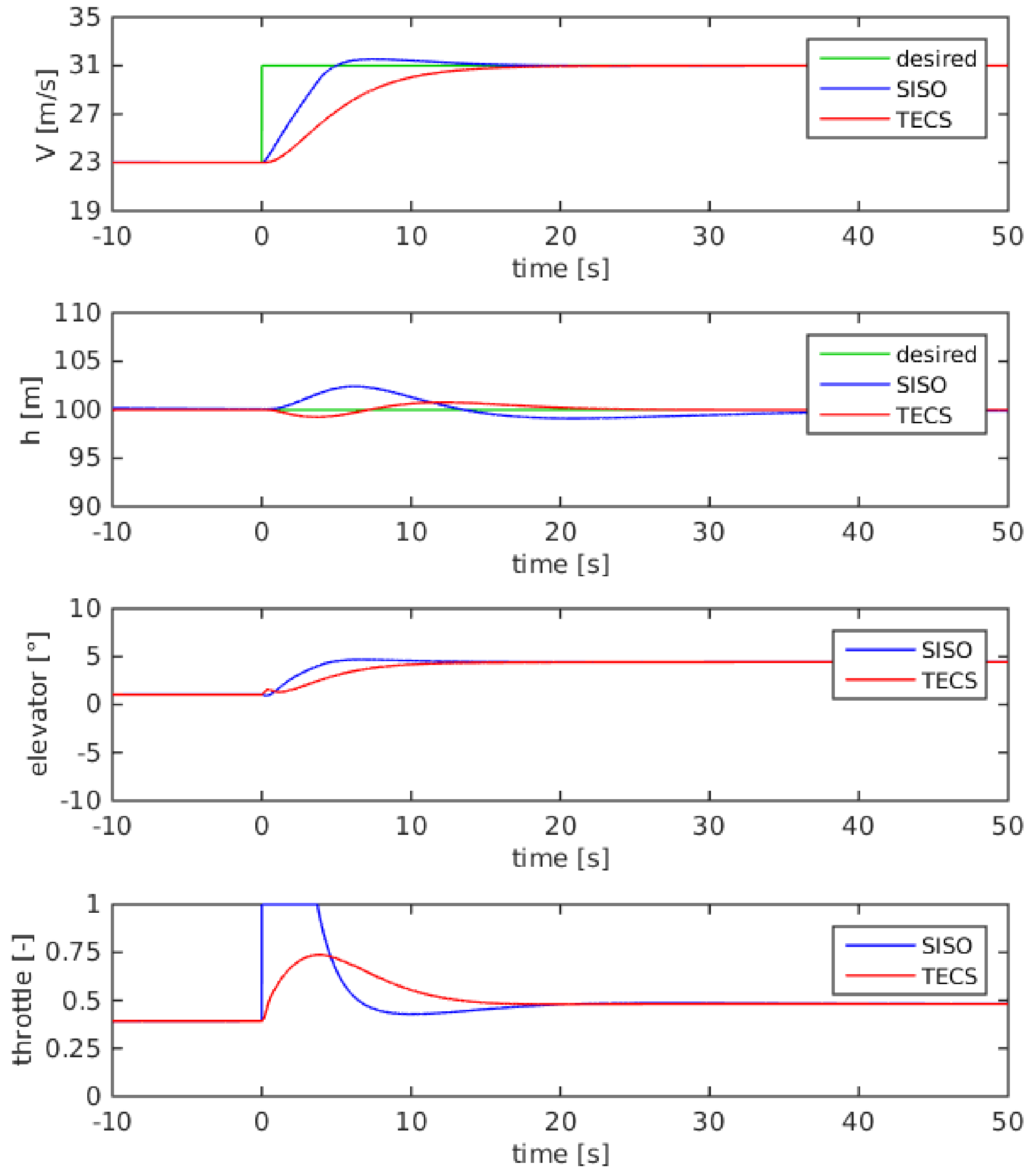

Figure 4.3.: Simulation of a 8 m/s speed increase. Here, the coordination of throttle and elevator in TECS is clearly noticeable. In the SISO case, the altitude error reaches 2.5 m, while the TECS controller keeps it under 1 m.

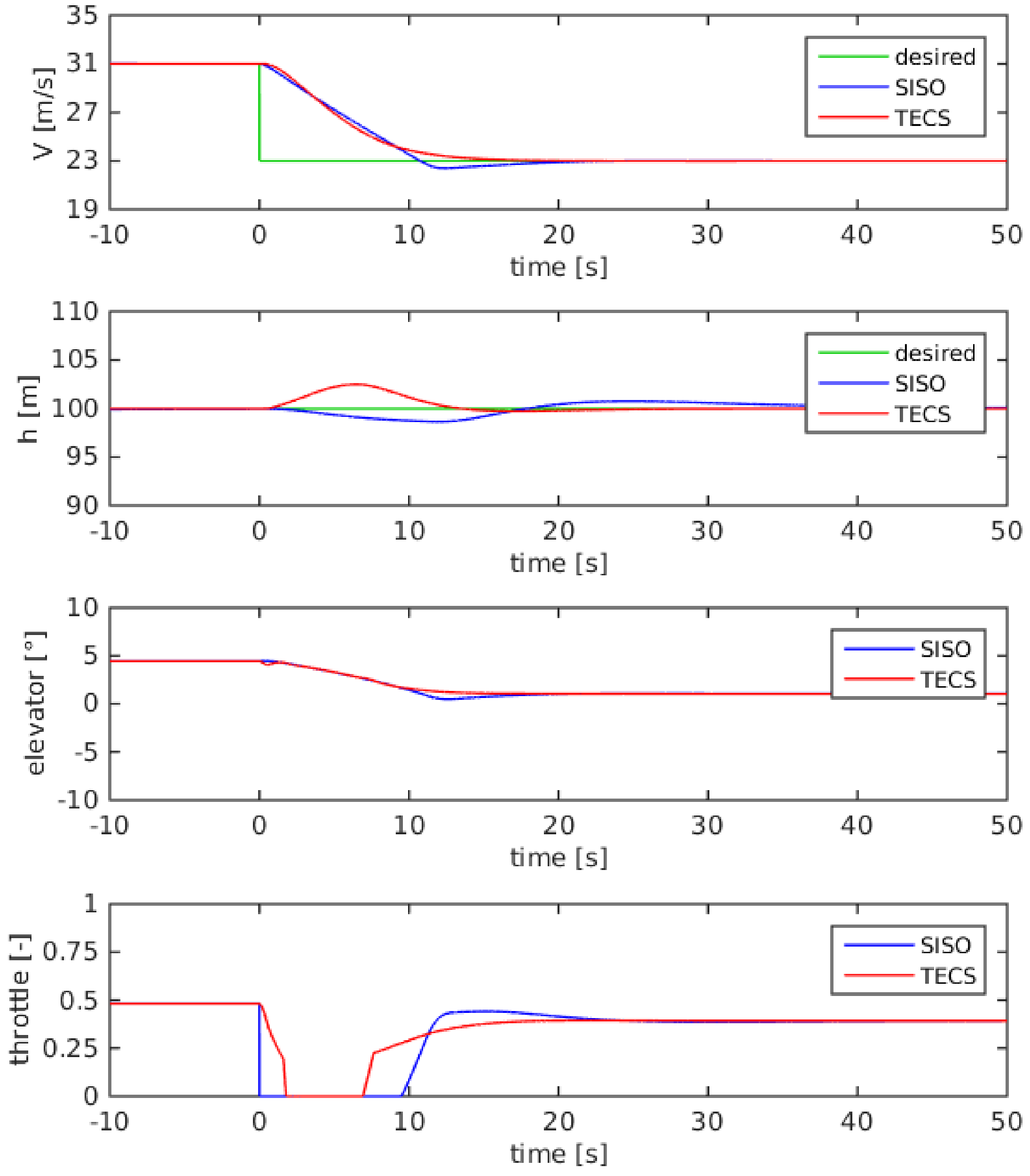

Figure 4.4.: Simulation of a 8 m/s speed decrease. At first sight, it looks like the TECS concept is failing in this case. However, the altitude error of 2.5 m is a result of the 'speed priority' logic at zero throttle.

Figures 4.1 and 4.2 show airspeed and altitude, as well as elevator and throttle commands, during a climb of 50 m and a descent of 50 m. In the SISO case, during the climb the airspeed drops by 2 m/s, which in this case is safe (well above the stall speed), but not desired anyway. The descent looks even worse, with the airspeed continuously rising during the whole descent. Stricter γ limits would of course reduce the airspeed deviation in these manoeuvres, however it would restrict γ unnecessarily much in other situations, e.g. in headwind or in a descent with the flaps fully extended, where larger γ can be achieved.

The TECS controller, on the other hand, manages to keep the airspeed very close to the desired value. The implicit dynamic limiting of γ is clearly active in the descent, where the throttle remains at idle for over 20 seconds, but the airspeed stays at the commanded 23 m/s. In the climb, the throttle command does not quite reach the upper limit, which means that the γ limit of the envelope protection block is slightly too strict.

4.1.2. Speed Changes

By applying steps in the desired airspeed, the coordination of elevator and throttle of the TECS controller becomes apparent. Figure 4.3 shows a speed increase of 8 m/s. The SISO speed controller rather aggressively increases the throttle, and the altitude controller only manages to stop the resulting climb when the altitude deviation reaches 2.5 m. The TECS controller on the other hand keeps the altitude error within 1 m at all times.

The 8 m/s speed decrease in Figure 4.4 looks almost the same in the SISO case, the altitude error is around 1.5 m. The TECS controller seems to perform worse now, but a closer look reveals the reason for this: the throttle reaches zero, and due to the 'speed priority' logic, the TECS controller then uses the elevator to assist the speed change by temporarily climbing a bit. Still, the altitude error is only 2.5 m, and corrected immediately when the desired airspeed is reached.

4.1.3. Combined Manoeuvres

In the manoeuvre shown in Figure 4.5, airspeed and altitude commands are given simultaneously, and their amplitudes are chosen to keep the total energy roughly constant. The airspeed is increased and the altitude is decreased, so it is essentially a transfer of potential to kinetic energy. As expected, the TECS controller only uses minimal control activity in throttle and completes the manoeuvre with excellent coordination. The SISO speed controller gives throttle commands in both directions, and due to the simultaneous change of airspeed and altitude, the overshoots are larger than in the previous manoeuvres.

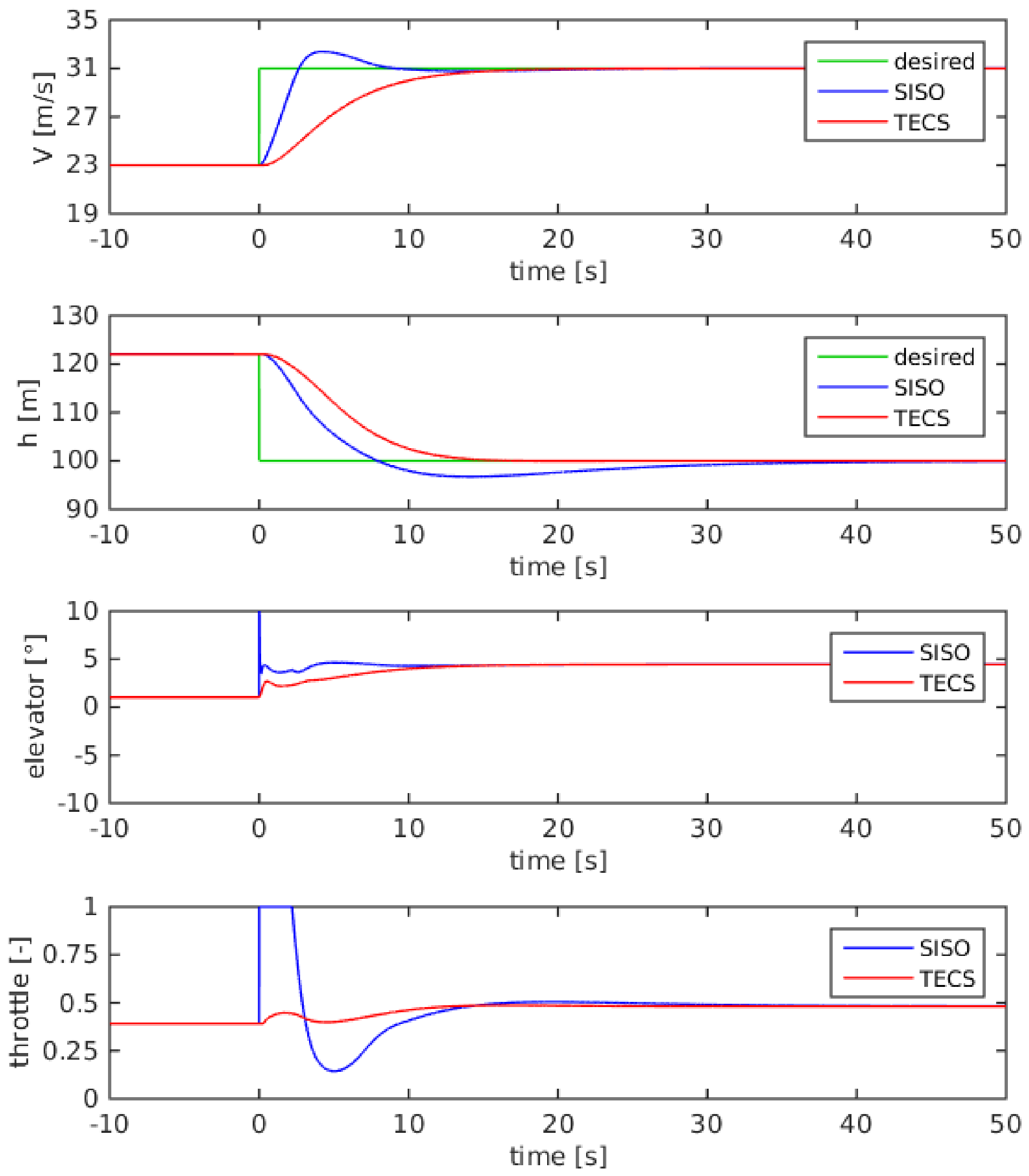

Figure 4.5.: Combined manoeuvres like this transfer of potential to kinetic energy show a big advantage of the TECS concept. If the total energy remains constant, TECS will have only minimal control activity in throttle, unlike the SISO controller, which gives big throttle commands in both directions for this manoeuvre.

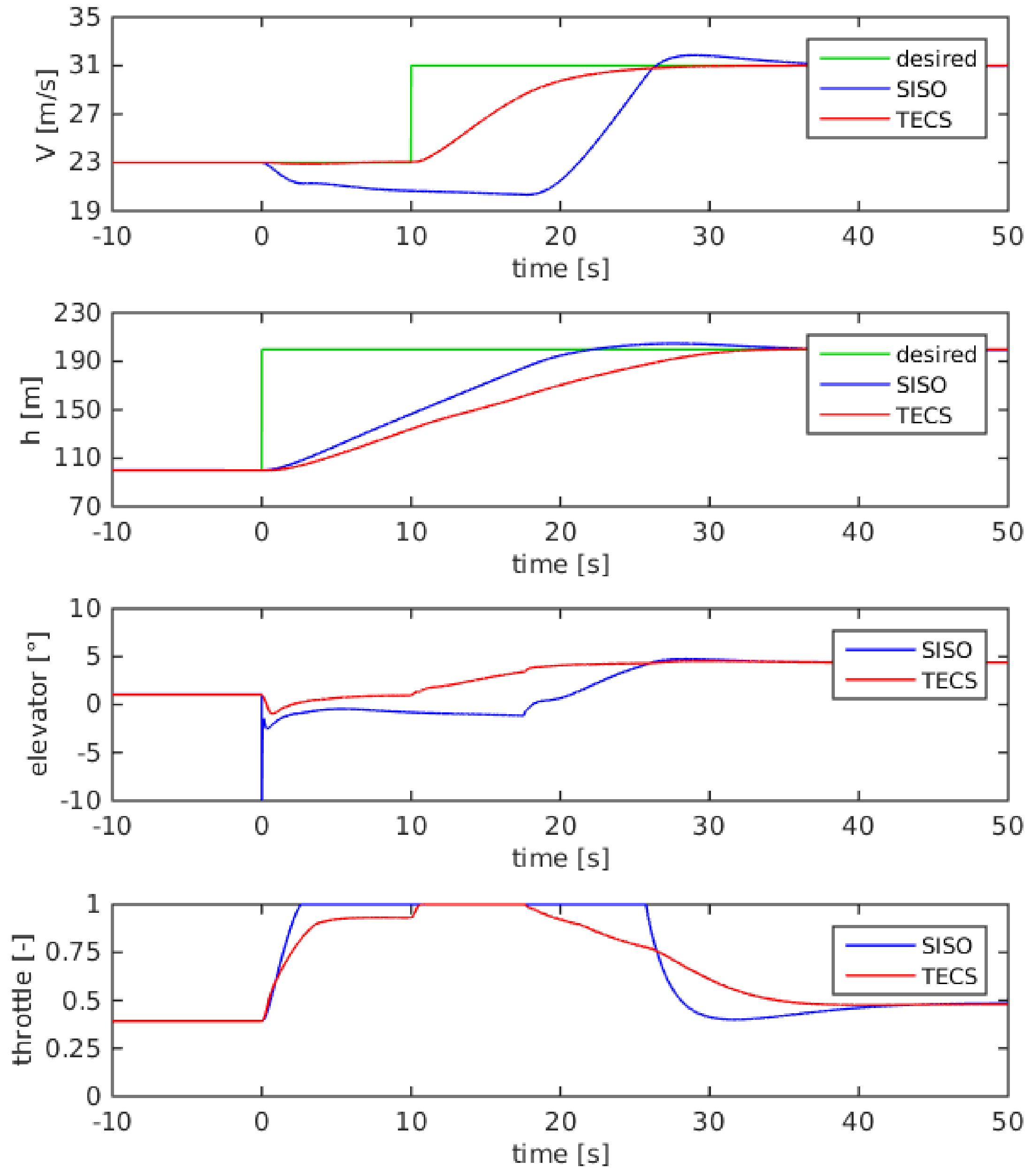

Figure 4.6.: Speed increase during a climb. In the SISO case, the speed command is ignored, because the throttle is already at the limit. TECS, on the other hand, reduces the climb angle slightly to achieve the desired velocity.

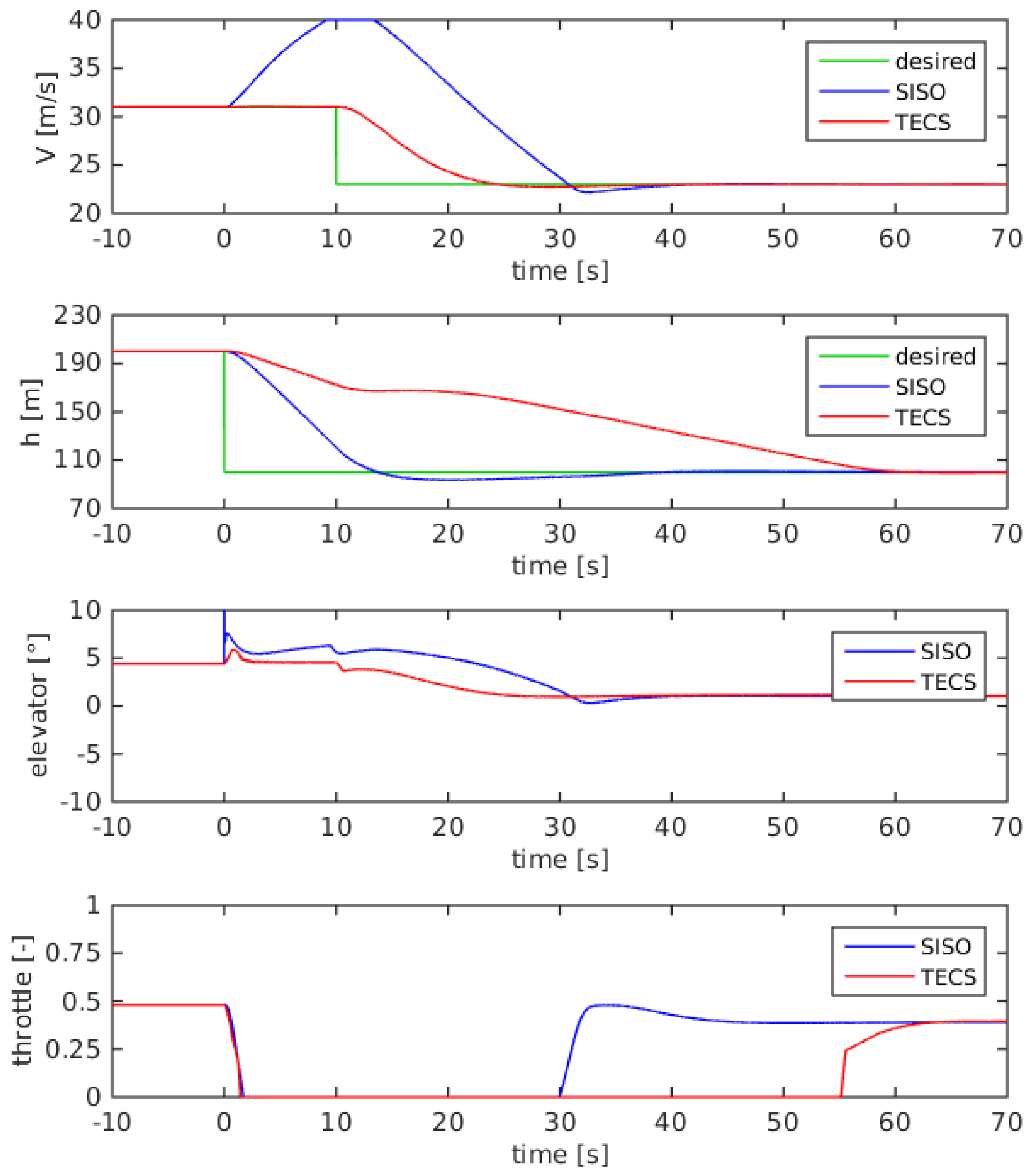

Figure 4.7.: Speed decrease during a descent. Similar to the case before, the SISO controller ignores the speed command until the altitude is reached. TECS goes as far as to level off in order to reach the desired airspeed. Of course the descent will take longer, but the airspeed must be kept within clear boundaries for reasons of flight safety.

The same ‘energy transfer’ manoeuvre was also performed in the opposite direction, i.e. a speed decrease simultaneously with an altitude increase. It had nearly identical results and is therefore omitted here.

Figure 4.6 shows a climb manoeuvre with a step increase of V_{des} during the climb. As seen earlier, the SISO controller is not able to maintain the airspeed during a climb, therefore the speed command is completely ignored until the level off starts. The TECS controller nicely shows its ‘speed priority’ function and reduces the climb angle slightly to reach the desired airspeed.

The opposite case is shown in Figure 4.7 where a speed decrease is commanded during a descent. In this case, the TECS controller goes as far as to level off temporarily to reach the desired airspeed, and afterwards it continues the descent at a shallower angle.

4.2. Flight Testing

Flight tests were performed with both the SISO controller and the TECS controller. The tests took place in two separate flights on the same day, the wind conditions can therefore be assumed to be similar and to impact the control performance of both controllers in the same way. During the flights, after a manually piloted takeoff and the initial controller tuning, the plane was flying a 8-shaped pattern, and altitude and airspeed commands were given on the long, straight parts of the pattern. The data presented here was logged at 100 Hz. Like in the simulations, the flaps were in the 12° position at all times.

4.2.1. Straight and Level Flight

Figures 4.8 and 4.9 show the performance of the SISO controller and the TECS controller, respectively, in horizontal flight. The SISO controller, after having lost some altitude in the preceding turn, overshoots the altitude command when the wings are levelled again. Neither the altitude loss nor the overshoot are seen with the TECS controller, which keeps the altitude error below 1 m at all times. The airspeed tracking is also improved with TECS, although the difference is not as distinct as in altitude.

4.2.2. Climb and Descent

Unfortunately, it was discovered only after the flight, that a rate limit for γ_{des} in the SISO controller was inadvertently set to 2°/s during the entire flight. This had no influence on the horizontal flight in Figure 4.8, as $\dot{\gamma}_{des}$ remained below that limit, however in the climb and descent manoeuvres, it had a big influence.

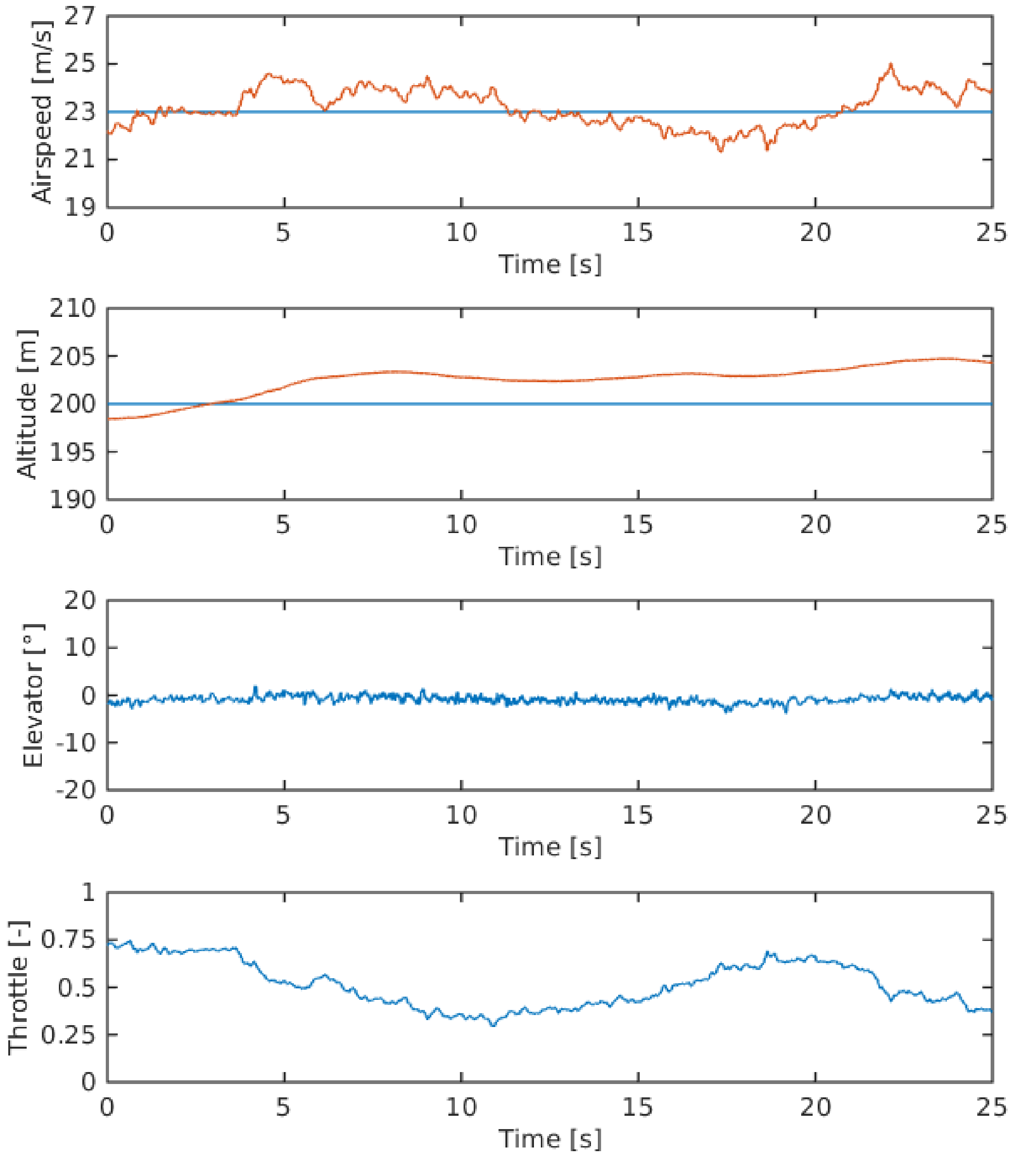

Figure 4.8.: Horizontal flight with the SISO controller. The airspeed tracking is acceptable, the altitude tracking is rather poor.

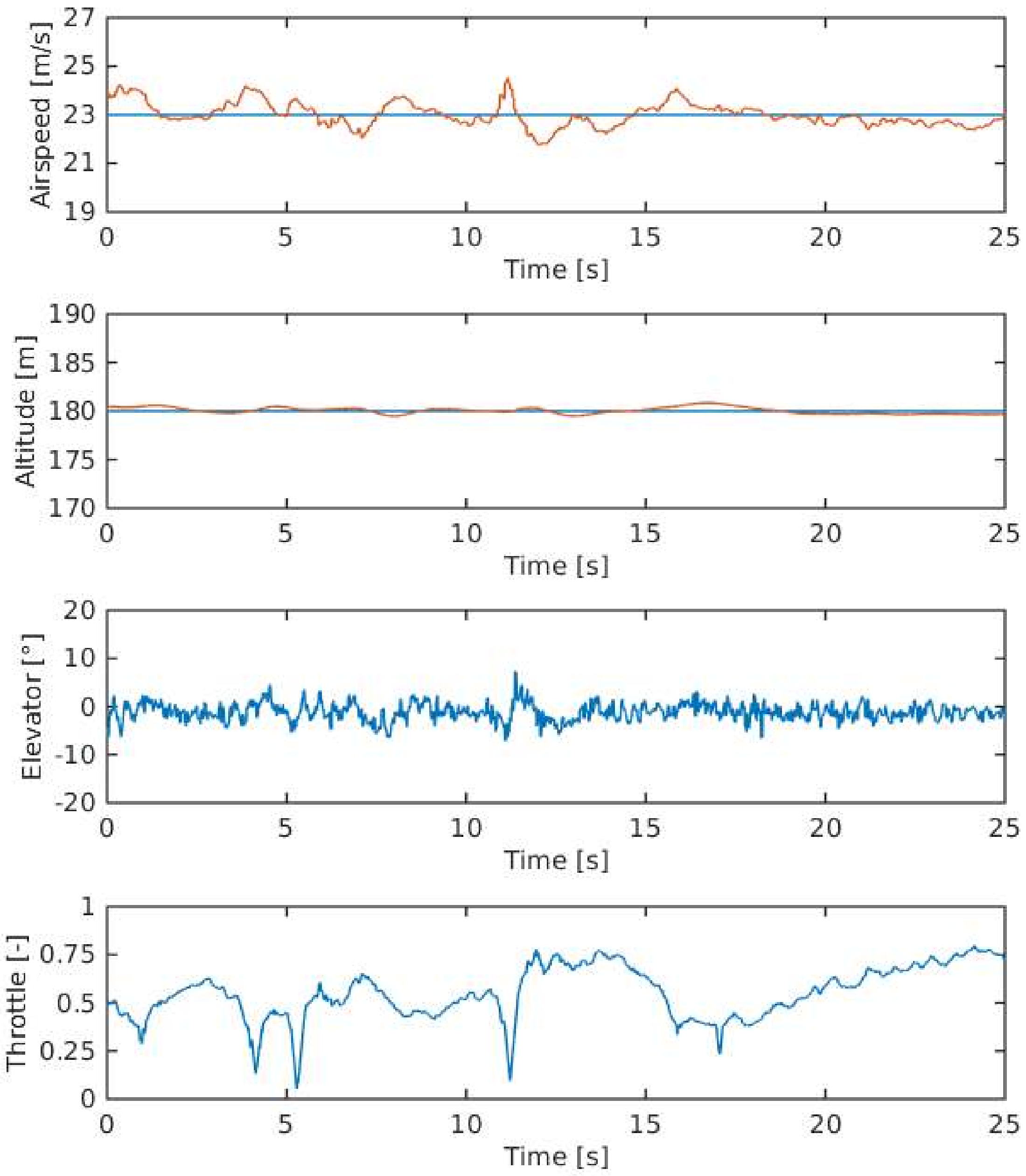

Figure 4.9.: Horizontal flight with the TECS controller. Tracking of airspeed and especially altitude is improved compared to the SISO controller, however this comes at the cost of increased control activity.

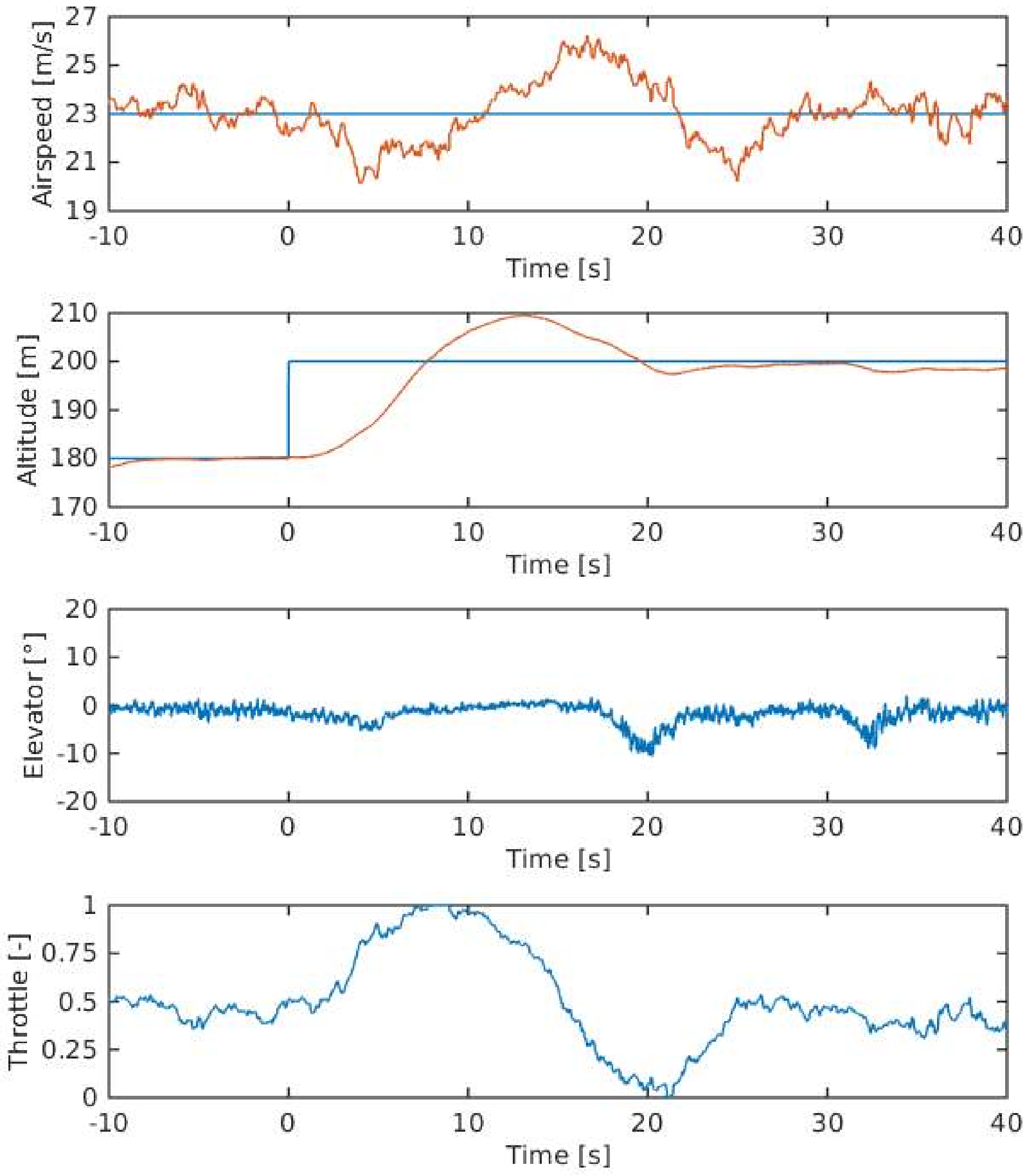

Figure 4.10.: A 20 m climb with the SISO controller. The altitude performance is very poor here, however it was discovered after the flight that a rate limit for γ was mistakenly set too low. Despite this "soft" control in altitude, the airspeed deviates quite significantly (around 3 m/s) from the desired value.

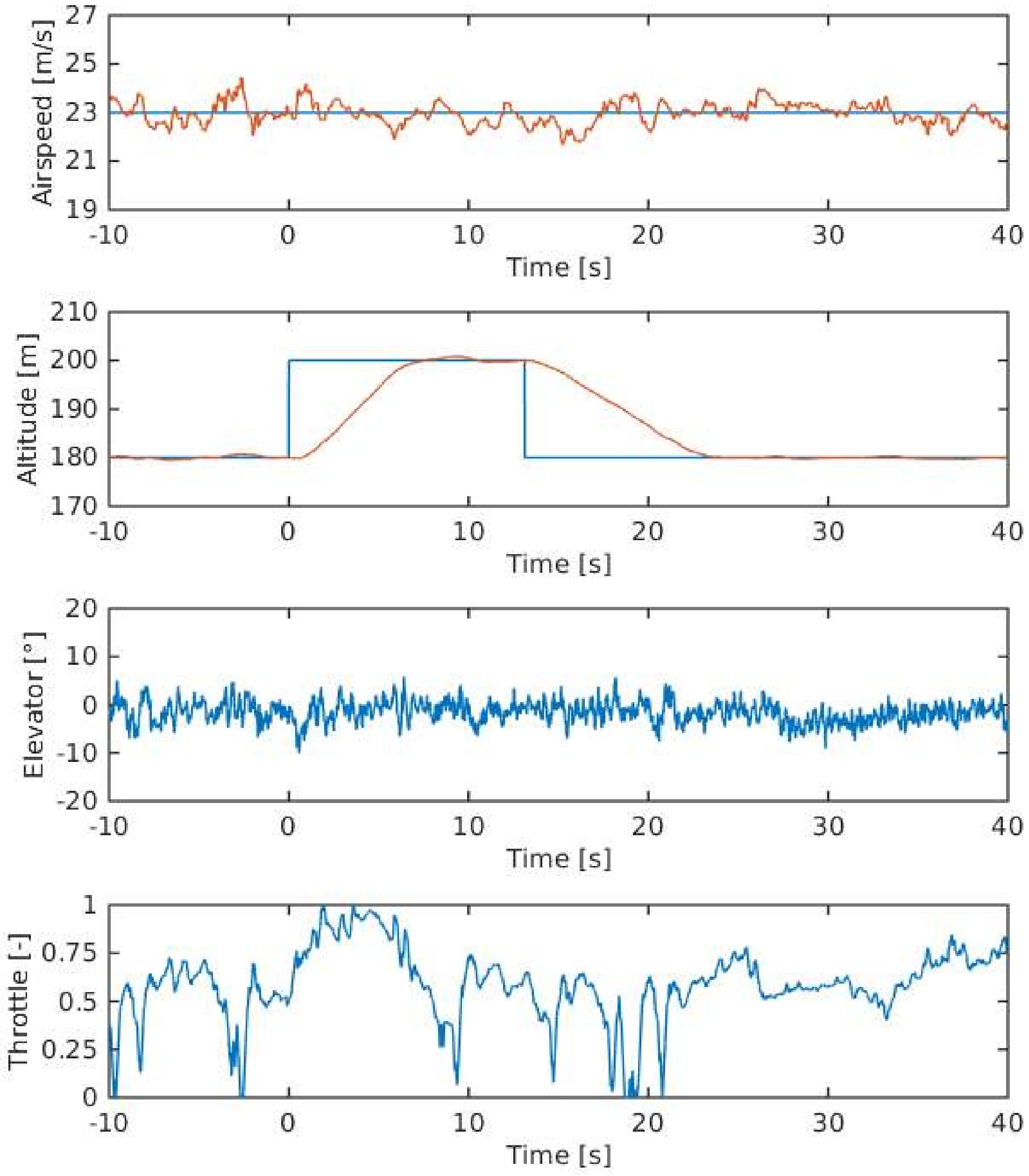

Figure 4.11.: Climb and descent with the TECS controller. The manoeuvres are nicely coordinated, the airspeed tracking is just as well as during level flight.

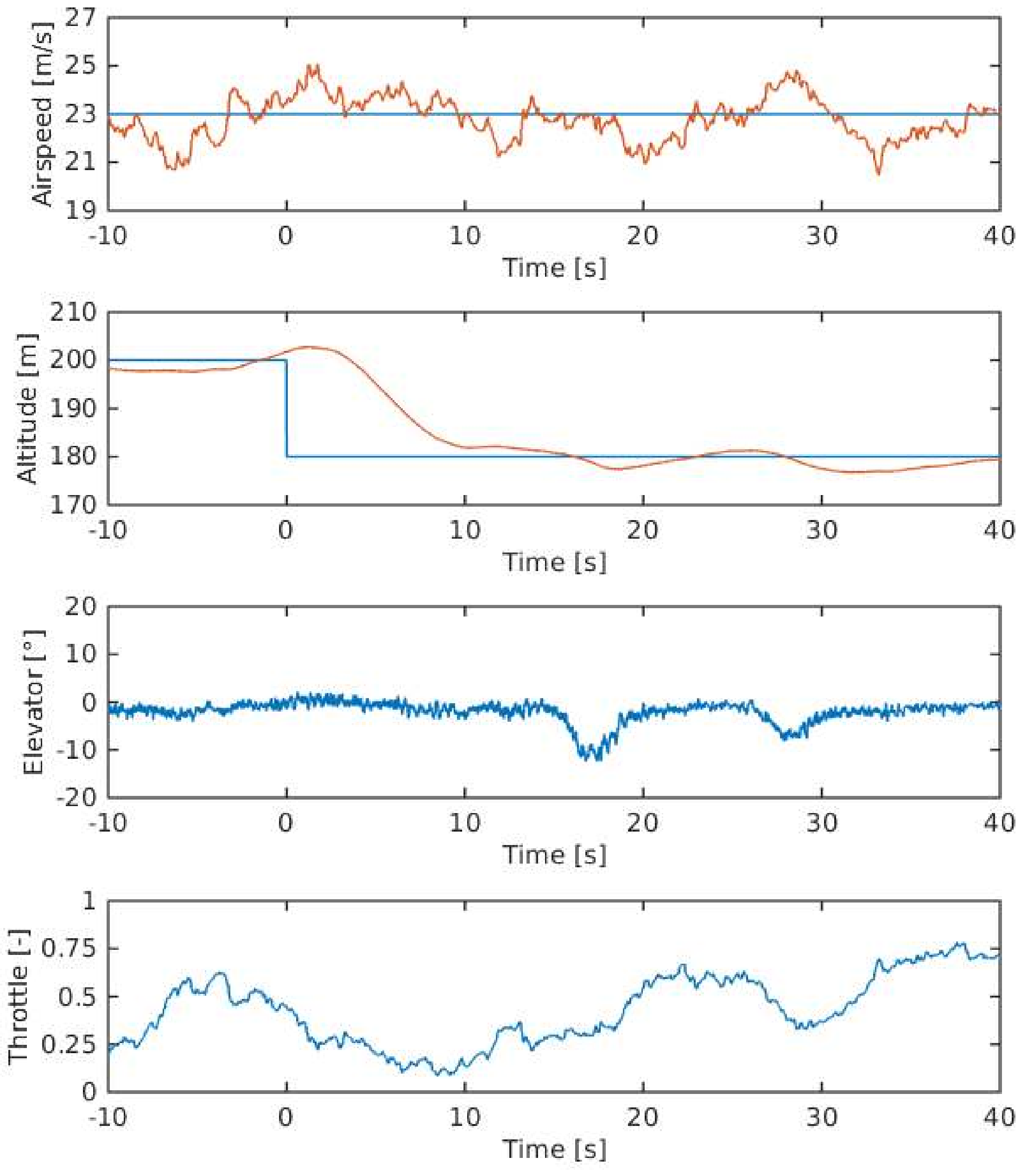

Figure 4.12.: A 20 m descent with the SISO controller. The airspeed tracking seems to be slightly better than during the climb, but it is still far from the TECS performance.

Because of the low γ rate limit, the altitude values of the climb manoeuvre in Figure 4.10 should not be compared directly to the corresponding data from the TECS controller, shown in Figure 4.11. However, the airspeed values can still be compared. If anything, the smooth changes in γ_{des} should improve the airspeed tracking, yet it is still clearly seen that the airspeed drops during climb and overshoots during level off. The TECS controller, on the other hand, maintains the airspeed just as well as during level flight.

The descent manoeuvres, Figures 4.11 and 4.12, show mostly the same result. The speed deviations in the SISO controller are not as big as during the climb, but still clearly visible. Note also how the altitude tracking remains poor after level off, when the γ rate limit does not play a role anymore.

4.2.3. Speed Changes

The plots in Figure 4.13 show the coupling of airspeed and altitude very nicely: at $t = -3\,\mathrm{s}$, the airspeed suddenly increases, likely due to a wind gust. At the same point in time, there is a sudden increase in climb rate visible in the altitude plot. The altitude error persists for a long time, while the speed controller is reacting to the commanded increase in airspeed and thus counteracting the altitude controller's effort to bring the altitude back to the desired value.

The TECS controller handles the speed increase much better, as can be seen in Figure 4.14. However, there are some oscillations between $t = 4\,\mathrm{s}$ and $t = 12\,\mathrm{s}$, mostly visible in the throttle command. If these are not caused by external disturbances, they may indicate a problem with the inner loop thrust controller, and should be investigated further. An improved model of the motor, taking into consideration its internal dynamics, may be useful or required to improve the design of the thrust controller.

The speed decrease with the SISO controller, Figure 4.15, looks similar to the speed increase. In fact, the altitude error is very small, until the speed command is given, where the altitude starts decreasing, showing again the undesired coupling effect. At around $t = 10\,\mathrm{s}$, both variables reach the desired value, but the integrator inside the speed controller is still commanding a high thrust, and the airplane continues accelerating and climbing.

With the TECS controller, the speed decrease is similarly well executed as the speed increase before, as can be seen in Figure 4.16. There is some more variation in the airspeed, but a combination of elevator and throttle inputs keeps it under control. The altitude is maintained nicely by the TECS controller in both speed change manoeuvres.

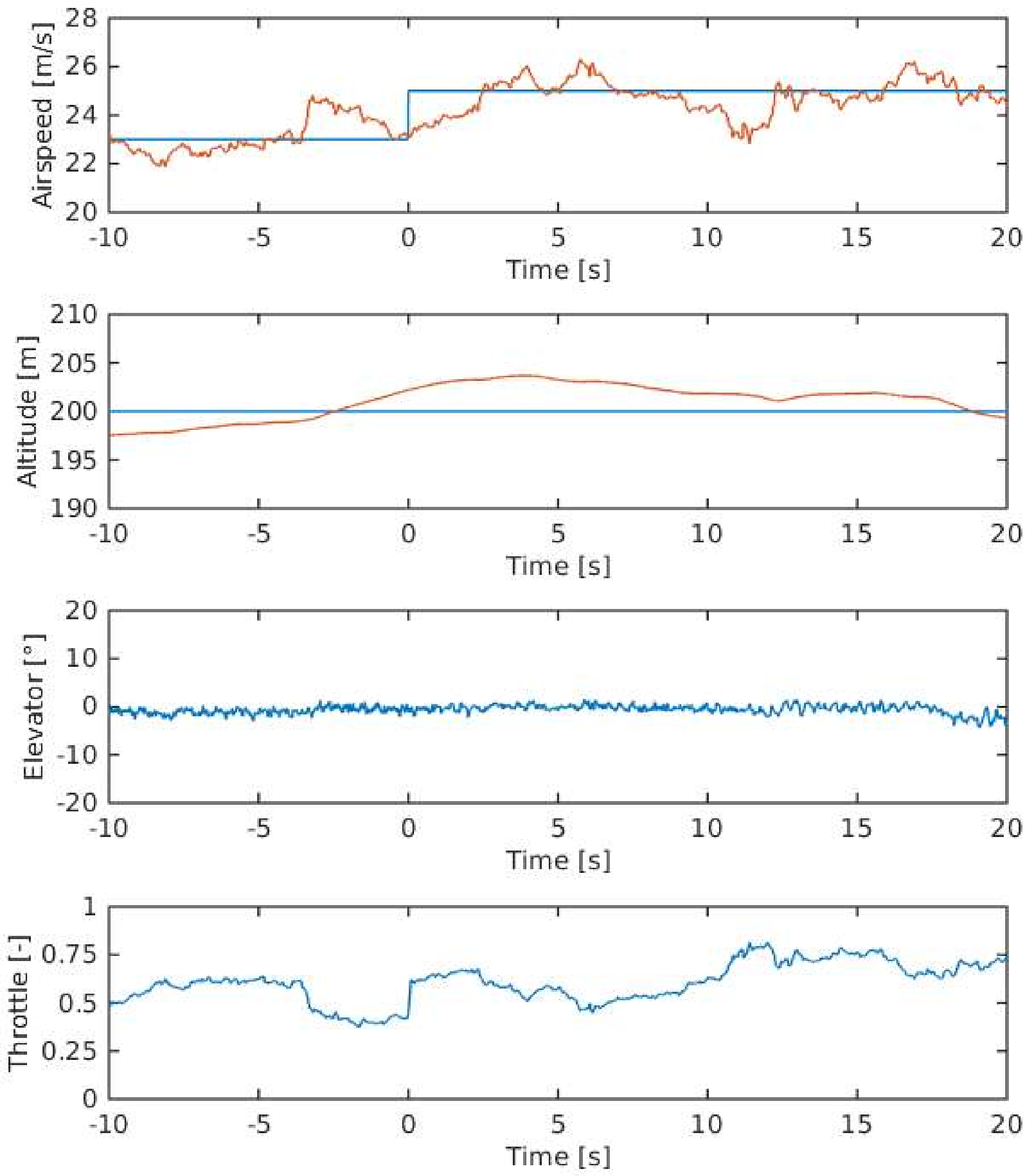

Figure 4.13.: A 2 m/s speed increase with the SISO controller. The coupling of airspeed and altitude is apparent here: at $t = -3$ s, the airspeed increases, possibly due to a wind gust, and instantly the altitude starts increasing. The speed command of course doesn't help, and the correct altitude is not reached again until much later.

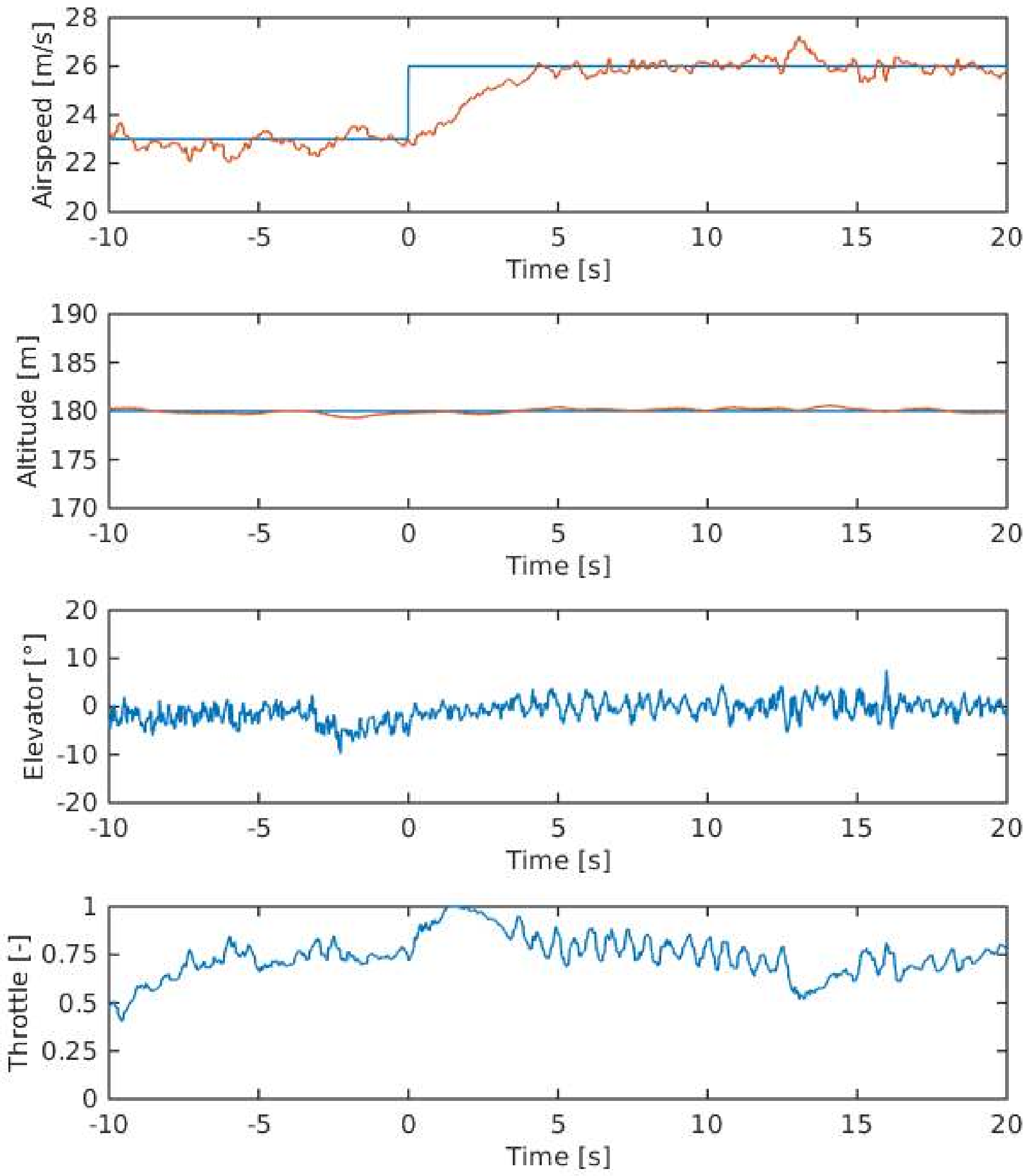

Figure 4.14.: A 3 m/s speed increase with the TECS controller. The altitude command is tracked just as well as during level flight, showing the good coordination of elevator and throttle.

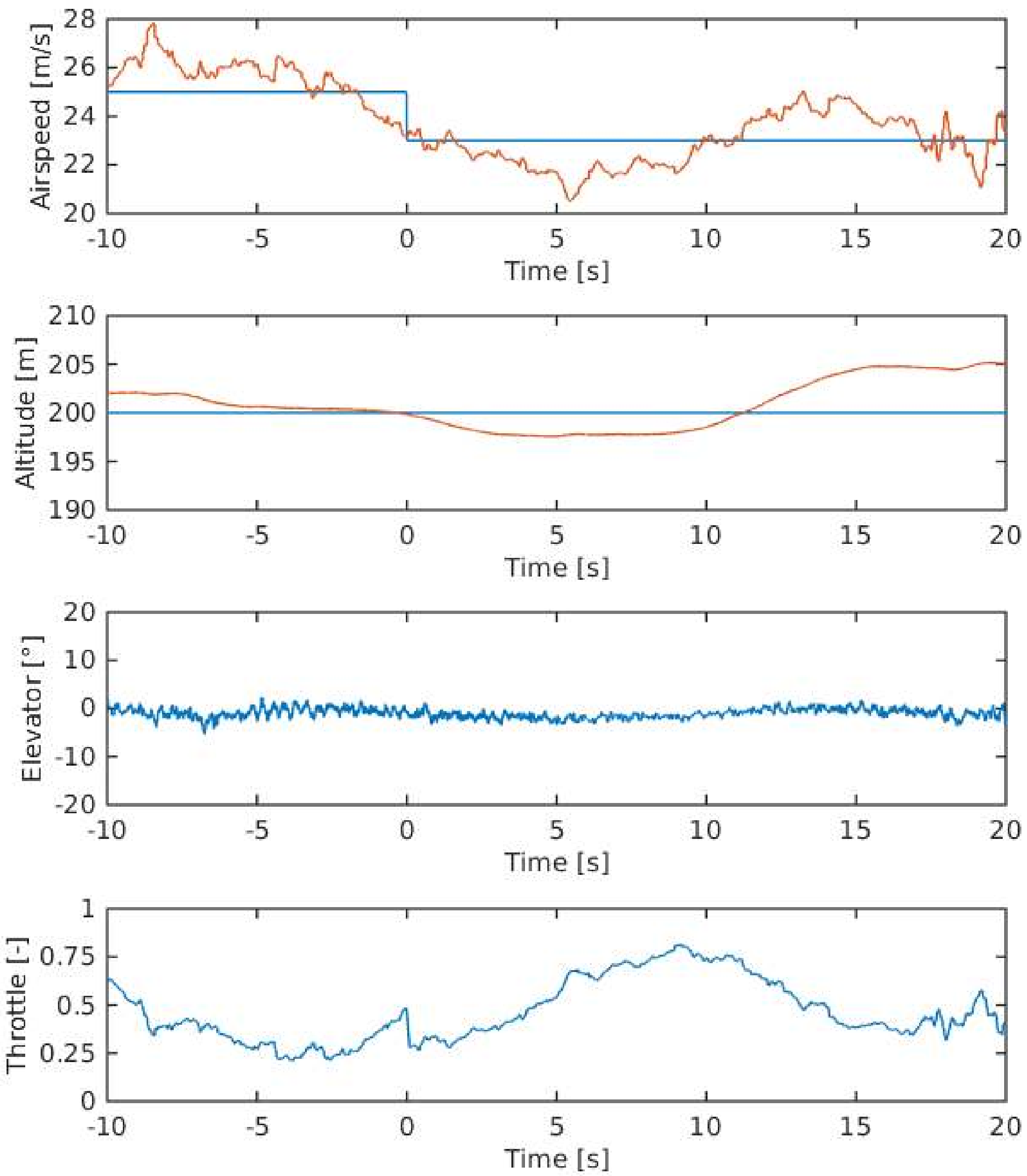

Figure 4.15.: A 2 m/s speed decrease with the SISO controller. Here it seems quite clear that the speed command indirectly causes an altitude error of around 2 m, and the controller has trouble reaching a stable horizontal flight. It seems almost as if altitude and speed controller have to fight each other.

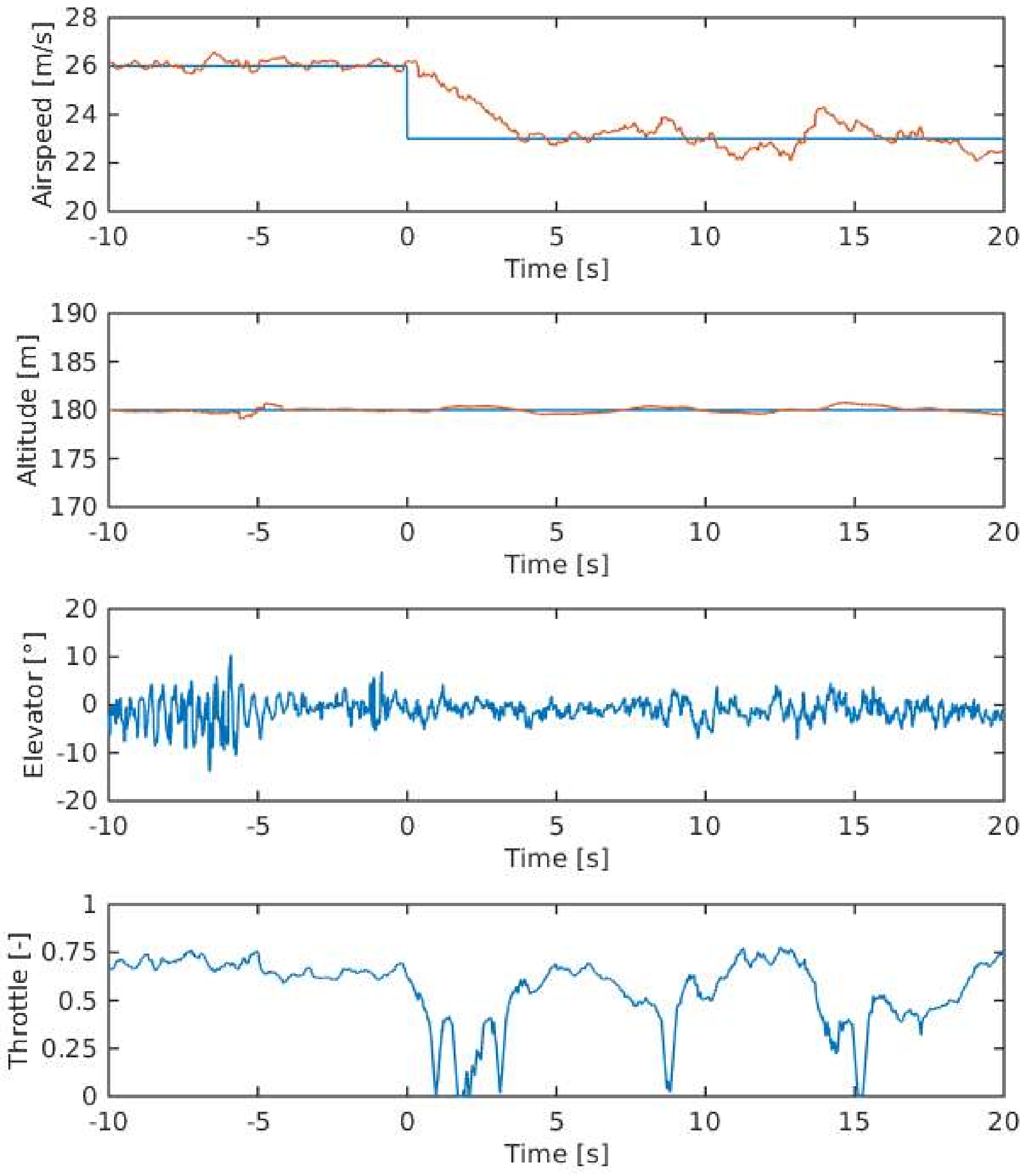

Figure 4.16.: A 3 m/s speed decrease with the TECS controller. Again, good tracking of altitude and airspeed. At $t = 14\,\mathrm{s}$, the airspeed increases, probably due to a wind gust, but it is quickly brought back to the desired value.

5. Conclusion

5.1. System Modeling

A flight dynamics model of the UAV was created. The aerodynamic coefficients were approximated with polynomials, based on data obtained from a potential flow solver, taking into account the wing and tail geometry as well as the airfoil drag polars. The propulsion model was derived from propeller performance values and some test data. The moments of inertia were calculated from the mass distribution, which was approximated by point masses for some of the components, and constant mass per surface for large structural components such as wings and fuselage.

Despite this simplified approach, which has been chosen in order to keep the effort at a reasonable level, a good agreement between the model and log data from flight tests was found. Several manoeuvres were compared, including excitation of the dynamic modes. The period and damping of two modes, phugoid and dutch roll, were determined for the model and the flight test data. The errors of these values were well below 10%, except for the phugoid damping, which was significantly lower (37%) in the simulation compared to the flight data.

5.2. Flight Control System

Two flight control systems were developed: one consisting of three SISO loops for altitude, airspeed and course, as it was previously used for fixed-wing UAV at DLR, and new controller with a MIMO approach for altitude and airspeed, based on total energy principles (TECS). The two controllers were compared in simulation and in flight tests, and the TECS controller performed significantly better in both altitude and airspeed. The desired decoupling of altitude and airspeed was achieved, and the TECS controller will be used for landings on a mobile landing platform in the near future. With the new control system, envelope protection features were introduced, which are expected to increase the flight safety.

5.3. Future Work

Within the Flying Robots Group at DLR, the work on the cooperative control of UAV and ground vehicle continues, with first landing experiments planned before the end of the year. While both the flight dynamics model and the TECS control system are considered usable for this purpose, they can of course still be improved.

In the aerodynamics model, the fuselage aerodynamics could be included, which would have an impact on the drag coefficient, and would potentially improve the accuracy of the phugoid damping. The servo actuators as well as the motor could be modelled in a more realistic way, which would give a greater accuracy in the model's response to control inputs. Also sensor dynamics could be included. Finally, the control system's performance may be enhanced by an analytic optimization of the control parameters, or by using a different control concept for the inner loops, e.g. nonlinear dynamic inversion.

Bibliography

[1] T. Muskardin, K. Kondak, J. Steffes, and R. Haas. Scenario 3 risk assessment and concept validation based on analysis, simulation and preliminary experiments. EC-SAFEMOBIL project deliverable, April 2015.

[2] Andreas Klöckner. Geometry Based Flight Dynamics Modelling of Unmanned Airplanes. In *AIAA Modeling and Simulation Technologies Conference*, Boston, MA, 19-22 August 2013. American Institute of Aeronautics and Astronautics. AIAA 2013-5154.

[3] Mark Drela and Harold Youngren. Athena Vortex Lattice (AVL). `http://web.mit.edu/drela/Public/web/avl/`. Accessed 27 August 2015.

[4] Mark Drela. XFOIL. `http://web.mit.edu/drela/Public/web/xfoil/`. Accessed 27 August 2015.

[5] John B. Brandt and Michael S. Selig. Propeller Performance Data at Low Reynolds Numbers. In *49th AIAA Aerospace Sciences Meeting*, Orlando, FL, 4-7 January 2011. American Institute of Aeronautics and Astronautics. AIAA 2011-1255.

[6] Unmanned Dynamics LLC. AeroSim Blockset. `http://www.u-dynamics.com/aerosim/default.htm`. Accessed 27 August 2015.

[7] National Imagery and Mapping Agency. *Department of Defense World Geodetic System 1984*, third edition, July 1997. NIMA TR8350.2, amended June 2004.

[8] A. Chulliat, S. Macmillan, P. Alken, C. Beggan, M. Nair, B. Hamilton, A. Woods, V. Ridley, S. Maus, and A. Thomson. The US/UK World Magnetic Model for 2015-2020. Technical report, National Geophysical Data Center, NOAA, 2015.

[9] National Oceanic and Atmospheric Administration, National Aeronautics and Space Administration, and United States Air Force. *U.S. Standard Atmosphere, 1976.* U.S. Government Printing Office, Washington, D.C., October 1976.

[10] Rudolf Brockhaus, Wolfgang Alles, and Robert Luckner. *Flugregelung*. Springer, third edition, 2011.

[11] Anthony A. Lambregts. Vertical Flight Path and Speed Control Autopilot Design Using Total Energy Principles. In *Guidance and Control Conference*. American Institute of Aeronautics and Astronautics, August 1983. AIAA 1983-2239.

[12] Anthony A. Lambregts. Generalized Automatic and Augmented Manual Flight Control. Berlin Technical University Colloquium, 19 May 2006.

[13] L. F. Faleiro and A. A. Lambregts. Analysis and tuning of a 'Total Energy Control System' control law using eigenstructure assignment. *Aerospace Science and Technology*, 3(3):127–140, April 1999.

A. Penguin BE UAV Specifications and Performance Data

The specifications and performance data presented here are reproduced from the manufacturer's website[1] and are for reference only. The actual aircraft used was customized by DLR and may deviate slightly from the data presented here.

Table A.1.: Penguin BE UAV specifications and performance.

Specifications	
Empty mass[2]	14.9 kg
Payload capacity	6.6 kg
Maximum takeoff mass	21.5 kg
Wing span	3.3 m
Length	2.27 m
Wing area	0.79 m^2
Propulsion type	electric
Propulsion power	2700 W
Battery type	Li-Po
Battery capacity	640 Wh

Performance	
Cruise speed	22 m/s
Stall speed[3]	13 m/s
Max level speed	36 m/s
C_L max (clean wing)	1.3
C_L max (45° flaps)	1.7
Takeoff run[4]	30 m
Endurance[5]	110 min
Ceiling	6000 m

[1] `http://uavfactory.com/product/69`, retrieved September 22, 2015
[2] including battery and standard landing gear
[3] sea level altitude, 15°C, 15 kg total mass, with flaps
[4] sea level altitude, 15°C, 15 kg total mass, concrete runway
[5] belly landing configuration, 2.8 kg payload

TRITA -MAT-E 2015:77
ISRN -KTH/MAT/E--15/77--SE