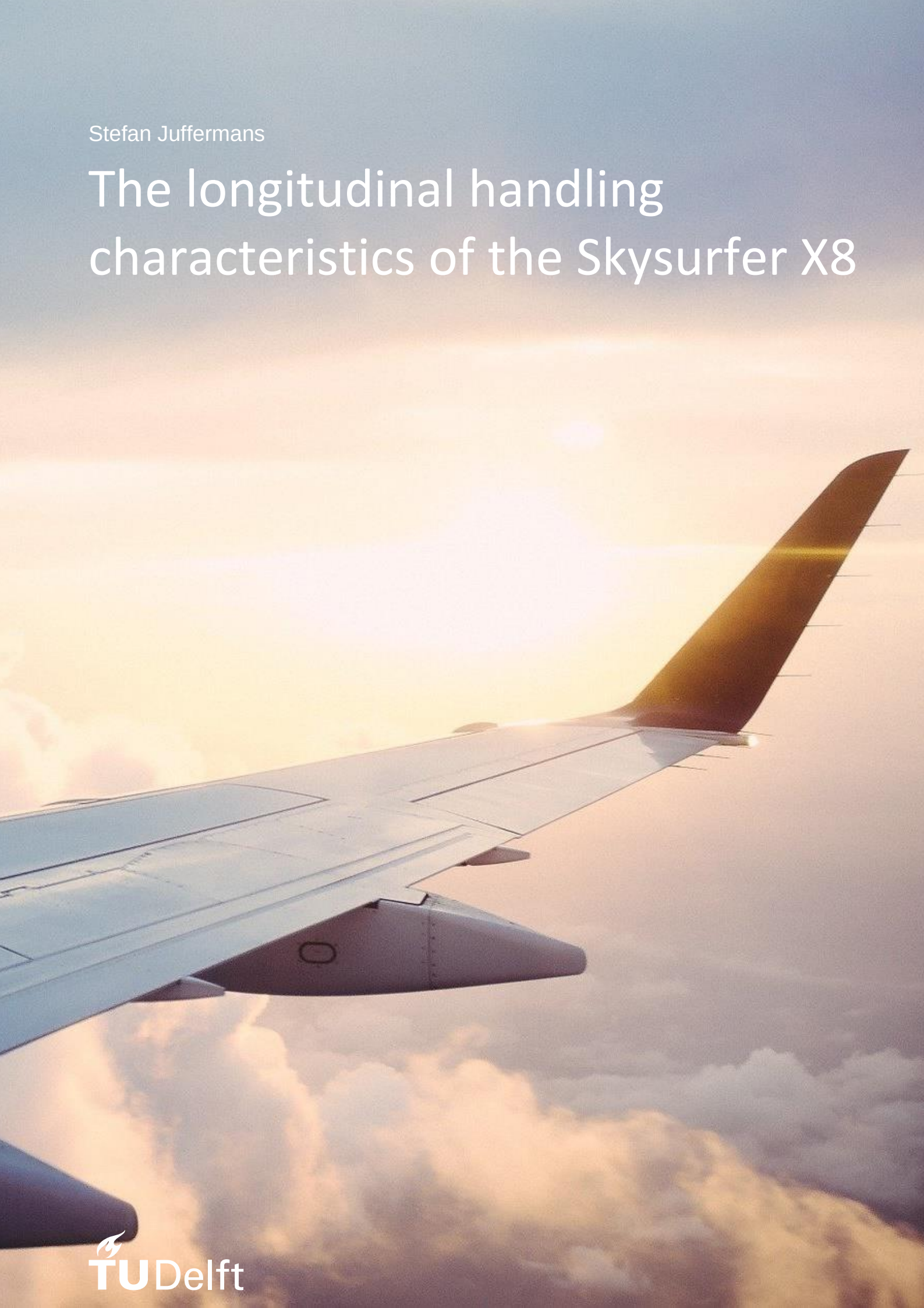
Stefan Juffermans

The longitudinal handling characteristics of the Skysurfer X8



TU Delft

The longitudinal handling characteristics of the Skysurfer X8

By

Stefan Juffermans

in partial fulfilment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at Delft University of Technology,
to be defended publicly on Thursday; April 1, 2021 at 09:00 AM.

Supervisor:	Dr. ir. R. Vos,	TU Delft
Thesis committee:	Prof. dr. ir. L.L.M. Veldhuis,	TU Delft
	Dr. ir. D.M. Pool,	TU Delft

TU Delft

Abstract

For decades the physical appearance of aircraft remained more or less unchanged. The increase in efficiency which still can be made by improving this design, is limited. More researchers have started to investigate new design options that represent a departure from established concepts. The main drivers behind these endeavors are the tightening regulations on emissions and noise as well as the expected profitability for the customer. With the use of scale models for flight testing, it is possible to validate novel aircraft concepts while minimizing investments in terms of costs and time. Novel aircraft often have unproved handling characteristics which is why a quick method to verify these new models would be beneficial.

The objective of this research is to investigate the longitudinal handling characteristics of the Skysurfer X8 by assessing whether the criteria for the short period damping, phugoid damping, bandwidth and Control Anticipation Parameter are in accordance with the military standards. Mathematical models for the aircraft's behavior are extracted from the aircraft by performing system identification in the frequency domain. This is done by measuring the aircraft's input and output signals while performing frequency sweeps on one of the control inputs during a flight test. These measurements are used to derive the input-output system. This procedure is followed for every longitudinal control variable of the aircraft. The models derived from the flight test are validated by comparing the handling characteristics (short period damping, phugoid damping and Control Anticipation) to a simulation performed in XFLR5. The frequency identification method described there is validated with frequency sweep data generated from an aircraft’s known state space system and the flight tests are performed with a modified Skysurfer X8 which is equipped with two wing mounted engines.

Based on the validation the method proves to be reliable. The experiment shows that the handling characteristics of the Skysurfer X8 are at least at level 2 of the military standards. This level is satisfactory, given the small size of the test aircraft. The method is capable of capturing a model of the pitch angle and pitch speed response relative to elevator deflection, which is used to estimate all handling characteristics. Improvements can be made in the frequency sweep maneuvers for the throttle and the measurement error and resolution should be further investigated.

Acknowledgements

I would like to thank my family and friends for their long-lasting support, for their listening to all my complaining when something did not go according to plan, which happened a lot, for their encouragement and their love. In particular I would like to thank my girlfriend, Monique who always was around to cheer us up with a freshly baked cake or pie. I also would like to thank my parents for their long-lasting support every time the deadline got delayed again and again, Alexandro, Ingmar, Joyce and Monique who helped me in creating a readable report and my friends of: FPP, omdat het kan and The LongIsland icetea borrel for all the games we played, other fun events and coffee we drank at the faculty while that was still possible.

Furthermore, I would like to thank the people who helped me to get the project off the ground. For space, tools and advice: Victor, Johan, Johan, Fred and Frank of the Delft Aerospace Structures and Materials Laboratory. Also, my project mates Alexandru, Jan-Willem and Nelson, for all the adventures on our trips to airfields, places to practice our flights skills and the hardware store and the time we spent building our models and working in the office. I would also like to thank ir. M.J. (Michiel) Schuurman and Dr.ir. O.K. (Otto) Bergsma for their countless number of visits and their comments on and talks about the project and the life lessons. Of course also Julian, with whom we often shared the workspace, who gave us practical advice and helped us out when we needed to work during weekends. Another big thanks to Erik who was always available for advice when big problems occurred and who could always point us in the right direction. Then there are of course my former colleagues Chris and David who visited the project and who gave us a crash course on PX4, Jan-Willem of the NLR as pilot in command who helped us with airport regulations and our first tests and the people of Aviolandia and Space53 who made testing at Woensdrecht and Twente Airport possible. And all of this would not have been possible without Roelof, who gave me a lot of freedom and the necessary support to create this project. Thank you all for making this possible.

Stefan Juffermans

Leiden, March 2021

Contents

List of figures

List of tables

Nomenclature

A	Amplitude of oscillation	[-]
b	Semi span	[m]
g	Standard gravity	[m/s^2]
k	Reduced frequency	[-]
K_n	Normal acceleration gain	[-]
K_u	Forward speed gain	[-]
K_w	Downward speed gain	[-]
K_θ	Pitch gain	[-]
M, F, G, H_0, H_1	Generalized equations of motion matrices	[-]
m	Mass	[kg]
M_{δ_e}	Moment gain relative to elevator deflection	[-]
n_z	Normal accelerometer	[-]
$n_{z_{\delta_e}}$	Normal accelerometer response gain	[-]
s	Laplace variable	[-]
t	Time	[s]
T	Time constant	[s]
$T_{\theta_1}, T_{\theta_2}$	Incidence lag constant	[s]
q	Pitch speed	[deg/s]
U, W	Velocity components forwards and downwards	[m/s]
V	Velocity	[m/s]
x	State vector	[-]
X, Z, M	Forward speed, downward speed and pitch terms	[-]
α	Angle of attack	[deg]
δ_e	Elevator deflection	[deg]
δ_t	Throttle setting	[%]
θ	Pitch angle	[deg]
$\varsigma_p, \varsigma_{sh}$	Damping ratio of phugoid and short period	[-]
τ_e, τ_t	Time delay elevator and throttle	[s]
ω	Frequency	[rad/s or Hz]
ω_p, ω_{sh}	Phugoid and short period frequency	[rad/s]

Subscripts

u, w	Forward and downward speed
q	Pitch speed
$\dot{w}$	Downward speed
δ_e	Elevator deflection

1 Introduction

After years of expansion and growth of the air travel industry, the COVID-19 pandemic (Coronavirus disease-2019) caused the number of passengers traveling by air to decrease. US airline passengers numbers show a dramatic drop in March until July through August 2020 compared to August 2019 through February 2020 (Figure 1.1). Due to this pandemic, governments are forced to lockdown their countries and/or restrict the travel in and out of their country, to make sure that hospitals do not get flooded by too many patients at once.

This pandemic will cause us to think differently about the world and question if it will ever be the same as before. This holds true for air travel in particular. Is it necessary to fly that often for businesses purposes or might the same goals be achieved by video calling? Do we need to travel for work at all or can we work from home as well? Is it necessary to fly across the world to relax on a beach or is that also possible in a beach house close to home? Do we need to transport food by air or is it also possible to produce it close to home? At the moment people are forced to live their life in a different manner and this will change our behavior and what we want.

Aviation companies will be affected by COVID-19 as are their customers. Due to the drop in passenger numbers, the aviation industry is expected to have a harsh year or even years. If the current situation will continue, it will cause companies to fail due to their lack of financial resources because orders for new aircraft will be cancelled or delayed. Before COVID, the industry made flying cheaper for their customers and by increasing the number of passengers the companies increased their profit. The question is if this can be sustained in the coming decades.

Another effect which might influence our way of traveling is the emission of greenhouse gases caused by this form of transport. For years scientists have been warning against problems caused by a rise of global temperatures as a result of emissions by aircraft and other polluters. This is slowly showing its first effects on the planet: Droughts cause large wild fires and the ice caps on the North pole are decreasing every year. Little action is taken because these effects seem to be far away. When these effects will become stronger it is expected that the urge to search for a solution will increase.

After the pandemic there are two options to keep aviation viable. The first is to drastically decrease air travelling either by human insight, or by marketing strategies such as price increases. The second is to rigorously change aircraft design in order to reduce emissions, accepting an ever-growing demand for air travel. The latter would imply a radical transformation of the aviation industry. To improve efficiency and reduce the emission of pollutants it will be necessary to design a totally new type of aircraft [1]. An improved design must do better on the following aspects, compared to the conventional design: Aerodynamics, structure and propulsion. Therefore, new configurations and methods of transport by air will have to be researched and tested quickly at low costs. One way to do this is to use a subscale model of the new design in order to verify if the model will meet the expectations.

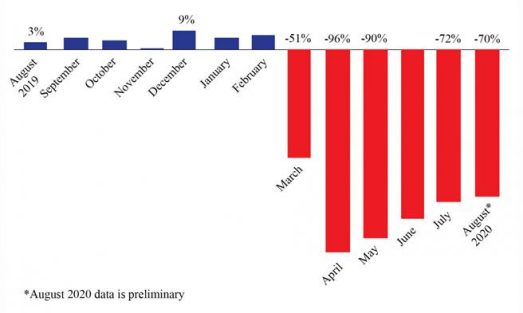


Figure 1.1: Increase of airline passengers August 2019 until August 2020[1]

[1] https://www.bts.gov/newsroom/us-airline-august-2020-passengers-decreased-70-august-2019-rose-2-july-2020-preliminary

1.1 Flight testing to find the aircraft of the future

In recent years, many tests using new aircraft concepts were performed by aircraft manufactures[2], research institutes [2, 3] and universities [4-6]. These new aircraft designs need a quick proof of concept, to evaluate the feasibility of the design. One way to do this, is by developing a scale model of the aircraft, which can verify computer simulations at low costs and little time. One of the main interests of flight testing is to investigate the handling characteristics of the aircraft, these are "the qualities or characteristics of an aircraft that govern the ease and precision with which a pilot is able to perform the tasks required in support of an aircraft role." [7].

At this moment the Flying-V [8], a new flying wing concept that is studied at Delft University of Technology, is at the stage of flight testing. A lot of research, which consists of numerical simulations and wind tunnel tests [9, 10], has already been put into the aerodynamic [11, 12], structural [13, 14] and propulsive [9, 15] performance of the aircraft. One key question so far unanswered is how the aircraft will behave in the air. Initially the goal of this research was to investigate the handling characteristics of the longitudinal dynamics of a scale model of the Flying-V but in spite of great efforts the required data from a flying scale model of the Flying-V unfortunately could not be obtained. The tests in this research are therefore executed with the Skysurfer X8. This research can be used as a guideline to design the experiments for the Flying-V.

1.2 Handling characteristics

The handling characteristics of aircraft have been investigated for many years as these are major drivers in the design process. It all started with pilot-based tests, often investigated with the Harper and Cooper scale [7, 16, 17] in which a pilot is asked how the aircraft behaves during certain flight conditions or a maneuver. Harper and Cooper set up a structured set of questions regarding handling characteristics for the pilot. Figure 1.2 shows the questions to get to the pilot rating. The pilot rating ranges from 1-10, with a pilot rating of 1 being highly desirable, and a pilot rating of 10 showing that the behavior of the aircraft has major deficiencies.

A pilot-based test is a very natural way to investigate the handling characteristics of an aircraft, because the pilot is the person who has to deal with the handling of the aircraft in the end. From a research prospective however it is interesting to determine objective and measurable characteristics that can be used to compare aircraft, for example: Bandwidth/phase delay [18], C* [19], CAP [18], Gibson [20], Neal/Smith [21, 22] and Smith/Geddes [23] criteria. Table 1.1 shows which criteria should be used during which stage in the design process, as described by Edmund J. et al. in “Landing Approach Flying Qualities Criteria For Active Control Transport.” [24]. This paper suggests to investigate the CAP and Bandwidth in the flight test phase. These two criteria are also proposed for assessing the handling characteristics of unmanned aircraft candidates [25]. Therefore, these two criteria are investigated including the short period damping and phugoid damping criteria which are prominently stated in the military specifications.

[2] https://www.airbus.com/newsroom/press-releases/en/2020/02/airbus-reveals-its-blended-wing-aircraft-demonstrator.html

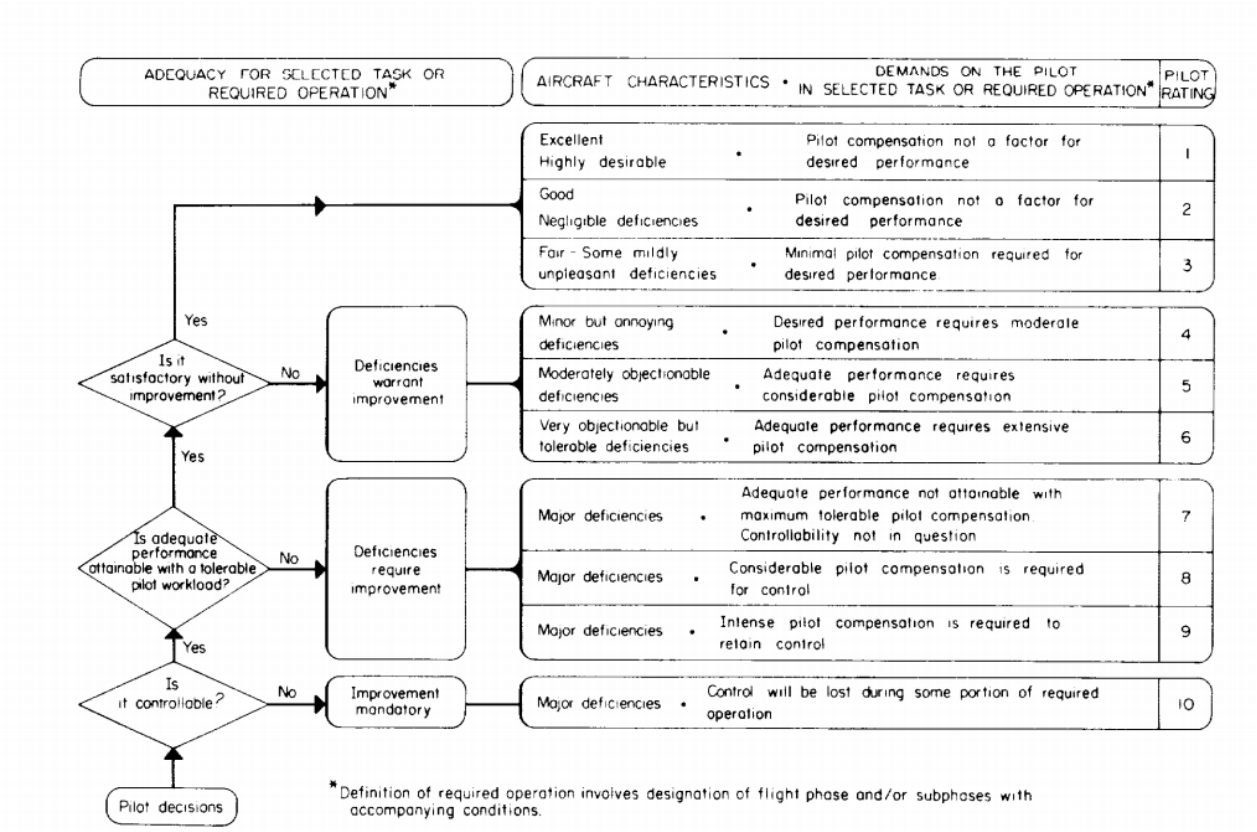


Figure 1.2: Handling characteristics rating scale of Harper and Cooper [7]

Table 1.1: The handling criteria interesting for the design phase [24]

	Initial design	Detailed design	Pre-flight evaluation	Flight test
CAP	X	X	X	X
Bandwidth	-	-	X	X
Gibson	X	-	X	-

The handling characteristics are evaluated conform the military standards MIL-F-8785C/MIL-STD-1797A [26, 27]. The military standards are chosen because they are the most commonly used standards for the handling criteria in both military and civil aircraft. The criteria are rated in three levels indicated below.

Level 1: The qualities are clearly adequate for the mission Flight Phase.
Level 2: The flying qualities are adequate for the mission Flight Phase, but either a slight increase in pilot workload can be noted or a decrease in mission effectiveness, or both.
Level 3: Flying qualities enable the airplane to be controlled safely but either pilot workload is excessive, or mission effectiveness is inadequate. [26, 27]

The parameters/handling characteristics of interest for this research are the short period damping, the phugoid damping, bandwidth and the Control Anticipation Parameter (CAP). These parameters can be extracted from the transfer functions and state space models derived during a flight test. The pitch response can be extracted from flight tests by deriving a model for the pitch angle or pitch speed relative to the elevator deflection. By measuring the elevator input and pitch angle or pitch speed output during a flight, a mathematical model which replicates the behavior of the aircraft

can be derived. This model is represented in a transfer function. The criteria for the damping ratio for the short period and phugoid, are stated in Figure 1.3.

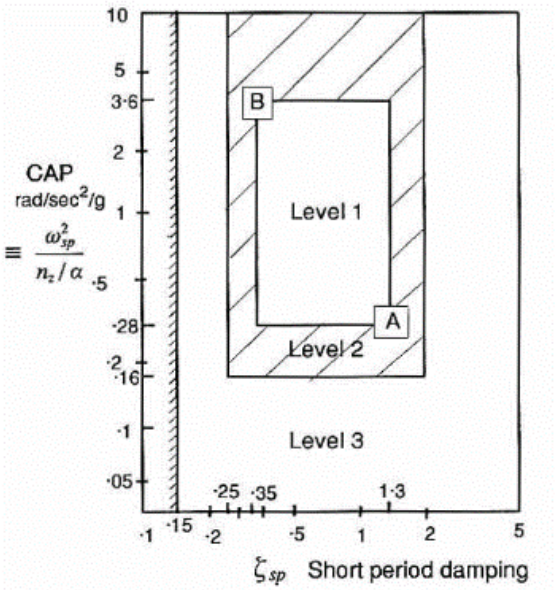


Figure 1.3: CAP criteria and short period criteria [26,27]

Table 1.2: Short period and phugoid criteria [26, 27]

	Level 1: Damping ratio [-]		Level 2: Damping Ratio [-]		Level 3: Damping ratio [-]	
	Min.	Max.	Min.	Max.	Min.	Max.
Phugoid	0.04	-	0	-	$T_2 \geq 55\text{sec}$	
Short period	0.35	1.30	0.25	2.00	0.15	-

CAP indicates the ability of the pilot to precisely control the flight path. It is also called: "The ratio of an aircraft's initial pitching acceleration to its change in Steady State acceleration" [18] as shown here:

$$CAP = \frac{\ddot{\theta}_0}{\Delta n_{z_{ss}}} \tag{1}$$

This metric gives an indication of the pitch response to the force on the pilot and aircraft. How this parameter can be extracted from the flight test results is shown in the method section. For the CAP, a graph is made to indicate the flight levels which also contains the damping ratio criteria (Figure 1.3).

The bandwidth criterion shows until which frequency the pilot is able to exert a signal to the control surfaces without causing instabilities of the aircraft. Within the boundaries of the bandwidth criterion, the pilot has good close loop control without excessive pilot demand and the aircraft's attitude can be precisely controlled. These instabilities are often referred to as Pilot Induced Oscillations (POI). For the longitudinal case the elevator defection and the pitch angle are investigated. This method is a "pilot in the loop analysis" in which the pilot is modelled as a gain. The phase delay and the attitude bandwidth are the two main parameters to classify the level (Figure 1.4). The derivation of these parameters is shown in the method section.

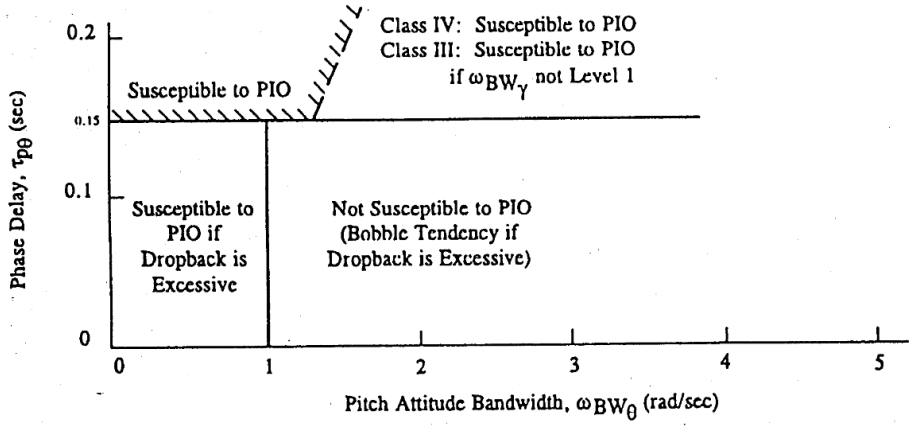


Figure 1.4: Bandwidth criterion for categories B and C [28]

Because this investigation is on an Unmanned Aerial Vehicle (UAV) and the handling characteristics are defined for manned aircraft (which are obviously larger than the investigated aircraft), there is a difference in the handling characteristics of these smaller aircraft [25, 29, 30]. UAVs become more affordable and common these days resulting in a larger demand for handling characteristics specifically designed for UAVs or an adjustment of these characteristics for the size. The latter is already proposed [31], it is possible to scale the handling characteristic parameters from the UAV to a larger size and then verify the handling characteristics on a larger scale. The aircraft in this research is not derived from a larger manned aircraft, making it impossible to scale the values. To adjust for this constraint, it is assumed in this research that both level 1 and level 2 handling characteristics are satisfactory. This widens the criteria, nonetheless level 1 handling characteristics are preferred.

1.3 System identification

To calculate the handling criteria, a mathematical representation of the dynamics of the aircraft is necessary to extract the parameters for the calculation of the criteria. These models are extracted from the flight test data by system identification in the frequency domain. This method is chosen because of its low number of required test maneuvers, the information density of the results and the promising results of prior research [32-35]. The models are derived from the input and output measurements of the aircraft during a frequency sweep, which is done for Single Input Single Output (SISO) models and for Multiple Output Multiple Input (MIMO) models. The SISO models are represented as transfer functions and the MIMO models are represented as state space models and a set of transfer functions.

A frequency sweep is a sinusoidal signal with a ranging frequency (Figure 1.5). Frequency sweeps are used to extract a Linear Time Invariant (LTI)-system from the input and output signals. Because of the behavior of an LTI-system and sinusoids, the system can only change the input signal to the output signal in magnitude and phase depending on the frequency of the input signal. The magnitude and phase change relative to the frequency, which is what is shown in a bode plot and therefore these plots can easily be derived. Another advantage of this system is that in one test, a range of frequencies are tested, while in time domain testing the right frequency has to be chosen to extract the dynamics of interest, which often results in more test flights or assumptions[32].

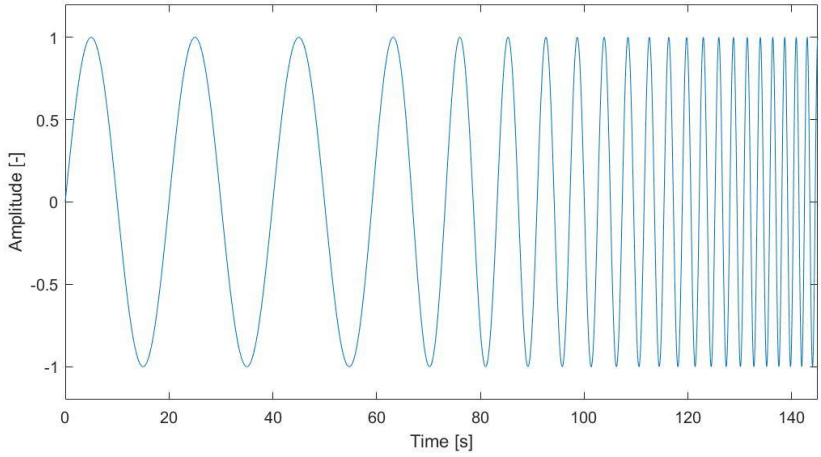


Figure 1.5: Example of frequency sweep

Combining frequency sweeps with system identification is an ideal match and often used to extract the dynamics of air vehicles [33, 34, 36, 37]. Comprehensive Identification from Frequency Responses (CIFER) [32] is a software package which assists to investigate the flight test data in the frequency domain. CIFER structures the input and output data from the different frequency sweeps and doublets from the flight test. The flight test data is converted from time domain to the frequency domain, resulting in the bode plots for the responses. By fitting a transfer function on the bode plot a SISO model can be estimated. By performing this for multiple input and output responses, a MIMO model can be derived which is represented in a state space model. To validate the derived models, the response of these models can be compared with the actual response of the aircraft (with the doublets from the flight test).

1.4 The Skysurfer X8

Due to the unavailability of test results from a scaled model of the Flying-V, the aircraft used for this research is the Skysurfer X8[3] (Figure 1.6). This model is chosen because of the affordability of this aircraft and the favorable controllability for beginning pilots. This simplified the test setup, as there was a low financial risk and it was unnecessary to have an experienced pilot present at the flight test location.

Figure 1.6: Skysurfer X8[3]

[3] https://nl.banggood.com/X-UAV-Sky-Surfer-X8-1400mm-Wingspan-FPV-Aircraft-RC-Airplane-KIT-p-1064615.html?cur_warehouse=CZ

1.5 Research goal and research questions

The objective of this research is to investigate if the longitudinal handling characteristics of the Skysurfer X8 are satisfactory for at least level 2 of the following criteria: The phugoid damping, short period damping and Control Anticipation parameter (CAP) according to the military standards. To do this, the following research questions have to be answered:

1. What are the state space and transfer function models for the longitudinal dynamics of the Skysurfer X8?
1.1. What are the transfer function models for the longitudinal dynamics derived from the simulation and flight test?
1.2. What is the state space model of the aircraft's longitudinal dynamics determined from the simulation and flight tests?
1.3. How accurate are the state space and transfer function models of the aircraft determined from a simulation and flight test?
1.4. What is the uncertainty of measurements gained from the flight tests?

2. Are the longitudinal handling characteristics of the Skysurfer X8 satisfactory (at least level 2) according to the military specifications?
2.1. What are the longitudinal handling characteristics according to the derived transfer functions and state space matrices for the damping ratio of the phugoid and the short period and the CAP?
2.2. How do the handling characteristics of the aircraft compare to the military specifications on the damping ratio of the phugoid and the short period and the CAP?

2 Method

In this section the setup of the experiment for this research is shown. The longitudinal handling characteristics of the Skysurfer X8 are determined by capturing the dynamics of the aircraft by system identification, performed on the results of a flight test. The model of interest for this research is the longitudinal dynamic system, represented as a set of transfer functions and a state space system. These transfer functions and the state space system are utilized to verify if the handling characteristics are conform the handling criteria stated in the military conditions. The flight test results are compared to the results from a simulation, which is performed in XFLR5. The results from the simulation contain the transfer functions and a state space model for the longitudinal dynamics wherefrom the handling characteristics are derived.

2.1 Flight test vehicle

In this research the Skysurfer X8[4] is used, a beginners Radio controlled (RC)-aircraft which is easy to fly with minimal experience. The aircraft has a conventional design with a fuselage, wings and horizontal and vertical tail (Figure 1.6). The aircraft type is common and produced by multiple manufacturers, for example the Hobby King Bixler[5] and Multiplex Twinstar[6], resulting in the benefit that research is already performed on these aircraft types. Examples of prior research on the Bixler are: The course (Design, Construction, and Testing of Autonomous Aircraft) problems of the Stanford University for the teams: Drone Identity Problem set 2[7], Chimera Problem set 2[8] and Fregata Problem set 2[9]. Examples of prior research on the Twinstar which is the research aircraft of the Georgia Institute of Technology [38-40].

The Skysurfer X8, used in this experiment, is equipped with a Pixhawk 4 including a GPS and airspeed sensor to make the aircraft ready for flight testing. These modifications make it possible to measure the velocity, acceleration, angles and angular accelerations of the aircraft. The aircraft is altered with two engines on the wings instead of one engine on top of the fuselage. The two engines are installed in order to increase the propellent redundancy of the aircraft and allow future one engine inoperative testing.

2.1.1 The Skysurfer X8

The Skysurfer X8 is a conventional remote-controlled aircraft manufactured by X-UAV. The stock aircraft is controlled by an elevator, a rudder and two ailerons and is propelled by one engine. The important dimensions and recommendations for electronics are summarized in Table 2.1. The aircraft is purchased as a kit, which means that all the necessary parts to build the aircraft are provided, except for the electronics.

[4] http://www.x-uav.cn/en/content/?466.html
[5] https://hobbyking.com/nl_nl/h-king-bixler-1-1-epo-1400mm-glider-pnf.html
[6] https://www.multiplex-rc.de/produkte/1-00912-bk-twinstar-nd
[7] https://sites.google.com/a/stanford.edu/droneidentity/problem-set-2
[8] https://manualzz.com/doc/44476263/aa241x-problem-set-2-report-team-chimera
[9] http://fregata-uav.weebly.com/uploads/7/7/5/8/77584296/problemset2.pdf

Table 2.1: Skysurfer X8 recommendations and dimensions[10]

Parameter:	Recommendation or dimension:
Battery:	3S 11.1V 2200 mAh
ESC:	40A
Flying weight:	610 g
Length:	915 mm
Propeller:	6040
Proposed engine:	2212 KV2200-2300
Servos:	9g
Wing span:	1410 mm

2.1.2 Dimensions of the Skysurfer X8

The dimensions of the Skysurfer are shown in table 4. This table is accompanied with Figure 2.1 which shows the dimensions.

Table 2.2: Aircraft dimensions and airfoil parameters

Parameter:	Dimension/airfoil:	Unit
Aircraft length	915	mm
Horizontal tail airfoil shape	NACA0013	-
Horizontal tail chord	104	mm
Horizontal tail span	470	mm
Main wing airfoil shape	NACA4410	-
Main wing chord	202	mm
Main wing span	1410	mm
Vertical tail airfoil shape	NACA0010	-
Vertical tail chord	120	mm
Vertical tail span	170	mm

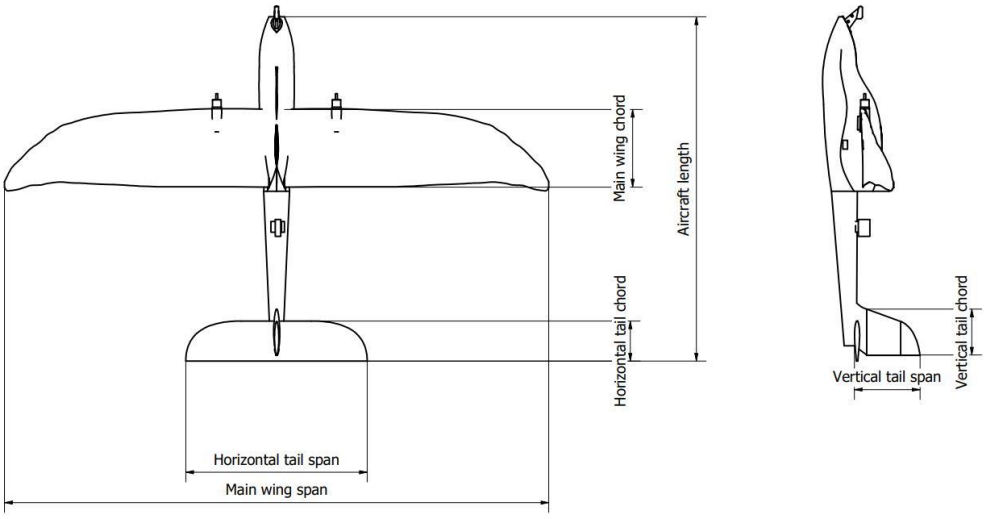


Figure 2.1: Dimensions of the Skysurfer X8

[10] https://nl.banggood.com/X-UAV-Sky-Surfer-X8-1400mm-Wingspan-FPV-Aircraft-RC-Airplane-KIT-p-1064615.html?cur_warehouse=CZ

2.1.3 Modifications

Some modifications were made to the original aircraft to improve the model and to enable flight testing. All servos are placed into brackets which simplifies replacement in case of failure. The aircraft is altered with two engines on the wings instead of one on the fuselage. Brackets for the pitot tube and the telemetry are designed and installed on the aircraft. The modified parts are designed in a CAD program (Inventor) and 3D-Printed. The modifications are explained in more detail in the following paragraphs. The modified aircraft is shown in Figure 2.2.

Figure 2.2: Test aircraft Skysurfer X8

Servo, telemetry and pitot brackets

There are three types of brackets designed to hold the servos, the telemetry and the pitot in place. The technical drawings of all brackets are included in Appendix A. The kit requires the servos to be glued onto the aircraft. To simplify the replacement of the servos, servo brackets (Figure 2.3a) are designed to hold the servo in place. Clamping and screwing makes it possible to attach the servos to the aircraft without glue. The servo location is not altered by the new bracket design. The telemetry bracket is placed on the tail of the aircraft at 544 mm from the nose. The pitot bracket (Figure 2.3b), which is designed to hold the pitot and keep it at a safe distance from dirt while landing, is placed on the nose of the aircraft.

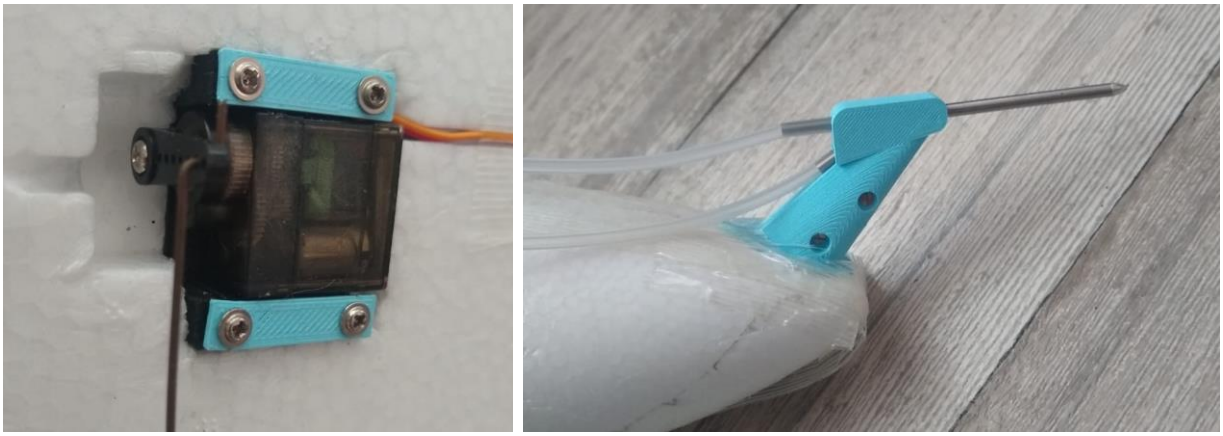


Figure 2.3: Modifications: a) Servo bracket b) Pitot bracket

Engines and attachment

In the original design only one engine is used, which should be mounted on top of the fuselage, right behind the wing. In this setup two engines are used. The engines are placed parallel to and 155 mm away from the center of the XZ-plane of the airplane onto the leading edge of the wing. The mount is designed to hold the engine and electronic speed controller and slide over the wing. The motor is then mounted on the motor mount and is glued onto the wing (Figure 2.4). The electronic speed controller is placed on the motor mount with double sided tape and placed on the bottom of the wing, this way the cables can easily be mounted in the wing.

Figure 2.4: Modifications: Engine mount

2.1.4 Technical details

The aircraft is equipped with electronics to fly, to control the aircraft and to monitor and measure the flight parameters. These electronics are carefully selected and installed to ensure proper working of the systems. In this section the important details of the electronics are explained. The aircraft is equipped with a flight computer, GPS sensor and pitot tube. This makes it possible to measure the groundspeed, airspeed, accelerations, attitude angles, attitude rates, height, pressure and temperature. The system is controlled with a transmitter and a computer, at which all parameters can be monitored.

The electronics used on the aircraft are shown in Table 2.3. Most of the terminology of the equipment needs no explanation, except for the Battery Elimination Circuit and Power Module. The battery elimination circuit is used to supply a 5 volt voltage to the servos and the power module. This is a DC-DC converter which converts the battery voltage to 5V. The Power module is a small electronic system which monitors the battery level of the aircraft and sends this information to the flight controller and powers the flight controller as well.

The radio frequency equipment that is used, is shown in Table 2.3. With these, the signals between the aircraft and the ground are send. There is a connection between the transmitter (the remote control) and the receiver. The receiver is the device in the aircraft which picks up the signal and converts it to the flight controller language. The telemetry forms a second connection, namely between the computer and the aircraft. On the ground and in the aircraft the same telemetry device is used, only in a different setting. This setting can easily be set with a jumper on the header pins of the telemetry device.

Table 2.3: Electronic details of the modified Skysurfer X8

	System:	Brand:	Details:
Flight system:			
Flight controller	Pixhawk 4[11]	Holybro	Main FMU Processor: STM32F765 IO Processor: STM32F100 On-board sensors: • Accel/Gyro: ICM-20689 • Accel/Gyro: BMI055 • Magnetometer: IST8310 • Barometer: MS5611
GPS/IMU	Neo-M8N[12]	Ublox	GPS/GLONASS receiver and integrated magnetometer IST8310
Pitot tube	Airspeed kit[13]	-	Pressure sensor: MS4525DO
Propulsion and control system:			
Engines:	Viking 1808 2600kv[14]	Multistar	2600KV, 485W, 14.4V 3S, max 565 g thrust
Propeller	X4040300[15]	DYS	4040, 3-blades, 3g, clockwise and counter clockwise
Electronic Speed Controllers:	Brushless Speed Controller[16]	DYS	Oneshot 125, 30A (3 ~ 6S)
Control surface servos:	MG90S[17]		Analog coreless servo, torque: 1.8 kgcm, speed: 0.002 °/sec, weight: 13.4g
Power system:			
Battery	3S1P Lipo Battery Pack	Stefanliposhop xtron	3000mAh 11.1V 30C 3S1P Lipo Battery Pack
Battery Elimination Circuit:	CC BEC High performance 6S 10A switching regulator[18]	Castle creations	Max input 25V (6S), Output regulated between 4.8 till 9 V with 0.1 V steps, peak current 10A
Power module:	Hallsensor 100 A[19]	Mauch	Max 6S batteries, 5.3V 3.0A output
Communication system:			
Receiver:	REX12[20]	Jeti	2.4GHz, 100 mW, RC-connection
Transmitter:	DS16[21]	Jeti	2.4GHz, 100 mW, RC-connection
Telemetry:	RFD868[22]	RFD	868MHz, 100 mW, Telemetry

[11] http://www.holybro.com/product/pixhawk-4/
[12] https://www.u-blox.com/en/docs/UBX-15031086
[13] https://docs.px4.io/master/en/sensor/airspeed.html
[14] https://hobbyking.com/nl_nl/multistar-viking-brushless-outrunner-drone-racing-motor-1808-2600kv-ccw.html
[15] http://ftec-shop.nl/shop/FMPro?-db=ftec%20producten.fp3&-format=rs%2Fdetail.html&-lay=cgi&-sortfield=Prijs&-sortorder=descend&Merk=DYS&-max=1&-skip=88&-find
[16] https://nl.banggood.com/DYS-XS-30A-3-6s-Lipo-BLheli_S-ESC-Support-Oneshot125-Oneshot42-Multishot-for-High-KV-Motor-for-RC-Drone-p-1060355.html?akmClientCountry=NL&
[17] https://opencircuit.nl/Product/MG90s-9G-micro-servo-motor
[18] https://www.castlecreations.com/en/cc-bec-010-0004-00
[19] https://www.mauch-electronic.com/hs-sensor-product
[20] http://www.jetimodel.com/en/katalog/New-Products/@produkt/Duplex-REX12-EPC/
[21] www.jetimodel.com/en/katalog/Transmitters/@produkt/DS-16/
[22] http://store.rfdesign.com.au/rfd-868-modem/

The wiring diagram of the Skysurfer X8 shown in Figure 2.5. The wire connections are simplified by showing the direction of the signal and information, indicated by an arrow. The illustrated connections contain multiple cables but are shown as one line to promote readability.

The battery powers the BEC, the engines and the ESCs. In between the battery and the ESCs is the power module which monitors the voltage and the power drawn from the battery and sends this information to the flight controller. The 5 volt of the BEC powers the servo rail and the power module. The power module subsequently supplies power for the flight controller. The flight computer is connected to the telemetry and receiver, from which the commands for flight are send. These commands are translated to Pulse Width Modulation (PWM) signals and send to the servo rail. From the servo rail, the signals for the servos and engines are divided. The servo rail contains for every servo a signal, 5V and ground. The 5V and ground are all interconnected, therefore the 5V and ground from the BEC is supplied to all the servos. Two modules are added to the system, these are an airspeed sensor (pitot tube) and a GPS/compass, which are connected to the flight controller, where the measurements are processed.

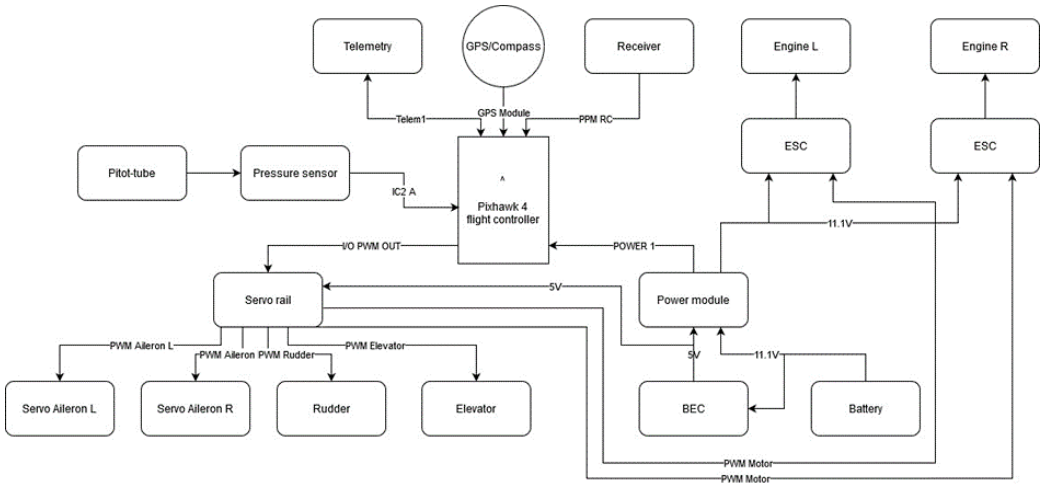


Figure 2.5: Wiring diagram Skysurfer X8

As can be seen in Table 2.4, the aircraft is equipped with a GPS/compass unit, airspeed sensor and flight computer which contain sensors that are used to extract the measurement data. In total there are 6 different sensors of which two accelerometers, two gyroscopes and one airspeed, GPS, temperature and absolute pressure sensor. The flight computer combines these measurements of the sensors to construct a measurement set which contains the needed variables for the analysis.

The accelerometers and gyroscopes measure the acceleration and angular rates in the X, Y and Z direction. The measurement range of these sensors can be changed, by which the resolution will be changed accordingly. The measurement range at which the flight computer operates is unknown, therefore the minimum and maximum range are both shown. The resolution indicates the smallest difference the sensors can measure. The resolution for these sensors is very small and therefore they are very precise; however, nothing is stated about the error of the measurements in the datasheets for these components. The magnetometer measures the magnetic field in the X, Y and Z direction, from which the orientation of the plane can be derived. The strength of the magnetic field is indicated in micro-Tesla: µT. For this component the information on the error lacks as well. The barometer and temperature sensor are well documented, they show a measurement range, resolution and accuracy. The manufacturer states that the measurement error could lead to a 10 cm error in the estimation of the height. The GPS sensor measures the velocity in X, Y, and Z direction, the heading and the horizontal position. The manufacturer only states the accuracy of the measurements. The airspeed sensor measures the pressure difference between stagnation and static pressure and from this pressure difference the velocity can be estimated. This sensor also contains a temperature sensor to adjust the measurement for temperature change. The manufacturer claims a error of maximum 0.25 percent deviation over the measurement range.

The measurements from these sensors are used in the flight controller. The software of the flight controller combines all of these measurements to extract the data for the flight with an Extended van Kalman filter (EKF). This filter combines all measurements at a specific timepoint and estimates the next position of the aircraft from these measurements. The calculated estimated position is compared this to the new measurements and then the a combination of the most likely parameters are used. Hence, this method increases the accuracy of the measurements by accounting for measurement errors.

In this research the acceleration in X and Z direction, speed in X and Z direction, pitch angle and pitch speed are used during the analysis. According to the data of the manufacturer the measurements are very precise. Also, the errors known are reasonably small which will lead to small errors in the estimation of the aircraft parameters. The extent of the influence of these errors is hard to estimate because of the EKF which accounts for errors itself. The best method to identify the accuracy of the system is to compare the results from this setup to a calibrated measurement device during a flight. Unfortunately, this was not possible due to the expensive equipment needed and the extra weight of this calibration system on the aircraft.

Table 2.4: Detailed sensor information

Device:	Component:	Bit:	Measurement:	Accuracy:			
				Measurement range:	Measurement resolution:	Measurement range:	Measurement resolution:
Accelerometer 1:	ICM-20689[23]	16	Accelerations in three axis	±2g	$6.1 \cdot 10^{-5}$g	±16g	$4.9 \cdot 10^{-4}$g
gyroscope 1:			Angular rates in three axis	±250°/sec	$3.8 \cdot 10^{-3}$°/sec	±2000°/sec	$3.1 \cdot 10^{-2}$°/sec
Accelerometer 2:	BMI055[24]	12	Accelerations in three axis	±2g	$9.8 \cdot 10^{-4}$g	±16g	$7.8 \cdot 10^{-3}$g
gyroscope 2:		16	Angular rates in three axis	±125°/sec	$1.9 \cdot 10^{-3}$°/sec	±2000°/sec	$3.1 \cdot 10^{-2}$°/sec
Magnetometer:	IST8310[25]	14	Magnetic field X and Y direction	±1600µT	0.2µT		
			Magnetic field Z direction	±2500µT	0.3µT		
Device:	Component:	Bit:	Measurement:	Measurement range:	Measurement resolution:	Measurement accuracy:	
Barometer:	MS5611[26]	24	Absolute pressure	10-1200 mbar	0.012-0.065 mbar	±1.5 mbar	
Temperature sensor:			Temperature	-40-+85°C	0.01°C	±0.8°C	
GPS/GLONASS:	Neo-M8N[27]		Velocity X, Y and Z direction	-	-	0.05m/s	
			Heading	-	-	0.3°	
			Horizontal position	-	-	2.5m	
Airspeed sensor:	MS4525DO[28]	14	Pressure difference	$7 \cdot 10^{3}$-$1 \cdot 10^{6}$Pa	63 Pa	±0.25%	

[23] https://product.tdk.com/info/en/documents/catalog_datasheet/imu/ICM-20689-v2.2-002.pdf
[24] https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmi055-ds000.pdf
[25] http://www.isentek.com/en/dlf.php?file=../ISENTEK/(201703-09)IST8310%20Datasheet%20v1.2_brief-105.09.20.pdf
[26] https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FMS561 1-01BA03%7FB3%7Fpdf%7FEnglish%7FENG_DS_MS5611-01BA03_B3.pdf%7FCAT-BLPS0036
[27] https://www.u-blox.com/en/docs/UBX-15031086
[28] https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FMS452 5DO%7FB2%7Fpdf%7FEnglish%7FENG_DS_MS4525DO_B2.pdf%7FCAT-BLPS0002

2.1.5 Software

Multiple kinds of software are used to perform the test flight,. The first is PX4, which is the flight control software installed on the flight computer installed in the aircraft. Second is the Q-Ground control, which is the software on the ground station for the control of the aircraft (in this case the ground station is a laptop). And third is a Lua script, which is developed by the University of Linköping to generate frequency sweeps and other maneuvers on the control surfaces of the aircraft.

PX4

The flight computer is equipped with a custom version of PX4, an opensource drone flight control software. PX4 has the ability to control the aircraft autonomously, to stabilize the aircraft and to give the pilot full direct control over the aircraft as well. During the flight, all the available sensor data is collected and it is possible to send data to the ground while testing.

A custom mixer file is created for the Skysurfer X8 to enable separate control of both engines and ailerons. These controls are not used during the flight test of this experiment, but with this it would be possible to perform a one-engine inoperative test. These doubled control systems make the aircraft more reliable, if one output is affected by an error, the other engine or aileron is not affected. This mixer file is included in Appendix B. More information on how these files are constructed can be found on the PX4 website[29].

Another modification is made for the logging of the data. Normally, the data is logged at a low rate, which is satisfactory for average use, but in this analysis the highest possible data logging rate is required. Therefore, the data logging is set to the system identification logging rate, which increases the logging rate. This modification is made in the parameters list in Q-ground control whereafter the setting is uploaded on the flight controller. The parameter which is changed is the: "SD_LOG_PROFILE"[30].

Q-groundcontrol

Q-groundcontrol is the software used on the ground station. This software can be used to set parameters, read parameters from the aircraft, set way points, set geofences, plan a mission, start a failsafe, etc. The software is used to assist the pilot. The co-pilot uses the software to give the pilot feedback on the speed of the aircraft and height. The co-pilot ensures that the reference speed or height is maintained for every flight test. The software assists to adhere to the regulations for drone flight, limits can be set for example: the height, distance from the pilot, speed, etc. The flight tests limits are set according to the regulations and the automatic return to home action would take over the flight when one of these limits is passed.

Figure 2.6 shows the screen of a mission in Q-ground control. In autonomous mode it is possible to set waypoints to plan a flight. On the top bar the status of the aircraft is shown, it shows a safety feature which tells if the engines are armed or disarmed. When they are armed, the engines and therefore the propellers can spin but when they are disarmed, they can not. Furthermore, the flight mode, battery level and the strength of the connections (GPS, RC and telemetry) are displayed in the top bar as well. In the right corner of the screen the measurement sensors are shown, these can be adjusted to show everything that is measured, for example: altitude, airspeed, pressure, etc. The attitude is shown by the gyroscope and compass displayed in the two circles above the displayed parameters.

[29] https://docs.px4.io/master/en/concept/mixing.html
[30] https://docs.px4.io/v1.9.0/en/advanced_config/parameter_reference.html
[31] https://docs.qgroundcontrol.com/master/en/index.html

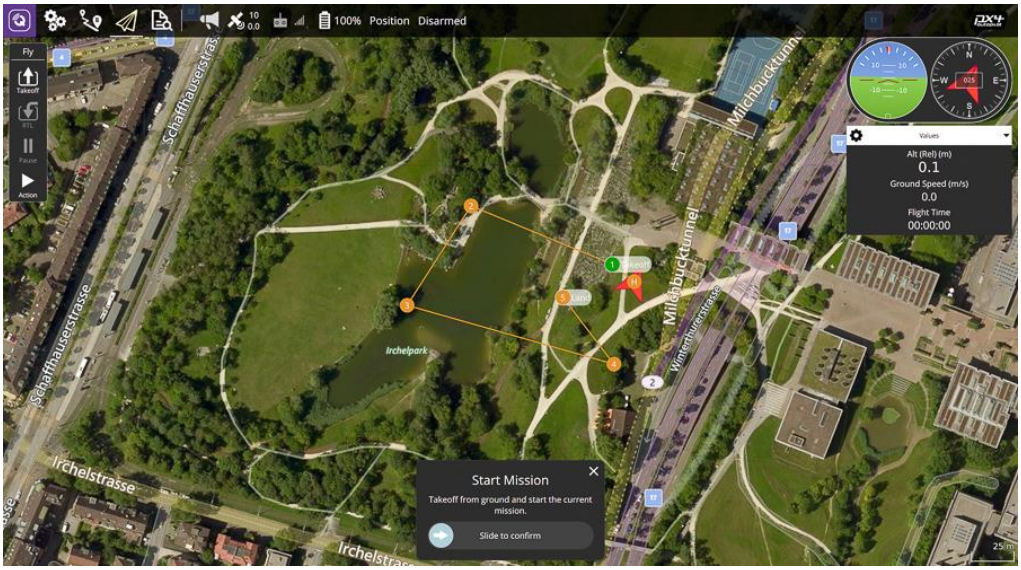


Figure 2.6: Q-ground control mission screen[31]

Lua

The Lua script is developed at the University of Linköping and used in their flight test programs [4]. This script is used during the flight tests to perform the frequency sweeps and doublets. This software is designed for the JETI transmitters, which are able to run LUA scripts. This program makes it is possible to control one control surface on the aircraft with a switch and a knob on the transmitter. The type of signal which is send to this control surface can be chosen, for example a step input, doublet or frequency sweep. To perform a frequency sweep, the switch should be activated to trigger the start of the maneuver and with the knob the frequency of the sweep can be modified manually. Before the flight, the amplitude, minimum frequency and maximum frequency of the sweep should be set.

Software versions and details

In Table 2.5 the software used is summarized, details on the version of the software and the location to get it is shown.

Table 2.5: Flight test software used

Software:	Version:	Source:
PX4	2.7.4	https://github.com/PX4
Q Ground Control	3.5.5	http://qgroundcontrol.com/#resources
Frequency sweep LUA script	Beta	-

2.2 Flight test and maneuver setup

The longitudinal dynamics of the aircraft are investigated with frequency sweeps and doublets are performed to validate the dynamic model extracted from the sweeps. With this method, the dynamics of the aircraft can be determined with minimal measurements and flight time. The inputs for this system are the elevator and the throttle, the outputs are the attitude, accelerations and speeds of the aircraft.

In this section the build-up of the flight test is explained. Firstly, the restrictions for the maneuvers that have to be performed to gain the best quality of response of the aircraft are summarized. Secondly, the setup of the test campaign is shown, which introduces the flights and checks that have to be performed to gain all the information needed for the analysis.

2.2.1 Frequency sweep

A frequency sweep (an example is shown in Figure 1.5) is a sinusoidal signal with a ranging frequency. These signals can be used as an input to extract the response of a dynamic system. By using this signal, the dynamics over the range of frequencies in the sweep can be determined. The frequency sweep is a powerful maneuver to maximize the information content with minimal time effort. The frequency sweep can be manually excited by the pilot, in this case the LUA program is used to generate the signal. The signal has a starting and end frequency, which are predefined for every test. The pilot can manually increase or decrease the frequency of the signal with a knob on the transmitter. Important aspects to keep in mind are shown below.

What is important:

- 3 seconds of trim before and after the frequency sweep
- start with two complete periods of the minimum frequency
- slowly increase the frequency
- keep the aircraft centered

What is not needed:

- constant input amplitude
- exact sinusoidal shape
- exact frequency progression
- exact repeatability
- increasing the amplitude at higher frequencies
- high frequencies and high amplitudes [32]

With these remarks in mind, the LUA program and stabilize mode are used to relieve the pilot, this way he only needs to focus on the altitude, speed and increasing the frequency. This has been proven to be already quite a high work load. The signal properties used for the frequency sweep are shown in Table 2.6.

Table 2.6: Frequency sweep parameters

Parameter:	Value:
Min. frequency	0.4 Hz
Max. frequency	5 Hz
Amplitude throttle	10%
Amplitude elevator	5 deg

The input signal to the servo is taken as the input signal for the analysis, it is important that the servo can keep up with the input signal. In Table 2.3, the speed of the servo is given. This indicates if the servo can follow the input signal. In one cycle the servo horn covers 20 degrees (5 degrees up, 10 degrees down and 5 degrees up), according to the manufactures' specifications it would take the servos 0.04 seconds to rotate this number of degrees. The minimum time of a signal is 0.2 seconds, therefore the servo should be able to keep up with the input signal. However, this holds true when the servo is unloaded, which is not the case. Nonetheless, there is still a factor of 5 in between these numbers, therefore it is expected that the servo is fast enough to keep up with the input signal. Preferably it should be checked if the servos can keep up to 5 Hz, by using servos with feedback.

2.2.2 Flight test

The flight test is divided into several parts. Firstly, the basic tests are performed to check if the system functions satisfactory. The first part of these tests are on the ground, also called the ground tests, and the second part consists of basic tests to verify the basic functions of the aircraft during flight. These tests are necessary to make sure it is safe to perform the actual flight test, as can be seen below in the breakdown of the tests. Secondly the actual flight test can be performed, this test consists of the flight maneuvers: Elevator and throttle sweeps and doublets.

The steps and checks that are performed are summarized in the flight-testcards, which are used to offer guidance to the team to structurally go through the tests without skipping important steps. The flight test cards, which are developed for this flight test, are included into Appendix C. All the cards have the same setup, to make it possible to go through them quickly and easily.

Flight test setup:

1. Basic tests
    - 1.1. Ground tests
        - 1.1.1. Initial setup
        - 1.1.2. Control deflection test
        - 1.1.3. Engine test
        - 1.1.4. Range test
    - 1.2. Basic flight tests
        - 1.2.1. Simple flight
        - 1.2.2. Return to home
2. Actual flight tests
    - 2.1. Elevator sweep and doublet
    - 2.2. Throttle sweep and doublet

Ground tests

The ground tests are all performed with the aircraft on the ground. For all components it is checked if they are fastened and working correctly. The temperature of critical components from the propulsion system are checked, to make sure the components would not overheat. These very specific tests were only performed once on the test day, the more general tests are included in the GO-NO-GO checklist which is performed before every flight.

The goal of the initial setup is to check if the aircraft is flightworthy. A long list of components and settings were checked in this part. All electrical components and bolts were fastened and secured, the safety switch was pressed to start the aircraft, the hatches were closed and secured, the center of gravity was checked, all calibrations were performed and lastly, the restrictions on the flight domain (geofence) were set.

The goal of the control deflection test is to check if the control surfaces and the transmitter are configured correctly. The pilot checks if the stick input of the transmitter corresponds to the deflections of the control surfaces. This check was performed multiple times during the tests, the pilot did this automatically before every flight and the check is included in the go-no-go list.

The engine test has as goal to test if the engine is working satisfactory and the components associated with it are not overheating. Therefore, the engine was set to maximum rpm for 30 seconds and half rpm for 3 minutes, this simulates a short flight. During the test the aircraft was fastened to make sure it stayed on the ground. While the flight was simulated the temperatures of the engines, electrical speed controllers and battery were checked.

The range test has as goal to verify the working of all the radio frequency connections at large distance. In this test, a distance of 1000 meter was created between the ground station and the aircraft. By doing so, the strength of the radio signal for the remote control and the telemetry from the ground station was checked. After this was proven to be satisfactory, the first flights were performed.

Basic flight tests

The primary goal of the simple flight was to check if the center of gravity of the aircraft is correct and the flight characteristics are satisfactory. Secondly, the stabilized function of the autopilot is tested. Thirdly, the energy consumption can be estimated, which was no concern in this aircraft setup. The flight was very straight forward: the plane took off, climbed to the desired altitude, rotated 180 degrees, flied straight again rotated and then the landing approach was started. When it is was not possible to land, the pilot could break off the landing and try again with a go-around.

The return to home-test has as goal to check if the return to home function and the autopilot functions correctly. The pilot engaged this function by changing the flight mode with a switch on the transmitter. When the switch is set to return to home, the autopilot takes over the plane and brings

the aircraft back to the home point. The home point can be set on the ground control station. In the case that the home point is not set, it will automatically take the take off point at 100 meter altitude as home point. The return to home test consisted of flying two rounds, then a straight flight away from the pilot, at a distance of 300 meter the return to home function was activated. This test is successful, if the aircraft returns to the home point safely in that case the autopilot is configured correctly. This ensures that the aircraft will be brought back safely when the aircraft has flown out of the geofence, which is set in the initial setup.

Actual flight tests

The actual flight test consisted of two types of maneuvers, which were the frequency sweeps and the doublet inputs. The frequency sweeps are used to gather the information on the behavior of the aircraft and the step doublets are used to verify the systems derived from the frequency sweeps of the aircraft. All four maneuvers (throttle sweep, elevator sweep and their corresponding doublets) had to be performed three times, in total there would be twelve passes needed to perform all the flights. With this aircraft all of these maneuvers can be combined in one flight and therefore all tests are stated on one flight-testcard.

The aircraft climbed to a height of 30m and was trimmed before starting the test maneuvers. The tests were performed with the LUA script, as is explained in section Software. The co-pilot guided the pilot to the height of 30 meter, speed of 18m/s and checked the trim conditions.

The tests were all performed following the Dutch drone regulations, which limited the flight tests in some ways. The maximum distance allowed from the pilot to the aircraft is 500 meter, which is the maximum line of sight. The altitude is limited as well, but in these tests the maximum height is not reached and therefore this restriction was not limiting. There are regulations on the transmitting of radio frequencies as well, to ensure that other important signals for example telecommunication services etc. will not be influenced.

2.3 System identification

The goal of system identification is to find the unknown parameters of a system from the input and output data. In this case the inputs are the elevator angle deflection and the throttle setting, the outputs are the forward speed, downward speed and the pitch angle including the derivatives of these parameters. The shape of the mathematical systems, which are several transfer functions describing SISO systems and a state space system describing a MIMO system, are known from flight dynamics. In a SISO system one input-output relation is investigated, for example the elevator input and pitch angle output for the pitch angle relative to elevator deflection response.

The flight test data has to be converted into a format which is ready for the conversion to frequency domain, whereafter this input-output data (SISO) is converted to frequency domain. This results into a bode plot for every input-output relation. These SISO systems can be represented with a transfer function. The transfer function can be extracted by an optimization which searches the best fit of the transfer function on the bode plot, which is generated from the flight test data. With this data, the handling characteristics can be estimated, as the phugoid and short period motions are present in the transfer functions, the CAP can be estimated from parameters in the transfer function and the bandwidth is estimated from the bode plot of the pitch response relative to elevator deflection.

When the SISO responses are combined, it is possible to fit a state space matrix on these responses. All responses are used to estimate the state space system that fits best to the flight test data. The state space system contains the information to estimate the phugoid, short period and CAP.

2.3.1 Data processing

The data processing consists of two parts. The first part is converting the flight test data to a dataset which can be imported into CIFER. The second part is importing the data in CIFER and filtering the data which is performed in Frequency RESPonse IDentification (FRESPID). The data from the flight test (in a Ulog file[32]) is saved on a SD-Card in the flight controller. This is converted to a MATLAB file

[32] https://docs.px4.io/master/en/log/flight_log_analysis.html

by Pyulog[33] and Matulog[34]. The log files contain data which is saved at different sampling rates in different folders. MATLAB scripts are made to do three things: To convert the data into one file containing the relevant data at the same sampling rate, to cut the data into pieces to extract the maneuvers and to make different files for every maneuver, which can be imported in CIFER. The data is further filtered in FRESPID, where the MATLAB files containing the separate maneuvers are imported. The program is set to delete drift, remove the mean of the signal, and resample the signal [32].

Pre-processing: MATLAB

In the MATLAB function preprocess are the parameters of interest copied from the logfile and resampled. The in preprocess generated data is cut into the desired fragments to extract the maneuvers by the sys_id program. In fre_input, the fragments of flight data are structured and saved in a .mat file, ready for processing in CIFER. These three MATLAB programs are included in Appendix D. The flight test parameters of interest from the log-file are shown in Table 2.7. Table 2.8 shows the structure of the transformed data, which can be imported into CIFER.

Table 2.7: Location retrieved data from log files

Parameter	Location:	Unit:
Aileron deflection	log.data.actuator_outputs_0.output_1_	PWM
Rudder deflection	log.data.actuator_outputs_0.output_4_	PWM
Elevator deflection	log.data.actuator_outputs_0.output_2_	PWM
Throttle setting	log.data.actuator_outputs_0.output_3_	PWM
Time stamp inputs	log.data.actuator_outputs_0.timestamp	µs
Quaternation 0	log.data.vehicle_attitude_0.q_0_	-
Quaternation 1	log.data.vehicle_attitude_0.q_1_	-
Quaternation 2	log.data.vehicle_attitude_0.q_2_	-
Quaternation 3	log.data.vehicle_attitude_0.q_3_	-
Time quaternations	log.data.vehicle_attitude_0.timestamp	µs
Speed North	log.data.estimator_status_0.states_4_	m/s
Speed East	log.data.estimator_status_0.states_5_	m/s
Speed down	log.data.estimator_status_0.states_6_	m/s
Time speed	log.data.estimator_status_0.timestamp	µs
Acceleration x	log.data.vehicle_local_position_0.ax	m/s^2
Acceleration y	log.data.vehicle_local_position_0.ay	m/s^2
Acceleration z	log.data.vehicle_local_position_0.az	m/s^2
Time acceleration	log.data.vehicle_local_position_0.timestamp	µs
Acceleration pitch	log.data.vehicle_attitude_0.pitchspeed	rad/s
Acceleration yaw	log.data.vehicle_attitude_0.yawspeed	rad/s
Acceleration roll	log.data.vehicle_attitude_0.rollspeed	rad/s
Height	log.data.vehicle_local_position_0.z	m
Time height	log.data.vehicle_local_position_0.timestamp	µs
True airspeed	log.data.airspeed_0.true_airspeed_m_s	m/s
Temperature	log.data.airspeed_0.air_temperature_celsius	°C
Time airspeed	log.data.airspeed_0.timestamp	µs
Air density	log.data.vehicle_air_data_0.rho	kg/m^3
Air pressure	log.data.vehicle_air_data_0.baro_pressure_pa	Pa
Time air data	log.data.vehicle_air_data_0.timestamp	µs

[33] https://github.com/PX4/pyulog
[34] https://github.com/CarlOlsson/matulog

Table 2.8: Structure of: Elevator_sweep_skysurfer.mat, Elevator_sweep_skysurfer2.mat, Elevator_sweep_skysurfer3.mat and Throttle_sweep_skysurfer.mat

Name:	Variable:	Unit:
/elevator	Elevator deflection	deg
/throttle	Throttle setting	%
/pit	Pitch angle	deg
/q	Pitch speed	deg/s
/u	Speed in X direction	m/s
/w	Speed in Z direction	m/s
/ax	Acceleration in X direction	m/s^2
/az	Acceleration in Z direction	m/s^2
/time	Time	s

2.3.2 Frequency analysis

The frequency response is investigated in CIFER by FRESPID. In FRESPID, the time domain data is transferred to frequency domain by a Fast Fourier Transform (FFT) with the Chirp-Z Transform (CZT). FRESPID imports and conjugates the data to start the frequency analysis and subsequently removes the average value and the linear drift from this data. Firstly, the drift and the constant average of the inputs and outputs in the time domain for the different sweeps are removed. Those different sweeps are conjugated to one signal per input and output and those inputs and outputs are linked together. [32] All of this is shown in the left side of Figure 2.9.

The prepared data is subsequently run through "a digital filter to eliminate potential aliasing of high frequency noise by the chirp z-transform. Digital filtering is also used as a precursor to data decimation which improves the computational speed of the CZT" [32]. Then, the overlapped windowing method is used, which is a common method to reduce the random error in the spectral estimates. This method cuts the signals into segments of T_{win} with an overlap. There is an overlap of 50% to make sure minimal data is lost because of the half sine method [41] which reduces the data at the beginning and end of the window. Figure 2.7 shows three segments with no overlap and Figure 2.8 shows segments with a 50% overlap showing less data loss. The input and outputs are cut into the segments and multiplied with window shaping functions, which are in this case the half sine waves. These segments are then converted to frequency domain with a chirp z- algorithm resulting in rough spectral density functions and are averaged to obtain smooth spectral densities. Lastly, the frequency response and coherence functions are generated for the given window size(s) by repeating this process in the case of multiple window sizes. [32]

The bode plot shows the magnitude [dB] and the phase [deg] versus the semi log scale of the frequency in [rad/s] or [Hz] of the response. The coherence function is added to these plots because it shows the accuracy of the frequency response. This function shows if the system is well represented over the range of frequencies and if the system can be modelled as a linear system in the frequency domain. The value of the coherence ranges from 0-1, in which 1 indicates the best and 0 indicates the worst possible representation of the results. The value of the coherence should be above 0.6 for acceptable results. Large oscillations in the coherence indicate that the results are not trustworthy. [32]

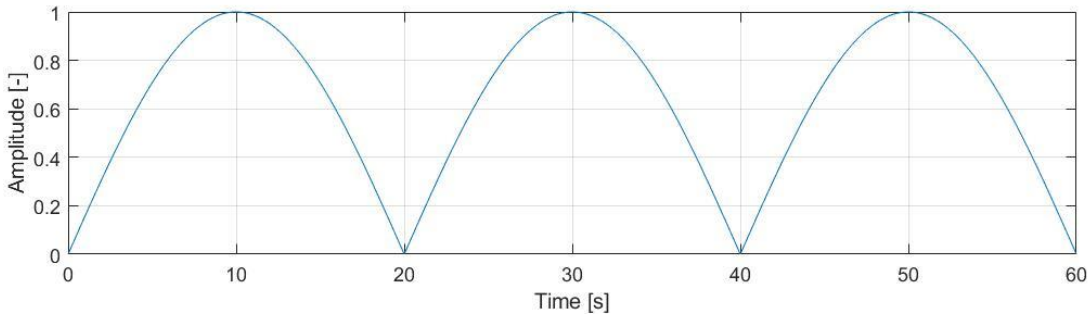


Figure 2.7: Half sine wave with no overlap with three segments

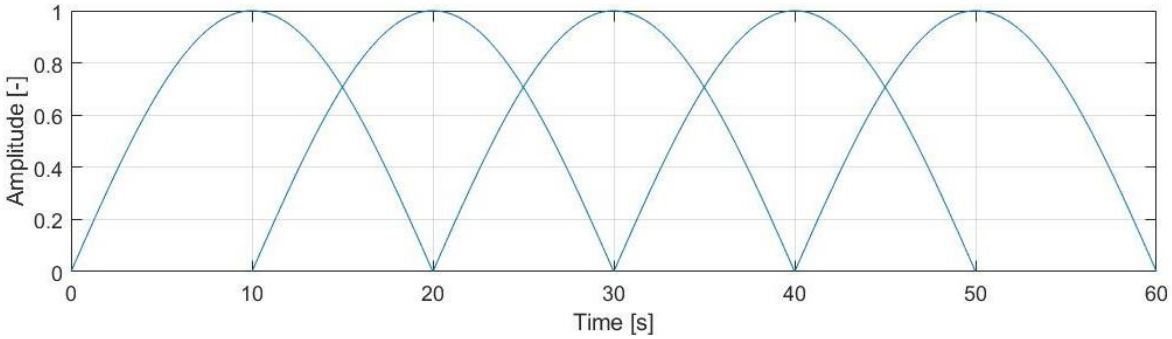


Figure 2.8: Half sine wave with 50% overlap with 6 segments

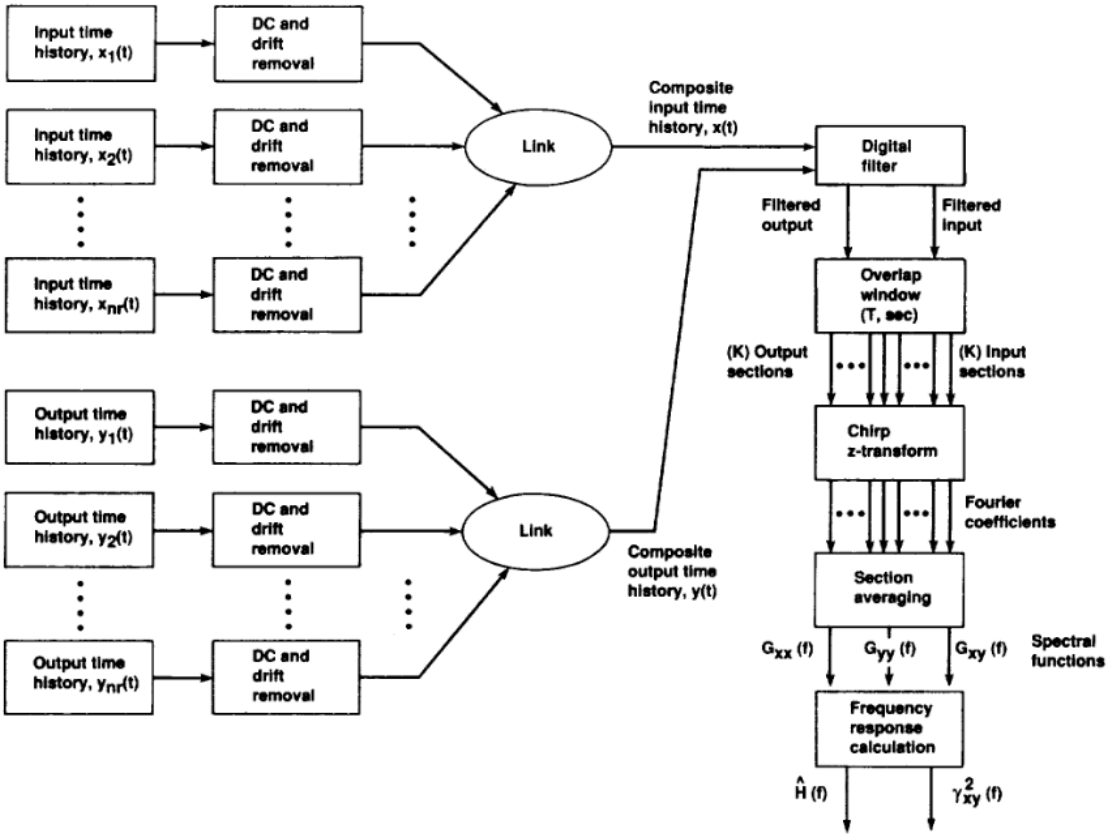


Figure 2.9: FRESPID work flow [42]

The frequency domain results are calculated in FRESPID for both the elevator and thrust sweep and these analyses are named: LOSWPELE (LOngitudinal SWeeP ELEvator) and LOSWPTHR (LOngitudinal SWeeP THRottle). The maneuver files which are generated in MATLAB are loaded into FRESPID, the inputs and outputs are defined according to Table 2.9. The sample rate is increased to 50 Hz to increase the points for the frequency analysis.

The -3 bandwidth frequency is the frequency at which the low pass filter is switched on. This value is chosen to be 20 Hz, which is four times the maximum frequency of the input signal. The sample rate is increased from 10 to 50 Hz and the T_{win}, the size of the window, is set to 10 Hz, according to the guideline [32]:

$$T_{win} = 2 \cdot T_{\max} \tag{2}$$

Different window sizes are examined in Appendix E.

Table 2.9: FRESPID settings

	Case 1:	Case 2:
Case name:	LOSWPELE	LOSWPTHR
Inputs:	Ele and Thr	Ele and Thr
Outputs:	U,W, Ax, Ay, Thet and q	U, W, Ax, Ay, V,.Thet and q
Filename:	Elevator_sweep_skysurfer.mat, Elevator_sweep_skysurfer2.mat , Elevator_sweep_skysurfer3.mat	Throttle_sweep_skysurfer.mat
-3DB bandwidth frequency:	20	20
Desired sample rate:	50	50
TWIN:	10	10

The frequency responses are visualized in bode plots, these contain the magnitude, phase and coherence in respect to the frequency. These plots show how the magnitude and phase of the input signal is changed by the system compared to the output signal. The coherence shows how well the input signal is correlated to the output and therefore, how reliable the results are and how well the system is estimated as an LTI-system.

CIFER contains two other programs to improve the frequency responses, however for these results these two programs did not have a major positive effect on the frequency responses, this analysis is included in Appendix F.

2.3.3 Bandwidth criterion

The Bandwidth criterion can be determined from the frequency response of the pitch angle to the elevator deflection. The criterion consists of the bandwidth frequency and the phase delay, these two parameters show in which level the criterion can be placed. In CIFER there is a special module to investigate this criterion, this module can find the right parameters to calculate the bandwidth frequency and phase delay.

The attitude response can be generated from pitch to the elevator response or $\frac{1}{s} \cdot \frac{q(s)}{\delta_e}$. The latter is often used because the pitch speed has a higher accuracy compared to the pitch, as it is directly measured by the accelerometers.

The bandwidth (ω_{BW}) is defined as the lowest value of either the 135 degree phase $(\omega_{135} = \omega_{WB_{phase}})$ or the 6-dB gain margin frequency $(\omega_{WB_{gain}})$. In Figure 2.10, the 6-dB gain margin can be found by adding 6 dB to the -180 degree phase frequency on the magnitude plot and the 135deg frequency is

found in the phase plot at the magnitude of -135 db. The phase delay (τ_p) [43] is calculated according to:

$$\tau_p = \frac{\varphi_{2\omega_{180}} + 180}{57.3 \cdot 2 \cdot \omega_{180}} \tag{3}$$

This consists of the frequency at the phase of -180 degrees (ω_{180}) and phase at twice the -180 degrees frequency $(\varphi_{2\omega_{180}})$ [32, 43]. First, the frequency at -180 degrees phase is determined in the plot and then this frequency is doubled to find the phase belonging the doubled frequency (Figure 2.10).

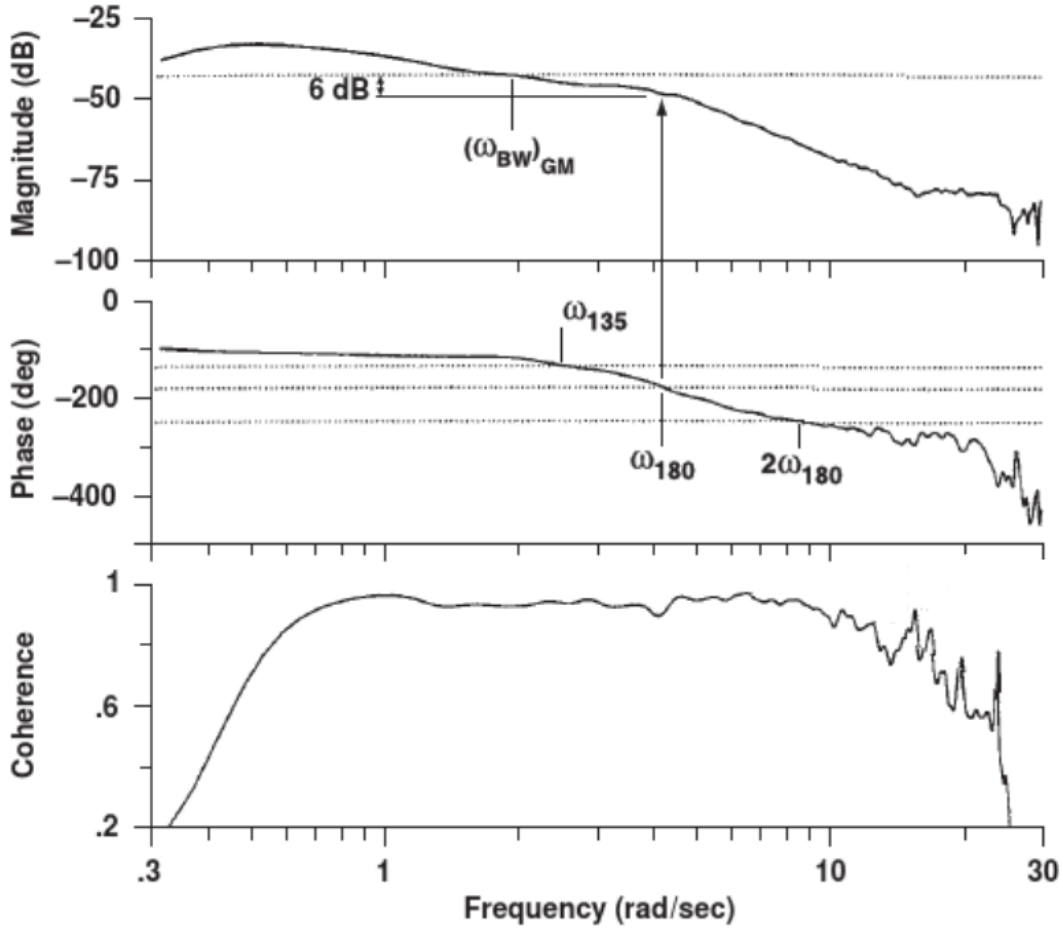


Figure 2.10: Bandwidth criterion [32]

2.3.4 CAP, short period and phugoid

The CAP, short period and phugoid are extracted from the pitch or pitch speed response as well. These results can not directly be seen in the bode plot, instead a transfer function has to be fitted on the measured response. The CAP can be derived with the response of the normal acceleration and pitch angle as well.

For the fit of the transfer function on the frequency response, the transfer function identification (NAVFIT) package in CIFER is used. The model structure of the transfer function can be defined (gain, number of poles, number of zeros and equivalent time delay) and the Rosenbrock's multi variable search method [44] is used to find the best fit. The accuracy of the fit is defined by the cost function (J). A cost function of below 100 reflects an acceptable accuracy and a cost function of below 50 a model which is nearly indistinguishable from the test signal [29]. These transfer functions are very valuable to estimate parameters on handling characteristics.

The pitch response is represented by this transfer function [26, 27]:

$$\frac{q(s)}{\delta_e} = \frac{K_\theta s\left(s+\frac{1}{T_{\theta_1}}\right)\left(s+\frac{1}{T_{\theta_2}}\right)e^{-\tau_e s}}{\left[\varsigma_p, \omega_p\right]\left[\varsigma_{sp}, \omega_{sp}\right]} \tag{4}$$

The denominator of equation 4 is a short hand notation for:

$$\left[\varsigma_p, \omega_p\right]\left[\varsigma_{sp}, \omega_{sp}\right] = \left[s^2 + 2\varsigma_p\omega_p s + {\omega_p}^2\right]\left[s^2 + 2\varsigma_{sp}\omega_{sp} s + {\omega_{sp}}^2\right] \tag{5}$$

The normal accelerometer response is represented by this transfer function [32]:

$$\frac{n_z(s)}{\delta_e} = \frac{K_n s\left(s+\frac{1}{T_{h_1}}\right)e^{-\tau_n s}}{\left[\varsigma_p, \omega_p\right]\left[\varsigma_{sp}, \omega_{sp}\right]} \tag{6}$$

The transfer functions contain both the phugoid and short period motion in the denominator, the short hand notation shows the damping $(\varsigma_p \,\&\, \varsigma_{sp})$ and frequency $(\omega_p \,\&\, \omega_{sp})$. $K_\theta \,\&\, K_n$ are the gains, T_{θ_1}, T_{θ_2} and T_{h_1} are the time constants [44], τ_e and τ_n are the equivalent time delays. It is also possible to estimate the CAP with a set of simpler transfer functions, in which case the phugoid will be excluded from the equations. The parameters can be estimated by simpler transfer functions as well, which are called Lower Order Equivalent Systems (LOES), but this method is of course less precise compared to the full system. The pitch response is estimated by this transfer function for the LOES [32]:

$$q(s) = \frac{K_\theta\left(1+T_{\theta_2}s\right)e^{-\tau_n s}}{\left[\varsigma_{sp}, \omega_{sp}\right]} \tag{7}$$

The accelerometer response in vertical direction is estimated by this transfer function for the LOES [32]:

$$\frac{N_z(s)}{\delta_e} = \frac{N_{z_{\delta_e}}e^{-\tau_n s}}{\left[\varsigma_{sp}, \omega_{sp}\right]} \tag{8}$$

The CAP can be estimated from the short period frequency, speed, gravity and incidence lag constant [26, 27]:

$$CAP = \frac{{\omega_{sp}}^2}{\frac{n}{\alpha}} \approx \frac{{\omega_{sp}}^2}{\frac{V}{g}\cdot\frac{1}{T_{\theta_2}}} \tag{9}$$

The second method to extract the CAP uses the short period frequency, pitch rate gain and normal accelerometer gain [32]:

$$CAP = \frac{\omega_{sp}^{\ 2}}{\frac{N_{z_{\delta_e}}}{K_\theta}} \tag{10}$$

2.3.5 State space representation

From the combined set of frequency responses, a state space system of the aircraft's dynamics can be estimated, this is called a Multi Input Multi Output (MIMO) model identification. The state space system and the variables are predefined and with a least squares method the best fit for this system on the frequency response is investigated. The state space system is assumed to be the linearized longitudinal equations of motion. The derivation of the state space system is explained in "Flight Dynamics Principles: A linear Systems Approach to Aircraft Stability and Control" [45], the system is shown here:

$$\begin{bmatrix} m & 0 & 0 & 0 \\ 0 & m - Z_{\dot{w}} & 0 & 0 \\ 0 & -M_{\dot{w}} & I_{yy} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} X_u & X_w & X_q & -g\cos(\theta) \\ Z_u & Z_w & Z_q & -g\sin(\theta) \\ M_u & M_w & M_q & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} X_{\delta_e} & X_{\delta_t} \\ Z_{\delta_e} & 0 \\ M_{\delta_e} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_t \end{bmatrix} \tag{11}$$

Often the angle of attack instead of the velocity in Z-direction, is in the set of equations. But in these, the latter is used because the velocity in Z direction is measured directly by the flight controller while the angle of attack can only be estimated from the measurements. The structure of the system which is investigated is:

$$M\dot{x} = Fx + Gu(t - \tau) \tag{12}$$

$$y = H_0 x + H_1 \dot{x} \tag{13}$$

The inputs for analysis of the system are the state vector (x), input vector (u), measurement vector (y) and state space matrices: M, F, G, and H:

$$x = \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} \tag{14}$$

$$u = \begin{bmatrix} \delta_e \\ \delta_t \end{bmatrix} \tag{15}$$

$$y = \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} \tag{16}$$

$$M = \begin{bmatrix} m & 0 & 0 & 0 \\ 0 & m - Z_{\dot{w}} & 0 & 0 \\ 0 & -M_{\dot{w}} & I_{yy} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

$$F = \begin{bmatrix} X_u & X_w & X_q & -g\cos(\theta) \\ Z_u & Z_w & Z_q & -g\sin(\theta) \\ M_u & M_w & M_q & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{18}$$

$$G = \begin{bmatrix} X_{\delta_e} & X_{\delta_t} \\ Z_{\delta_e} & 0 \\ M_{\delta_e} & 0 \\ 0 & 0 \end{bmatrix} \tag{19}$$

$$H_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{20}$$

$$H_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{21}$$

To investigate the system of equations which belongs to the Skysurfer X8, the MIMO state space model identification (DERIVID) software module in CIFER is used. This program is will search for the optimal state space system that fits the flight test data. This is achieved with a least squares optimization method searching for the best fit of the state space system in the responses of the flight test.

For all parameters identified in the analysis two measures of accuracy are calculated, these are the Cramér-Rao bound and the insensitivity value. The Cramér-Rao bound is used as an estimator of the standard deviation in the error of the estimated parameter [46]. "A large bound means that the computed parameter value is not estimated with good confidence. This provides an indication of poor data quality or improper model formulation. " [47] "The parameter insensitivity provides information on the effect of a converged parameter on the final cost function. Parameter insensitivity is high when changes in the value of a parameter do not lead to a significant change in the cost function. " [47]

Figure 2.11 shows how the program works. First, the state space system which should be estimated, equations 14-21, is put into the program. Subsequently, the frequency responses derived in FRESPID are loaded and parameters in the state space system are set free or fixed. In this case, all variables and parameters named $X_{\dots}$, $Z_{\dots}$, $M_{\dots}$, m and I_{yy} are free, the other parameters in the M, F and G matrix are fixed. The optimization is started and takes several iterations. At last, the results are shown and the accuracy of these results is calculated. The user can decide if the estimation stops there or that it is necessary to delete a parameter of the state space system because the accuracy is not acceptable. If the latter is chosen, the process starts over again.

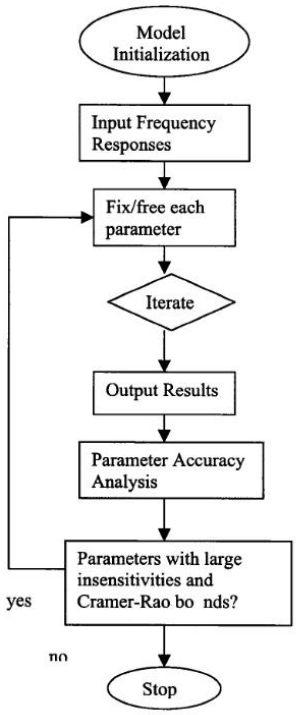


Figure 2.11: DERIVID work flow [47]

Not only the state space system is derived but also the separate responses for the pitch angle and pitch speed which are already known, as they are shown in equation 4. The response for the forward speed is:

$$\frac{u(s)}{\delta_e} = \frac{k_u\left(1-T_{u_1}\right)\left(1-T_{u_2}\right)}{\left[\varsigma_p, \omega_p\right]\left[\varsigma_{sp}, \omega_{sp}\right]} \tag{22}$$

And for the downward speed:

$$\frac{w(s)}{\delta_e} = \frac{k_w\left(1-T_{w_1}\right)}{\left[\varsigma_p, \omega_p\right]\left[\varsigma_{sp}, \omega_{sp}\right]} \tag{23}$$

2.3.6 Repeatability of the data check

The random error in the measurements can be estimated by performing a measurement multiple times and comparing the outcome. In this case it is not possible to compare the time domain signals for different sweeps because they differ in frequency build up due to the manual execution of the pilot. Therefore, the responses are converted to frequency domain and the bode plots are compared to each other.

2.3.7 Consistency of the data check

The consistency of the measurements can be checked in the frequency domain. In the frequency domain the derivative of a transfer function can be calculated by multiplying with s. This can be seen in the transfer function of the pitch and pitch speed response, as the factor in between them is s. This can also be executed for more parameters such as the forward speed and the forward acceleration and downward speed and downward acceleration.

2.4 Simulation

The results of the flight test can be compared to the results of a simulation which is performed in XFLR5. These simulations are for Quasi steady aerodynamics, which can be assumed if the reduced frequency is between 0 and 0.05. This depends on the air flow speed, the characteristic length of the airfoil and the maximum speed of the control surface during the sweep. The reduced frequency for this case is 0.011. The reduced frequency can be calculated with this equation[35]:

$$k = \frac{\omega \cdot b}{V} \tag{24}$$

A simulation in XFLR5 is chosen because it can quicky estimate the dynamics of a model. The software is designed for glider aircraft and therefore the throttle is not taken into account. This means that the state space function is slightly less complicated, as the thrust terms fall out of the equations. Because of these simplifications, there will be a difference in the results from the simulation and flight test.

Geometrical setup

In the simulation, the weight, center of gravity and moment of inertias of the aircraft have to be given. The weight of the aircraft is known, the center of gravity and moments of inertia are estimated. XFLR5 has a module to estimate these values from point masses and therefore, the aircraft is divided into different parts and the weight of each part is measured.

Firstly, a CAD model of the aircraft is created in Inventor[36] (Figure 2.12). In this program, all major parts of the aircraft are drawn and all separate parts are assigned a weight. In the end, the program will calculate the center of gravity of the aircraft which should be around 2 cm aft of the leading edge of the wing. This is where the plane would stabilize if this would be manually tested. The parts, masses and CG locations are shown in Table 2.10. The X-direction is along the fuselage, the Y-direction along the wing and the Z-direction is upwards.

Simplifications are made in this process: The wires in the fuselage are taken into the weight of the fuselage and therefore it is assumed that the weight of the cables is distributed over the whole fuselage. The weight of the servo mounts is added to the weight servos and are not separately calculated. All tape, control horns and other small components are not taken into account separately but are also distributed over the fuselage and wings. The airfoils of the aircraft are estimated in four-digit NACA airfoils, as shown in Table 2.2.

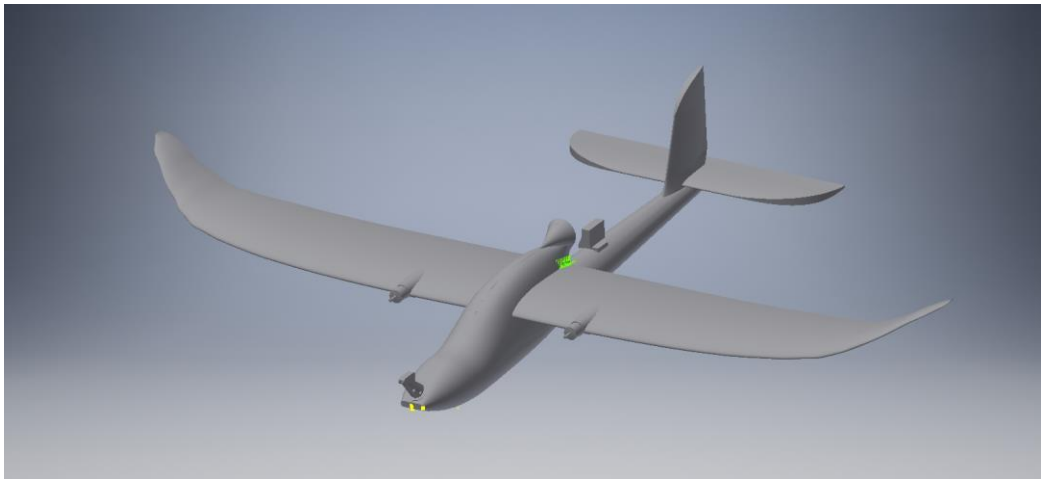


Figure 2.12: CAD model of the aircraft

[35] https://en.wikipedia.org/wiki/Reduced_frequency
[36] https://www.autodesk.nl/products/inventor/overview

Table 2.10: Components of the aircraft for center of gravity estimation

Part:	Weight [kg]:	CG X [mm]:	CG Y [mm]:	CG Z [mm]:
Fuselage	0.235	410	0	63
Wing L	0.085	330	292	66
Wing R	0.085	330	-292	66
Elevator	0.025	861	0	15
Rudder	0.010	834	0	86
Motor mount R	0.006	258	155	63
Motor mount L	0.006	258	-155	63
Motor R	0.026	225	155	63
Motor L	0.026	225	-155	63
ESC R	0.010	276	155	47
ESC L	0.010	276	-155	47
Telemetry mount	0.005	554	7	47
Telemetry	0.034	547	3	60
Servo	0.015	331	27	11
Servo	0.015	331	-27	11
Servo aileron R	0.015	331	347	63
Servo aileron L	0.015	331	-347	63
Battery	0.262	110	0	-12
BEC	0.015	252	0	-29
GPS	0.033	180	0	42
Power module	0.025	254	-10	-25
Receiver	0.023	375	0	11
Pitot assembly	0.020	3	0	41
Flight computer	0.033	210	0	0
Total	1.034	291	0	38

The center of gravity is estimated manually in the CAD program and with the estimation in XFLR5 as shown in Table 2.11.

Table 2.11: Center of gravity estimations

Direction:	Manually:	CAD:	XFLR5:
X	290±10 mm	291 mm	291 mm
Y	-	0 mm	0 mm
Z	-	38 mm	38 mm

Flight condition/simulation setup

The simulation consists of two parts: The wing analysis and the stability analysis. The wing analysis investigates the behavior of the airfoils used at different angles of attack and flow conditions. The stability analysis investigates the stability of the whole system, this will output the state space system of equations. The Guidelines for XFLR5 are followed to perform the analysis.

The wing analysis is performed for the NACA4410, NACA0013 and NACA0010 airfoils. The angle of attack is ranged from -6 till 10 degrees with an increment of 0.01 degrees. Reynolds number is ranged from 30 000 till 150 000 with increment of 20 000, and ranged 200 000 till 500 000 with an increment of 50 000. The Mach number is set to 0.05 and the transition is set to the trailing edge of the airfoil. The flight conditions for the wing and stability analyses are summarized in Table 2.12.

Table 2.12: Average flight condition of the flight test

Condition:	Value:	Dimension:
Temperature	17.2	°C
Airspeed	18	m/s
Air density	1.21	kg/m^3
Pressure	$1.01 \cdot 10^5$	Pa
Dynamic viscosity	$18 \cdot 10^{-6}$	Pa/s
Characteristic length	0.195	m
Speed of sound	342	m/s
Mach	0.05	-
Minimum test frequency	0.4	Hz
Maximum test frequency	5	Hz
Max deflection	5	deg
Maximum circular frequency	2.09	rad/s
Reynolds	$1.4 \cdot 10^5$	-
Froude	13	-
Reduced frequency	0.011	-
Elevator trim	4	deg
Throttle trim	66.5	%

The state space system that will be estimated has a slightly different setup then the one investigated in for the flight test results. The state space system that will be estimated is shown here:

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} X_u & X_w & X_q & -g\cos(\theta) \\ Z_u & Z_w & Z_q & -g\sin(\theta) \\ M_u & M_w & M_q & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} X_{\delta_e} \\ Z_{\delta_e} \\ M_{\delta_e} \\ 0 \end{bmatrix} \delta_e \tag{25}$$

3 Verification

Much of research is dedicated to investigating the dynamics of an aircraft in the frequency domain, for example the: Novion[37] [48-50], Multiplex Twinstar [38-40], Ultra stick 120 [51-53], Cessna scale model [36], other UAVs [31, 34, 54, 55] and XV-15 [35, 37]. Of which the Novion and XV-15 are aircraft and the other are model aircraft. Other research data can be used to verify the method proposed in this thesis. The research data on the handling characteristics of an aircraft or an Unmanned Aerial Vehicle, which also provides the state space model for the longitudinal dynamics, would be preferred.

It was attempted to acquire frequency sweep data from publicly available resources to verify the results. Unfortunately, there was no frequency sweep data found for the longitudinal direction but only frequency sweep data on the lateral direction, for example of the XV-15 [32, 35, 37], a vertical lift off vehicle which has rotating propellers.

The second best option would be to investigate a state space matrix of an aircraft, compute the time response of that system to a frequency sweep and use that data to do the verification. There are more papers who describe linearized state space matrices of the dynamics of the aircraft themselves, than ones who include the data of the frequency sweeps to derive them. Nonetheless, it is difficult to find a paper with a complete state space matrix for the longitudinal directions which include the handling criteria (CAP and bandwidth). Often parts of the state space matrices are deleted, due to unreliable measurements (with a low coherence) and the paper stops at the point that the state space system is found and therefore the handling criteria are not further investigated.

In the publicly available thesis from the Ryerson University about the Short Take Off and Landing (STOL) aircraft [56], the state system of the aircraft is derived from an aerodynamic and flight simulation and it contains the full decoupled longitudinal state space representation. The aircraft data is derived from an example in this book [57].This case is chosen for the verification of the method described in this thesis because it contains the complete decoupled dynamics of the longitudinal direction. Unfortunately, herewith the CAP and bandwidth can not be checked, but the short period and phugoid will be. The basic parameters of this aircraft are shown in
Table 3.1 and a top view of the aircraft in Figure 3.1.

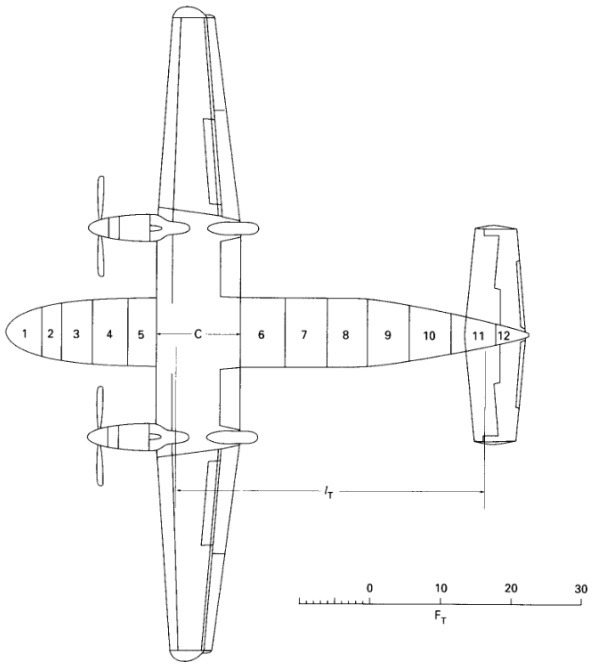


Figure 3.1: STOL aircraft [57]

[37] http://www.flightlevelengineering.com/assets/training_material/ryannavionlongitudinal.pdf

Table 3.1: STOL aircraft basic dimensions

Dimension:	Measurement:	Unit:
Wing span:	29	m
Horizontal tail span:	10	m
Lift coefficient:	0.77	-
Fuselage length:	23	m
Fuselage diameter:	2.9	m
Flying weight:	18 000	kg
Speed	66	m/s
Air density	1.22	kg/m^3

The state space system is stated in this configuration:

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} X_u & X_w & X_q & -g\cos(\theta) \\ Z_u & Z_w & Z_q & -g\sin(\theta) \\ M_u & M_w & M_q & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} X_{\delta_e} & X_{\delta_T} \\ Z_{\delta_e} & Z_{\delta_T} \\ M_{\delta_e} & M_{\delta_T} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_T \end{bmatrix} \tag{26}$$

3.1 Results

The input and output signals for the analysis are calculated in a MATLAB program. This program is included in Appendix G. The program generates an input frequency sweep for the elevator and throttle. This is a sine sweep with first two periods of the minimum frequency of 0.05 Hz and then the frequency is increased to the maximum frequency of 10 Hz. Figure 3.2 & Figure 3.3 show the input signals used for the verification. From these inputs the program generates the output signals with the known state space representation. These output signals are shown in Figure 3.4-Figure 3.7.

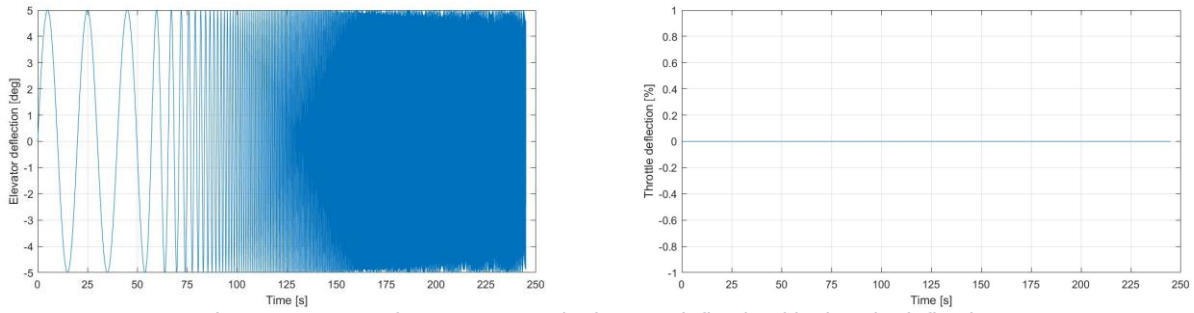


Figure 3.2: Input elevator sweep a) Elevator deflection b) Throttle deflection

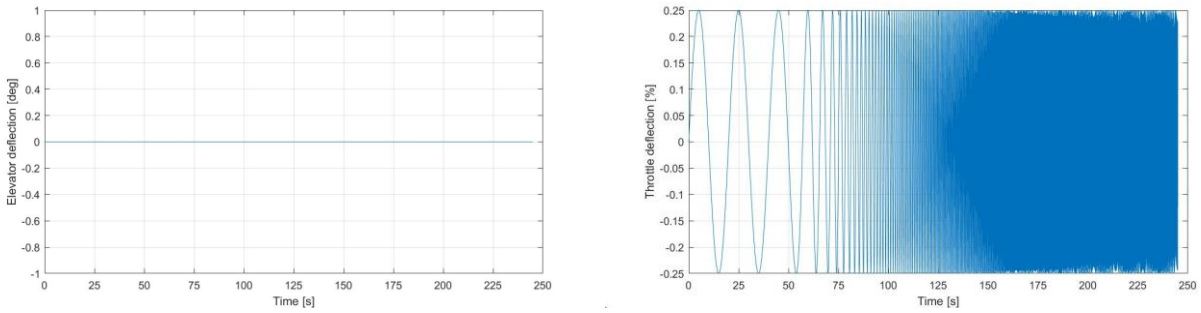


Figure 3.3: Input throttle sweep a) Elevator deflection b) Throttle deflection

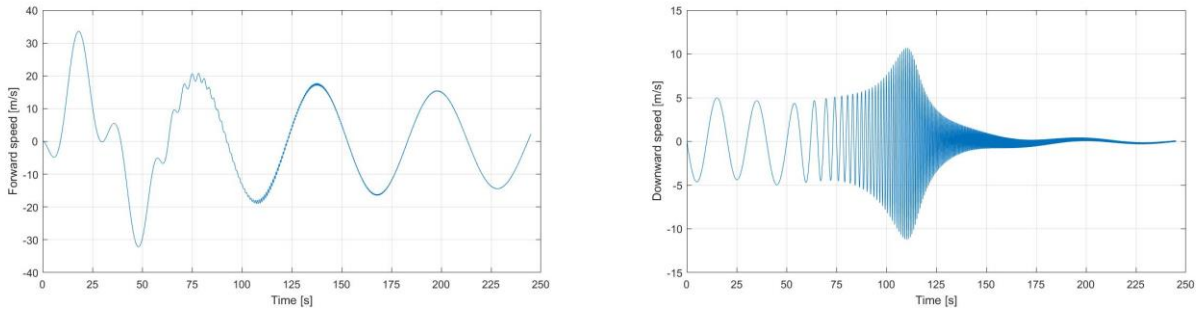


Figure 3.4: Output elevator sweep a) Forward speed b) Downward speed

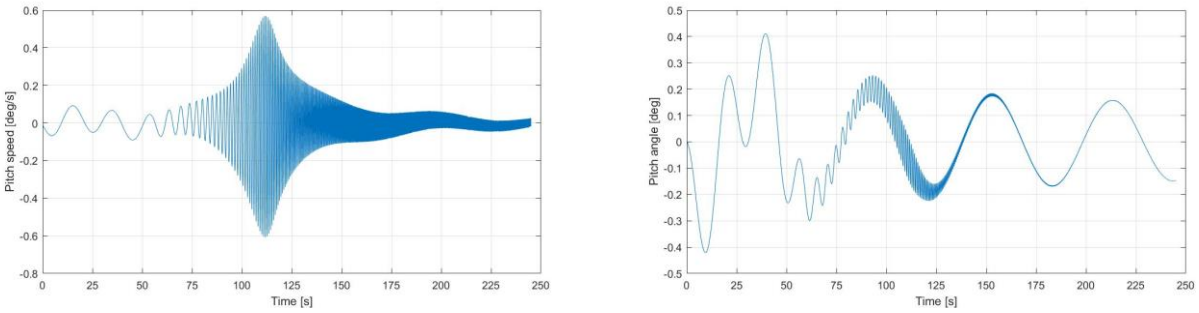


Figure 3.5: Output elevator sweep a) Pitch speed b) Pitch angle

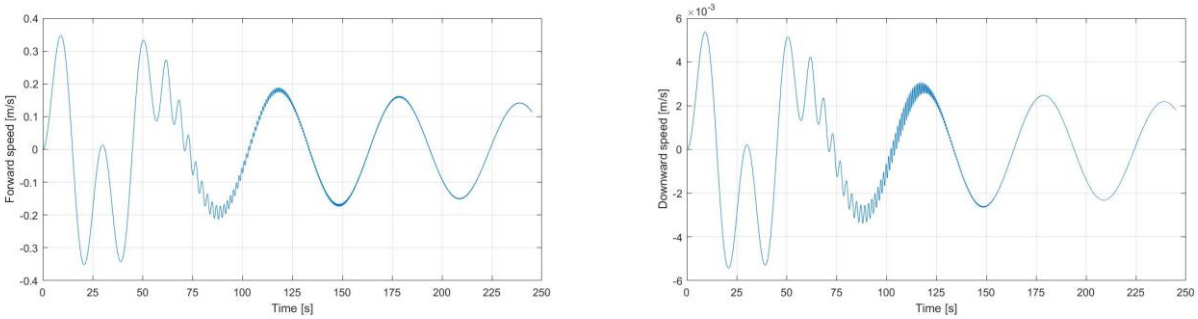


Figure 3.6: Output throttle sweep a) Forward speed b) Downward speed

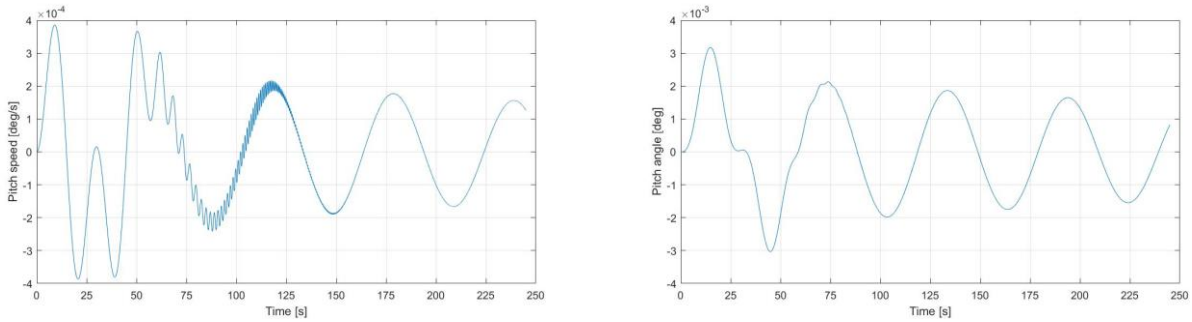


Figure 3.7: Output throttle a) Pitch speed b) Pitch angle

At this point, the steps described in the method are followed. These input-output signals are converted to frequency domain. In Appendix H, all of the bode plots for the input-output relations are included. From these bode plots, the SISO system for the pitch response can be derived. Figure 3.8 shows the LOES for the pitch response and Figure 3.9 shows the normal pitch response. Table 3.2 contains the estimated parameters from the pitch response.

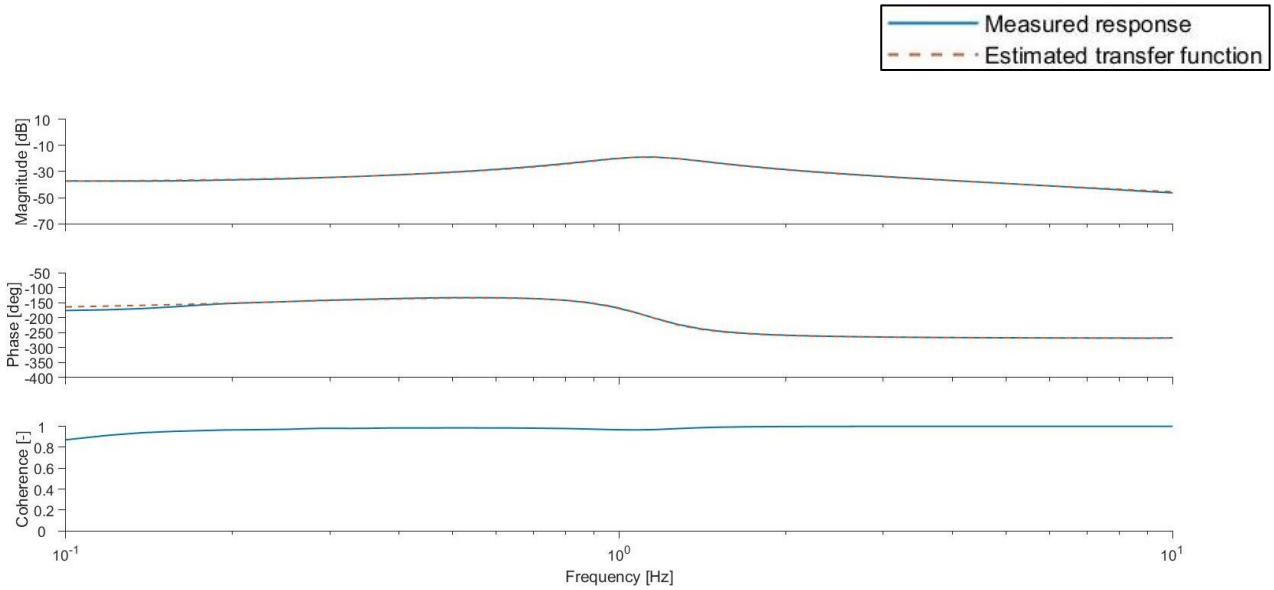


Figure 3.8: LOES Pitch speed response relative to elevator deflection fit

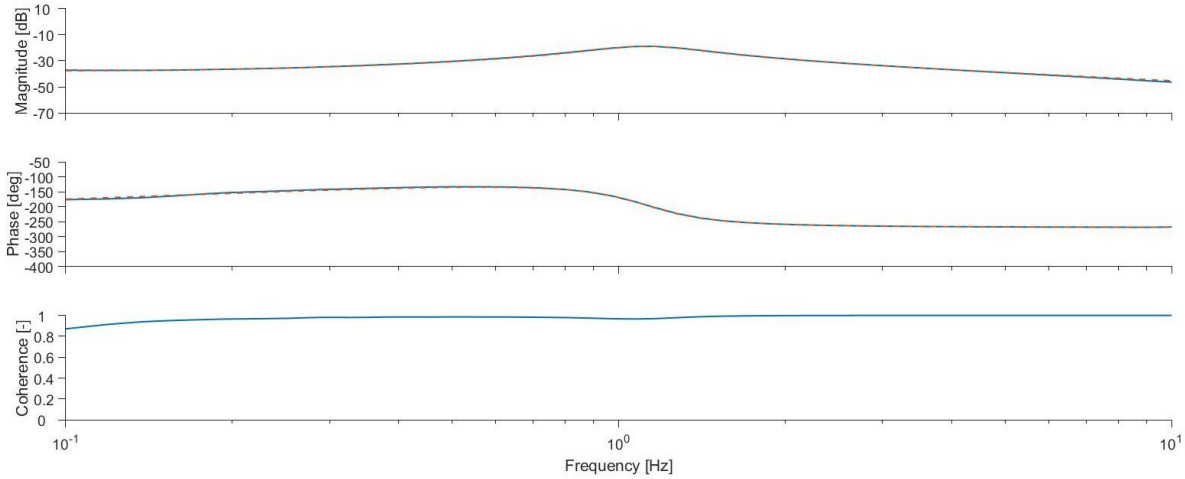


Figure 3.9: Pitch speed response relative to elevator deflection fit

Table 3.2: Pitch response parameters

Parameter:	LOES:	Pitch response:	Unit:
K_q	-	-0.33	-
T_{θ_1}	-	8.2	s
T_{θ_2}	0.51	0.57	s
M_{δ_e}	-0.33	-	deg
ζ_{sp}	0.21	0.22	-
ω_{sp}	7.1	7.1	rad/s
ζ_p	-	-0.07	-
ω_p	-	0.11	rad/s
J of $\frac{q(s)}{\delta_e}$	4.8	2.1	-

The MIMO model is derived from 7 of the 8 input output relations. The coherence of the pitch angle response relative to throttle change is below 0.6 and therefore this response is discarded. The fits of the state space system on the responses are shown in Appendix I. The derived parameters of the state space system are shown in Table 3.3.

3.2 Comparison

The final result of the MIMO analysis is shown in Table 3.3, from this can be concluded that the method works satisfactory. The results of the MIMO analysis show that when the Cramér-Rao bounds are unsatisfactory, the estimation of the parameters is also unsatisfactory.

The estimated phugoid is very close to the minimum frequency of the frequency sweep, still the frequency is well represented. The damping on the contrary, is not well represented, in the report, the phugoid is lightly damped but in the verification results it is slightly undamped. This can be explained by the bad representation of the forward speed response parameters, the pitch speed contribution and the low coherence (below 0.6) for most of the response pairs at the lower frequencies. The Cramér-Rao bounds of these parameters (X_u, X_w, X_q and M_q) are high compared to the other parameters, which indicates that the value of these parameter is not trustworthy.

Table 3.3: MIMO analysis results compared to initial values of the longitudinal state space matrix

Parameter:	[56]:	Computed results:	Unit:	Cramér-Rao:	Insensitivity:	Difference:
X_u	0.0028	0.0206	1/s	820%	320%	640%
X_w	-0.308	-0.287	1/s	20%	8.8%	6.8%
X_q	-2.52	-2.65	m/s	44%	17%	4.8%
X_{δ_e}	-0.895	-0.877	m/s^2	4.8%	2.2%	2.0%
X_{δ_T}	0.257	0.252	m/s^2	4.2%	1.2%	1.8%
Z_u	-0.102	-0.105	1/s	9.4%	2.6%	2.4%
Z_w	-2.83	-2.82	1/s	8.6%	1.6%	0.5%
Z_q	132	134	m/s	4.1%	0.7%	1.3%
Z_{δ_e}	-0.967	-0.848	m/s^2	17%	4%	12%
M_u	0.0059	0.0061	1/s	8.9%	2.2%	3.4%
M_w	-0.371	-0.367	1/s	4.5%	0.7%	1.1%
M_q	-0.156	-0.228	1/s	140%	22%	46%
M_{δ_e}	-0.341	-0.335	1/s^2	3.0%	3.0%	1.7%
J of $\frac{u(s)}{\delta_e}$	-	2.7	-	-	-	-
J of $\frac{w(s)}{\delta_e}$	-	0.4	-	-	-	-
J of $\frac{q(s)}{\delta_e}$	-	4.5	-	-	-	-
J of $\frac{\theta(s)}{\delta_T}$	-	0.9	-	-	-	-
J of $\frac{u(s)}{\delta_T}$	-	2.6	-	-	-	-
J of $\frac{w(s)}{\delta_T}$	-	1.5	-	-	-	-
J of $\frac{q(s)}{\delta_T}$	-	1.4	-	-	-	-
ζ_{sp}	0.212	0.215	-	-	-	1.4%
ω_{sp}	7.04	7.06	rad/s	-	-	0.3%
ζ_p	0.02	-0.07	-	-	-	450%
ω_p	0.104	0.105	rad/s	-	-	1.0%

4 Results

4.1 Flight test results

The elevator input, throttle input, forward acceleration, downward acceleration, forward speed, downward speed, pitch angle and pitch speed are measured during the maneuvers. In these experiments three sweeps are performed for the elevator and one sweep is performed for the throttle. Figure 4.1, Figure 4.5 & Figure 4.9 show the inputs for the elevator sweeps. Figure 4.2-Figure 4.4, Figure 4.6-Figure 4.8 and Figure 4.10-Figure 4.12 show the outputs for the elevator sweep. Figure 4.13 shows the inputs for the throttle sweep and Figure 4.14-Figure 4.16 shows the outputs of the throttle sweep.

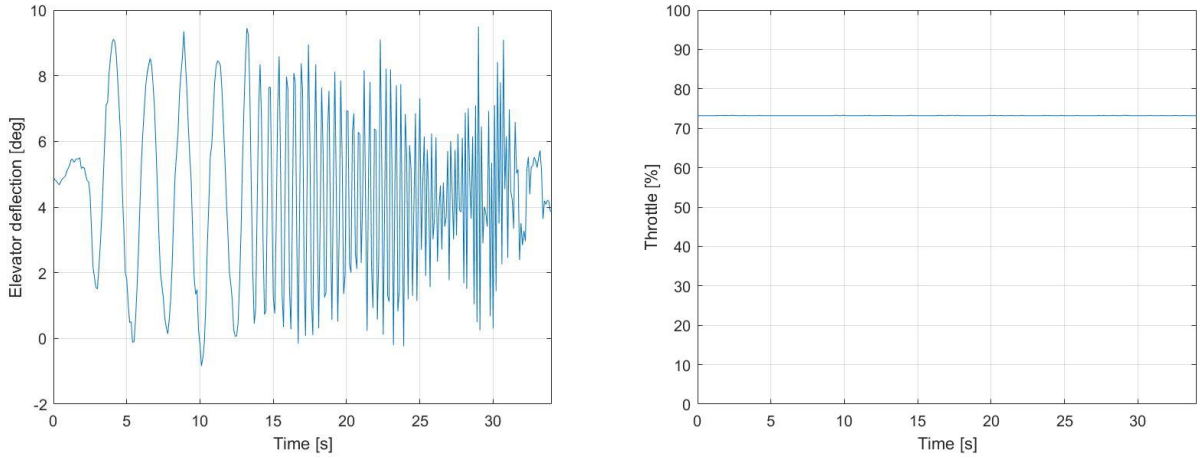


Figure 4.1: First elevator sweep: a) Elevator deflection b) Throttle setting

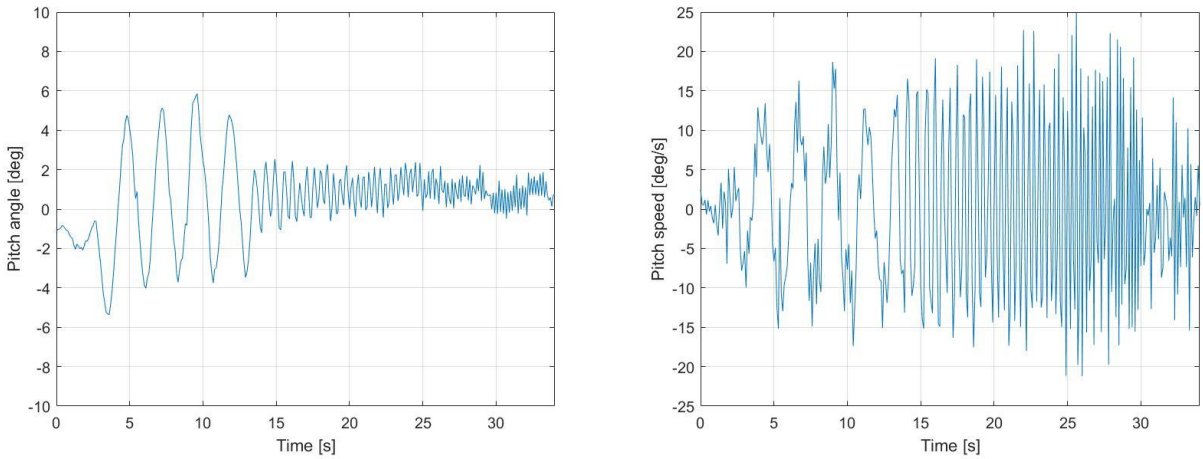


Figure 4.2: First elevator sweep: a) Pitch angle b) Pitch speed

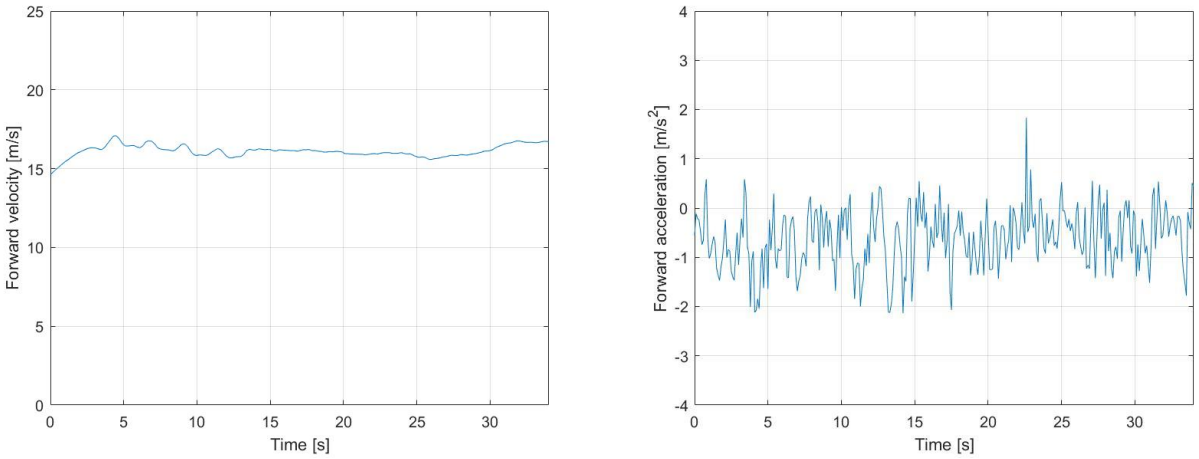


Figure 4.3: First elevator sweep: a) Forward velocity b) Forward acceleration

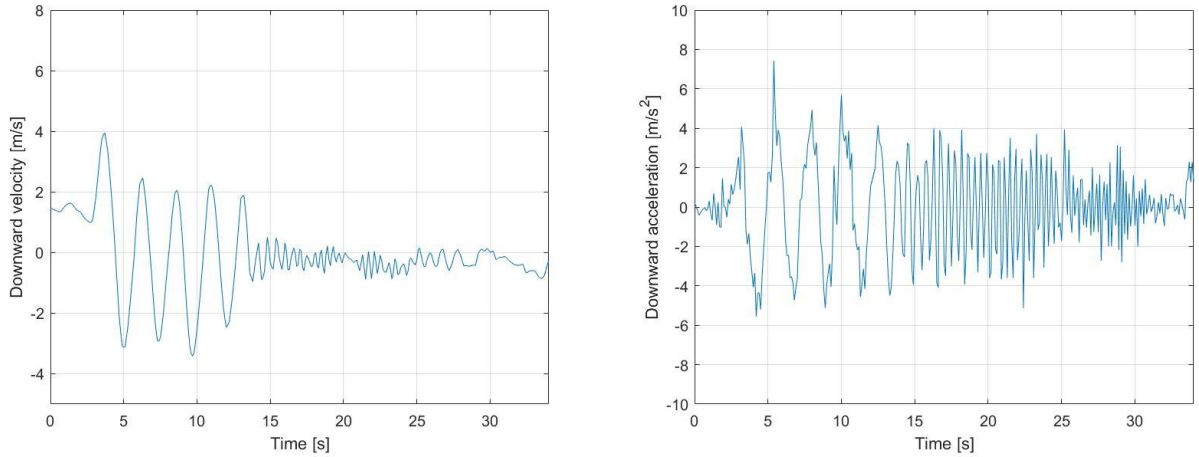


Figure 4.4: First elevator sweep: a) Downward velocity b) Downward acceleration

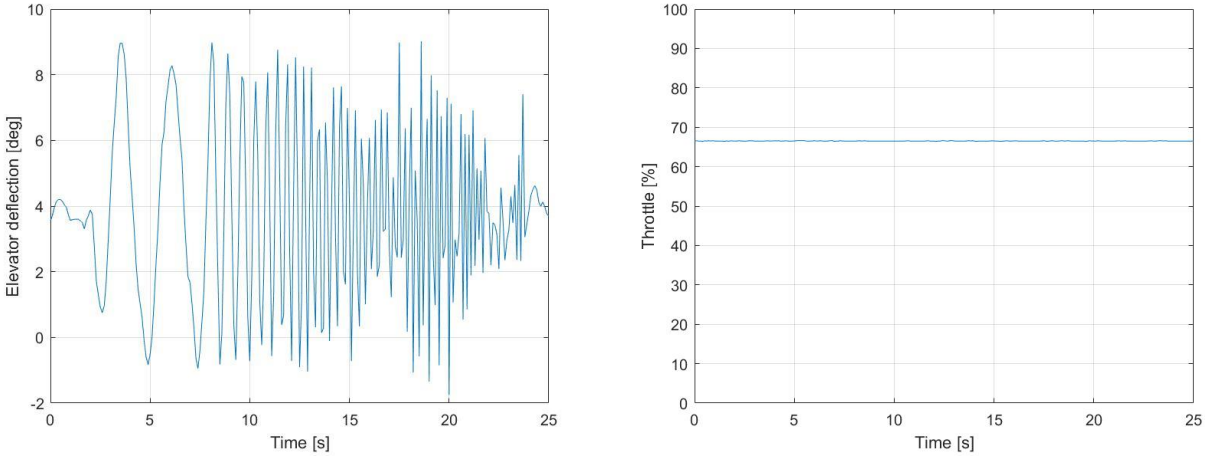


Figure 4.5: Second elevator sweep: a) Elevator deflection b) Throttle setting

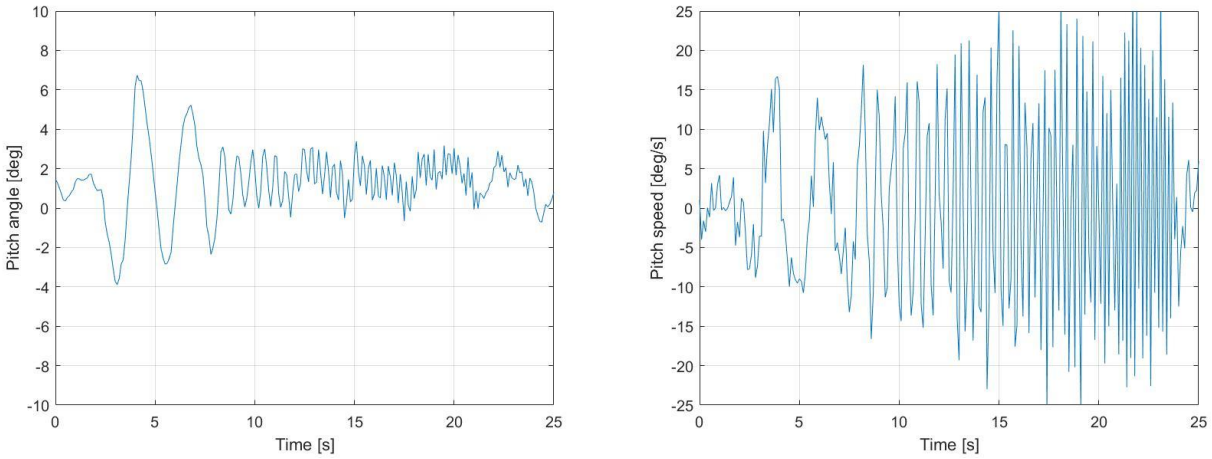


Figure 4.6: Second elevator sweep: a) Pitch angle b) Pitch speed

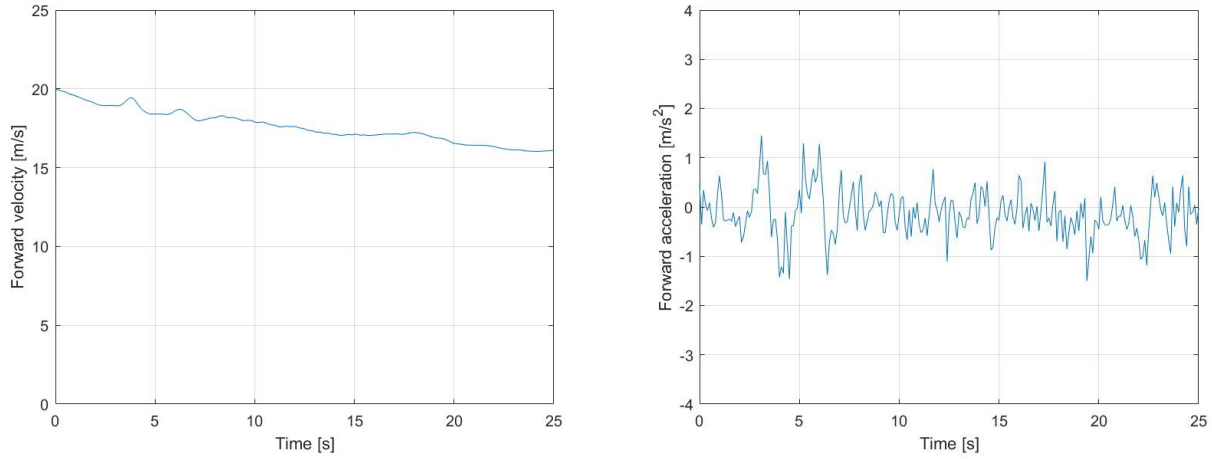


Figure 4.7: Second elevator sweep: a) Forward velocity b) Forward acceleration

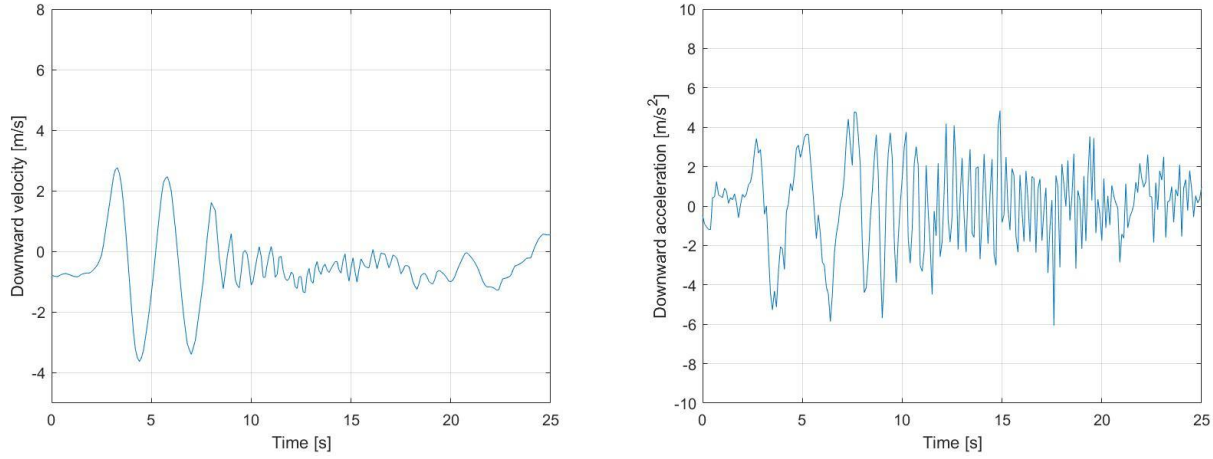


Figure 4.8: Second elevator sweep: a) Downward velocity b) Downward acceleration

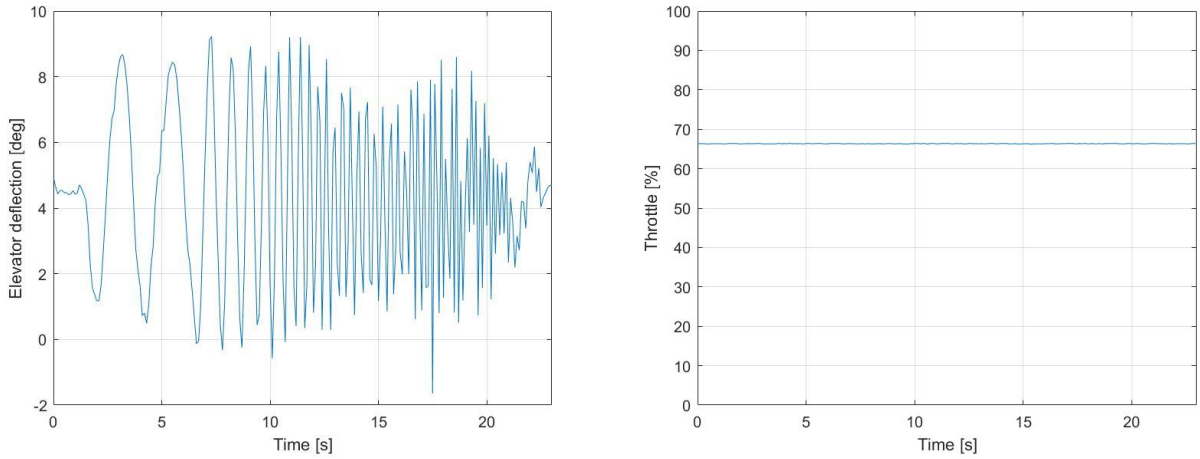


Figure 4.9: Third elevator sweep: a) Elevator deflection b) Throttle setting

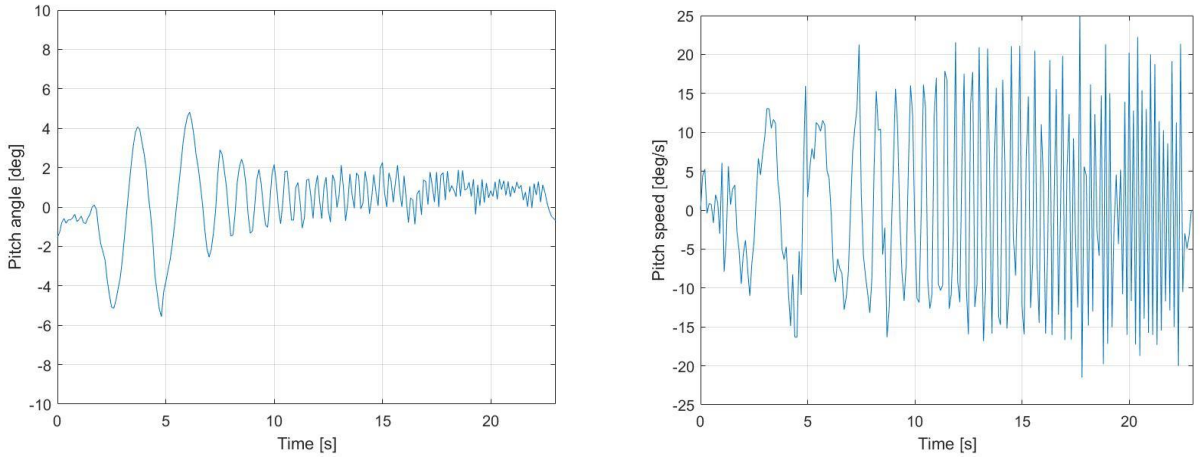


Figure 4.10: Third elevator sweep: a) Pitch angle b) Pitch speed

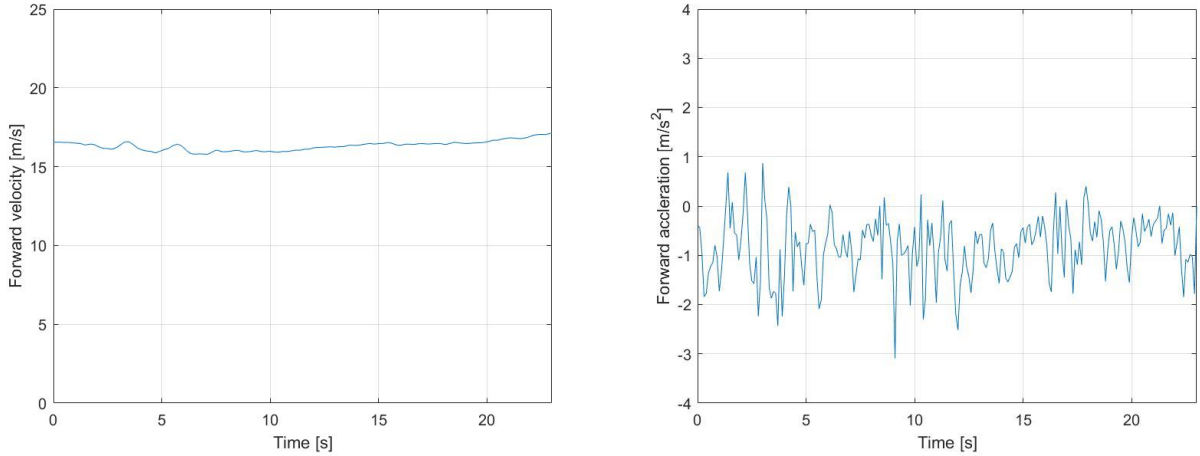


Figure 4.11: Third elevator sweep: a) Forward velocity b) Forward acceleration

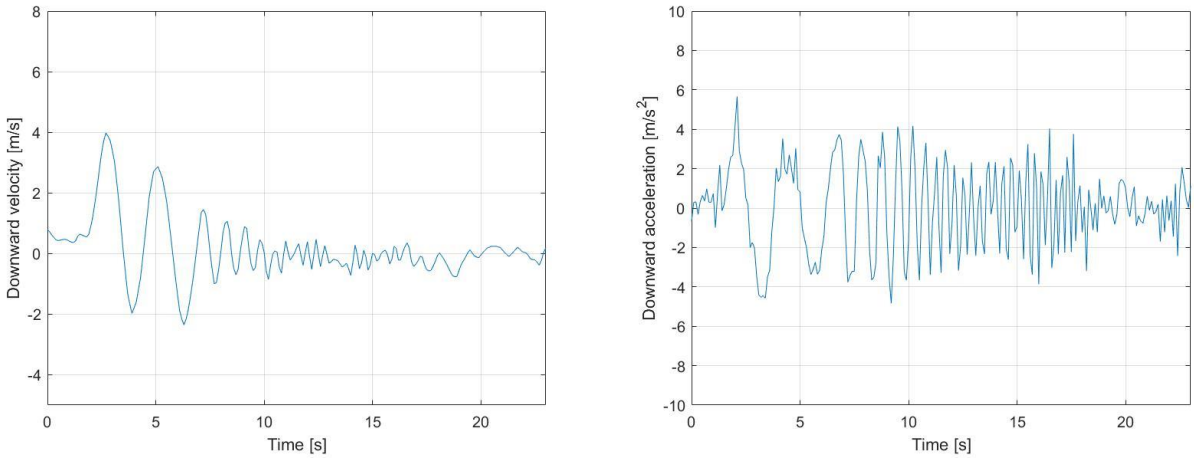


Figure 4.12: Third elevator sweep: a) Downward velocity b) Downward acceleration

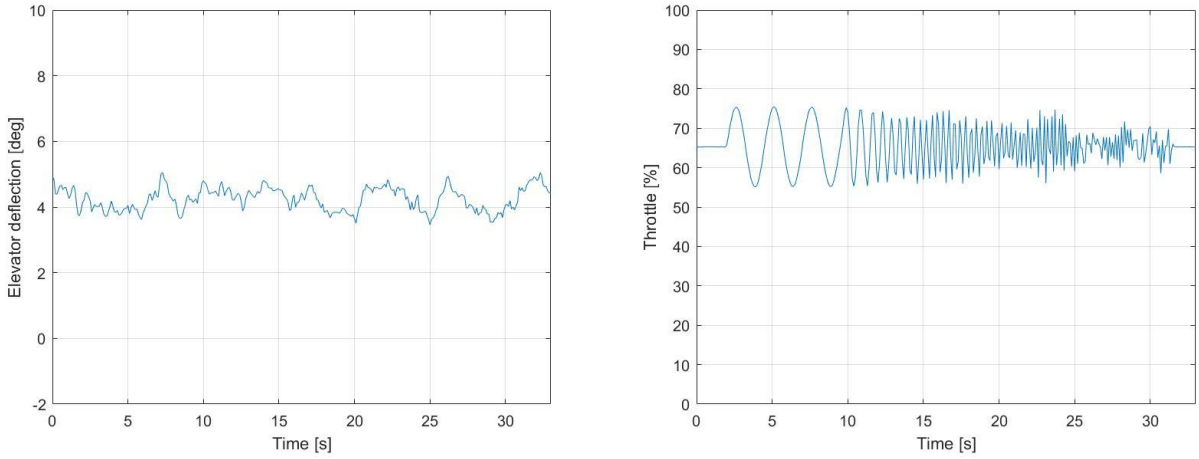


Figure 4.13: Throttle sweep: a) Elevator deflection b) Throttle setting

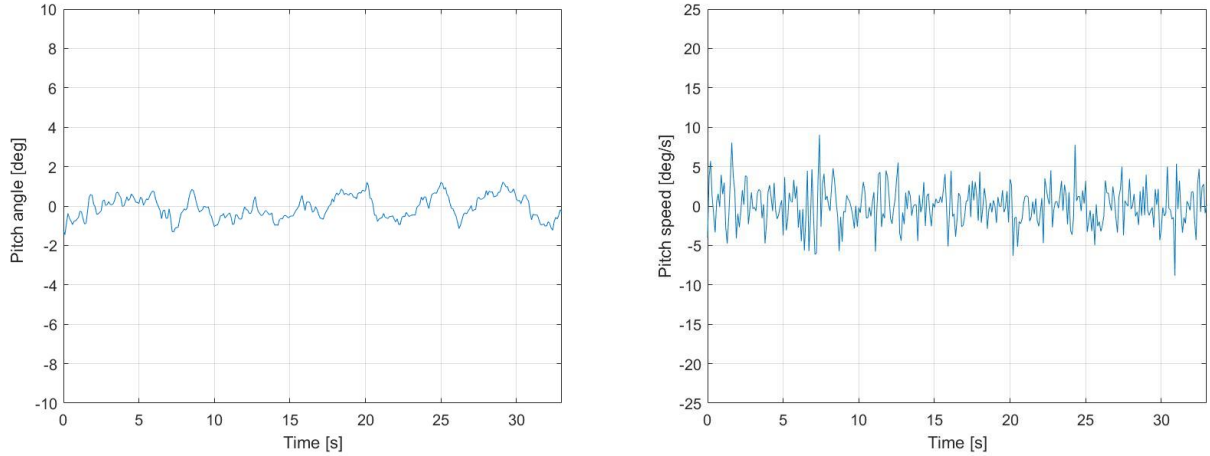


Figure 4.14: Throttle sweep: a) Pitch angle b) Pitch speed

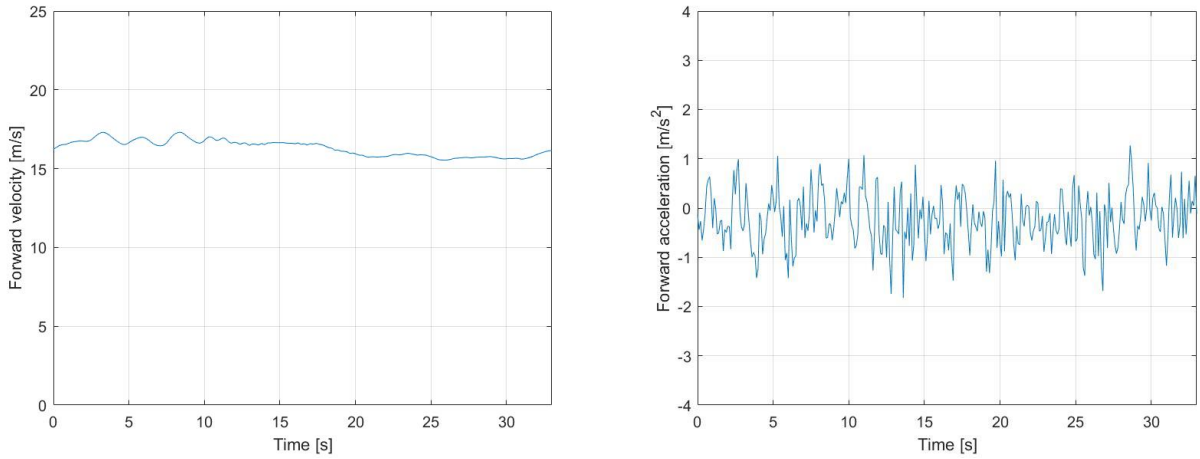


Figure 4.15: Throttle sweep: a) Forward velocity b) Forward acceleration

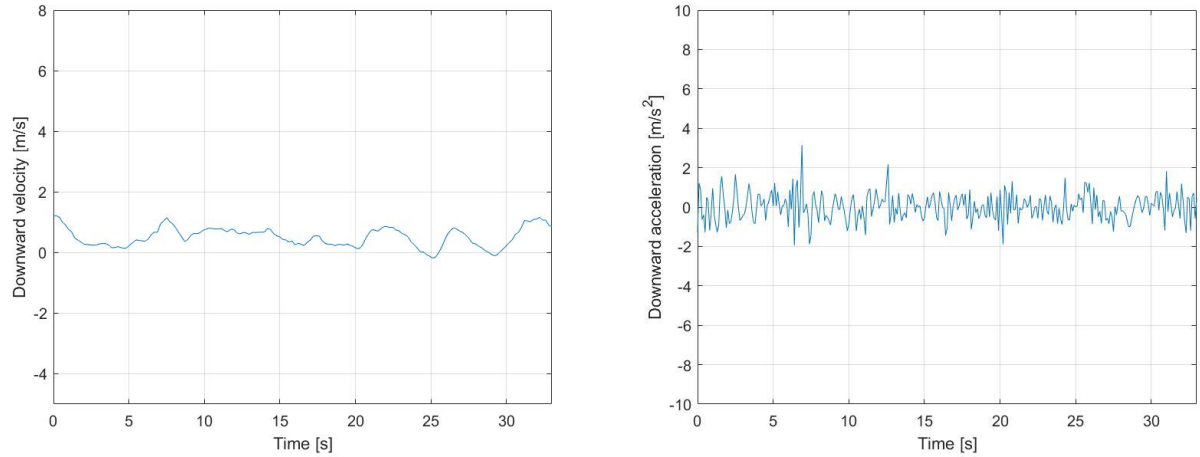


Figure 4.16: Throttle sweep: a) Downward velocity b) Downward acceleration

The time domain results indicate that it is difficult to accurately trim the aircraft, in every output plot the first two seconds are never a straight line. This trim issue can be explained by the size of the aircraft, the aircraft is easily distorted by the surroundings and therefore the autopilot is always correcting for small errors.

The resemblance between the input of the elevator and the pitch angle and downward speed indicates a good response. The first two low frequency periods are clearly visible, which indicates that the input signal is related to the output. Unfortunately, this is not visible in the forward speed, as at low frequencies there is a low ripple visible in the plot but there is no clear influence from the input signal. The acceleration outputs are more difficult to interpret because they are more distorted. The downward acceleration and pitch speed seem to be influenced by the input signal, which is clearly visible for the low frequencies. The forward acceleration is clearly not influenced by the input signal and therefore the forward speed is probably not well represented.

4.2 System identification

4.2.1 Input signals and output signals

In this section the filtered flight test data is shown. The data is modified by removing the drift and the mean of the signal and by filtering of the signal. When available, multiple sweeps are combined into one signal to improve the accuracy of the results. For this reason, all three sweeps for the elevator are used and for the throttle only one sweep is used.

In Figure 4.17 & Figure 4.18 the input signals of the filtered and conjugated frequency sweeps of the elevator and in Figure 4.19-Figure 4.24, the output signals of the filtered and conjugated frequency sweeps of the elevator are shown. This is also done for the throttle sweep with the inputs in Figure 4.25-Figure 4.26 and the outputs in Figure 4.27-Figure 4.32.

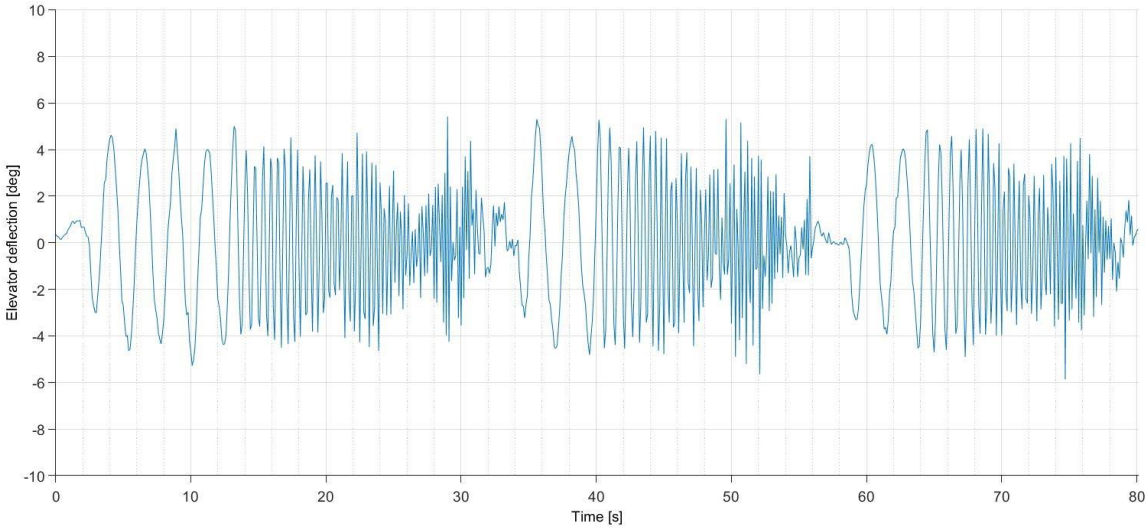


Figure 4.17: Filtered elevator deflection of the three combined elevator sweeps

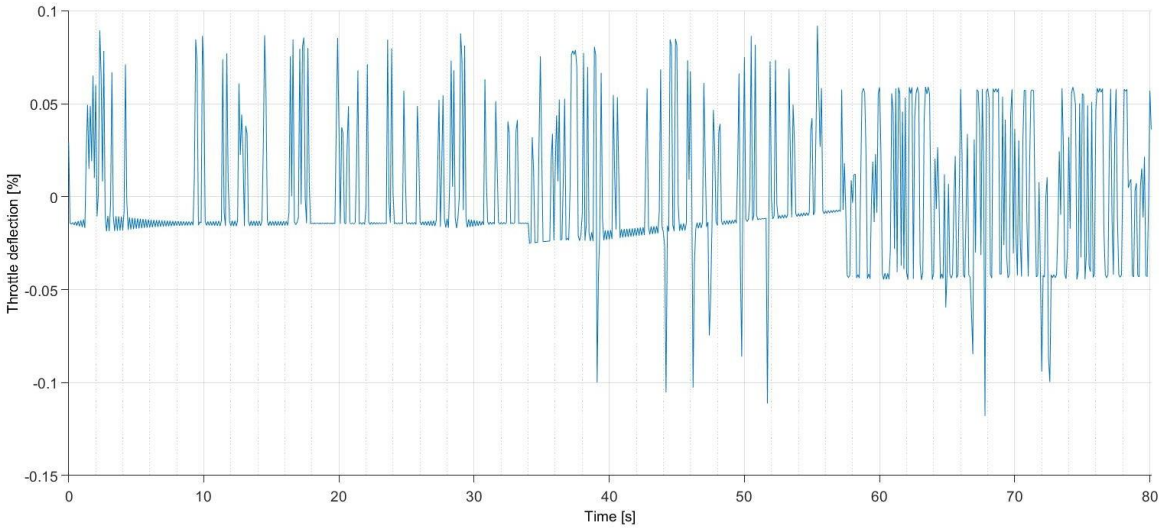


Figure 4.18: Filtered throttle deflection of the three combined elevator sweeps

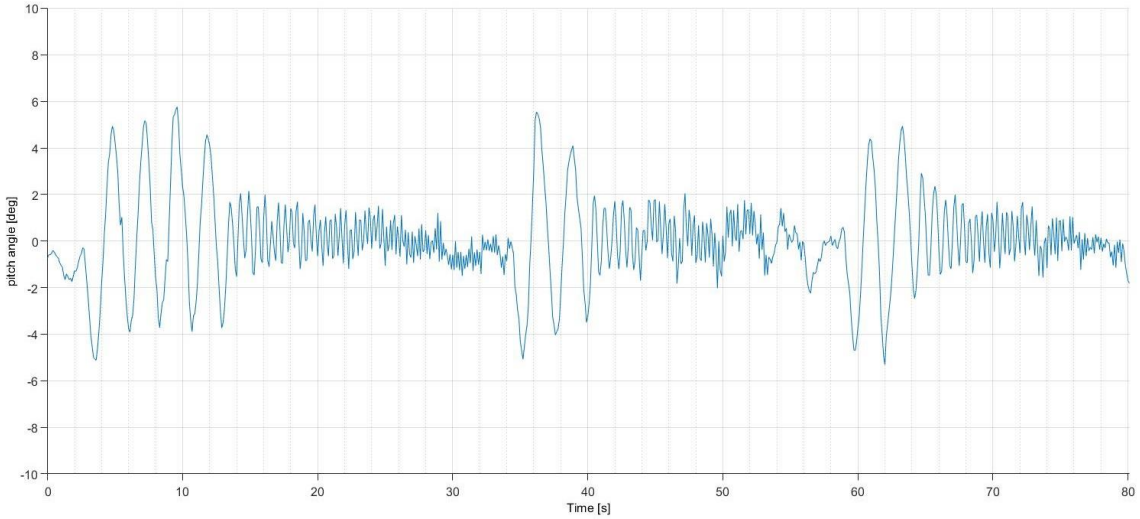


Figure 4.19: Filtered pitch angle of the three combined elevator sweeps

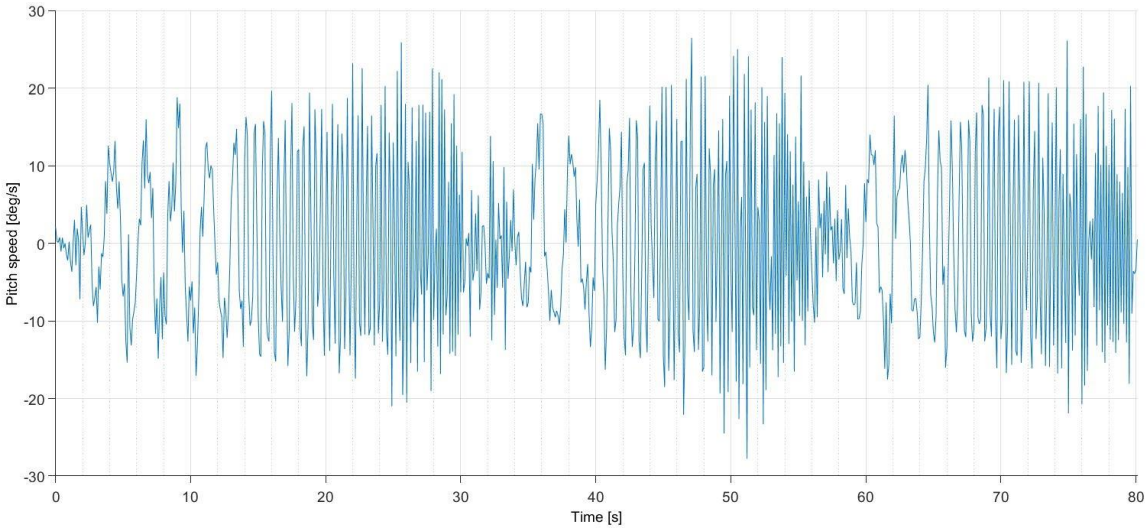


Figure 4.20: Filtered pitch speed of the three combined elevator sweeps

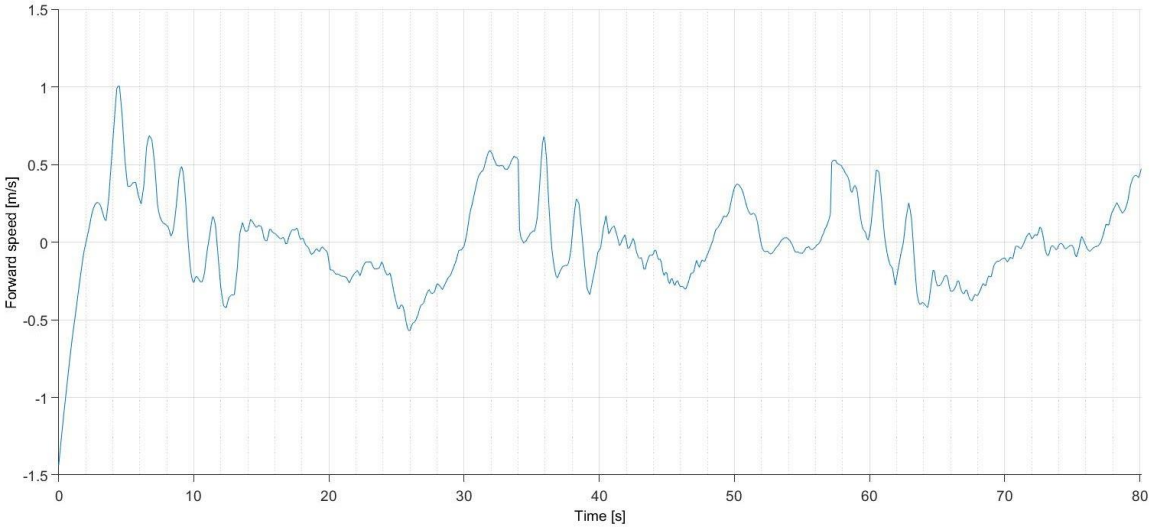


Figure 4.21: Filtered forward speed of the three combined elevator sweeps

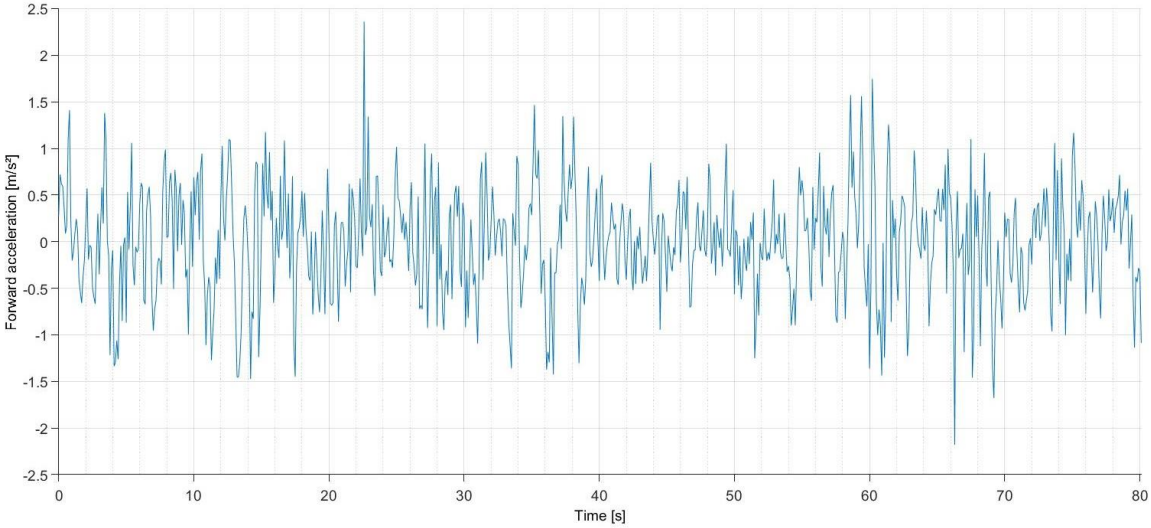


Figure 4.22: Filtered forward acceleration of the three combined elevator sweeps

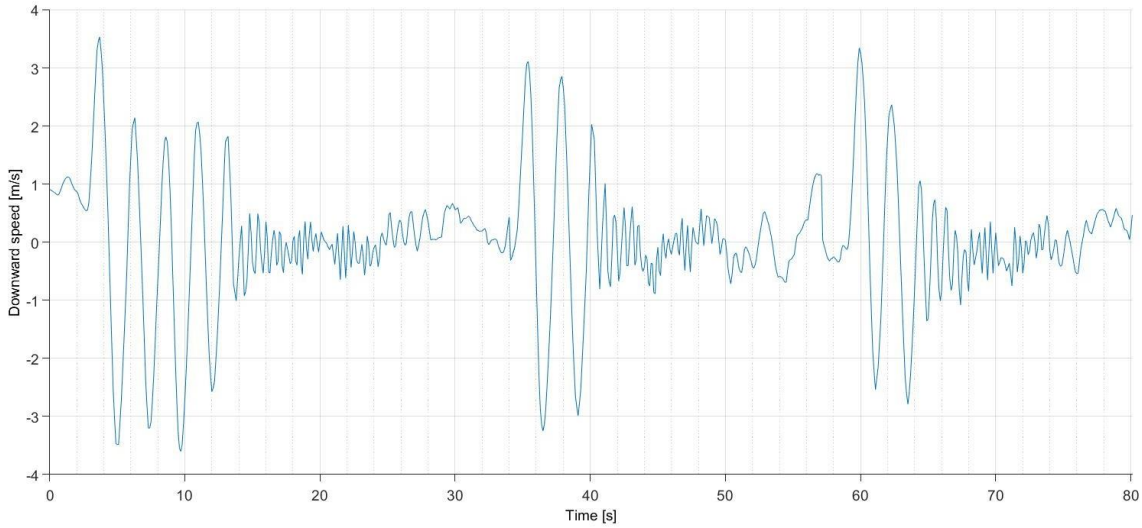


Figure 4.23: Filtered downward speed of the three combined elevator sweeps

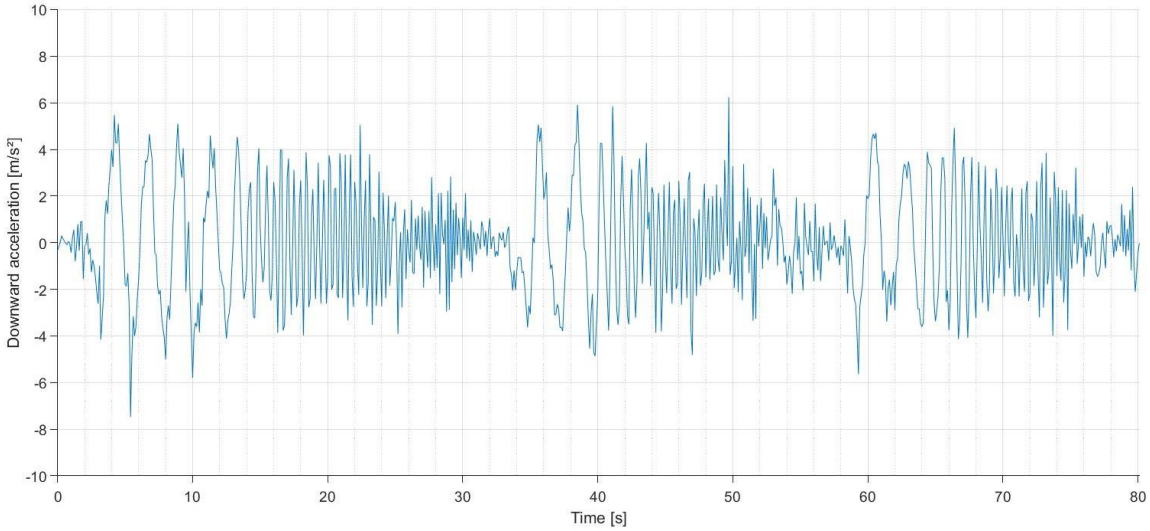


Figure 4.24: Filtered downward acceleration of the three combined elevator sweeps

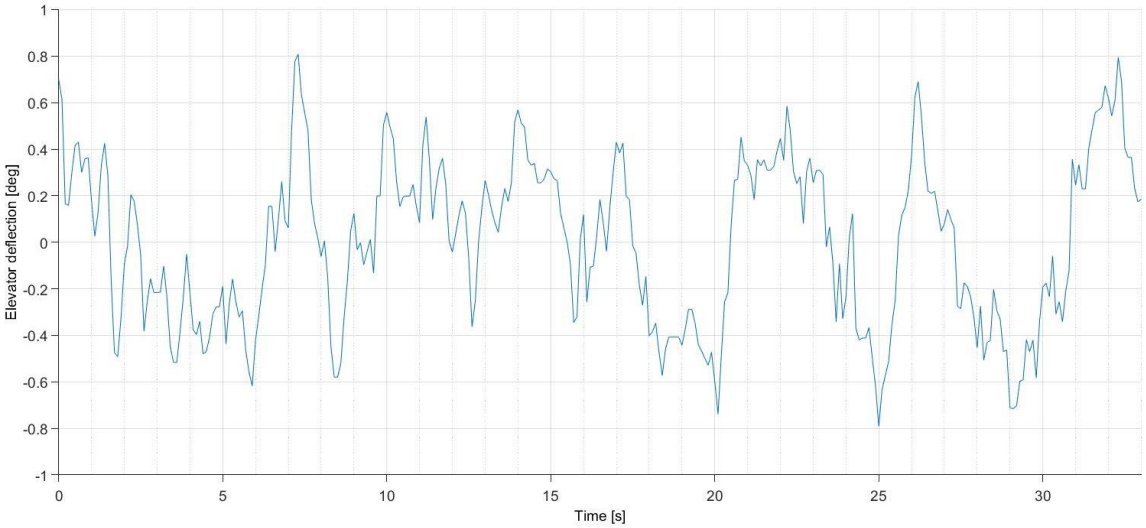


Figure 4.25: Filtered elevator deflection of the throttle sweep

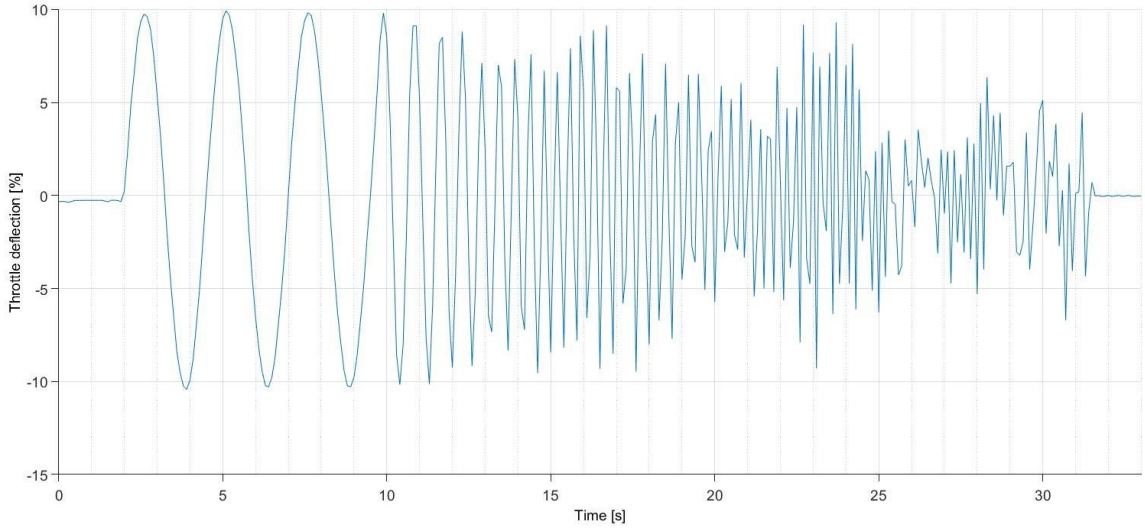


Figure 4.26: Filtered throttle deflection of the throttle sweep

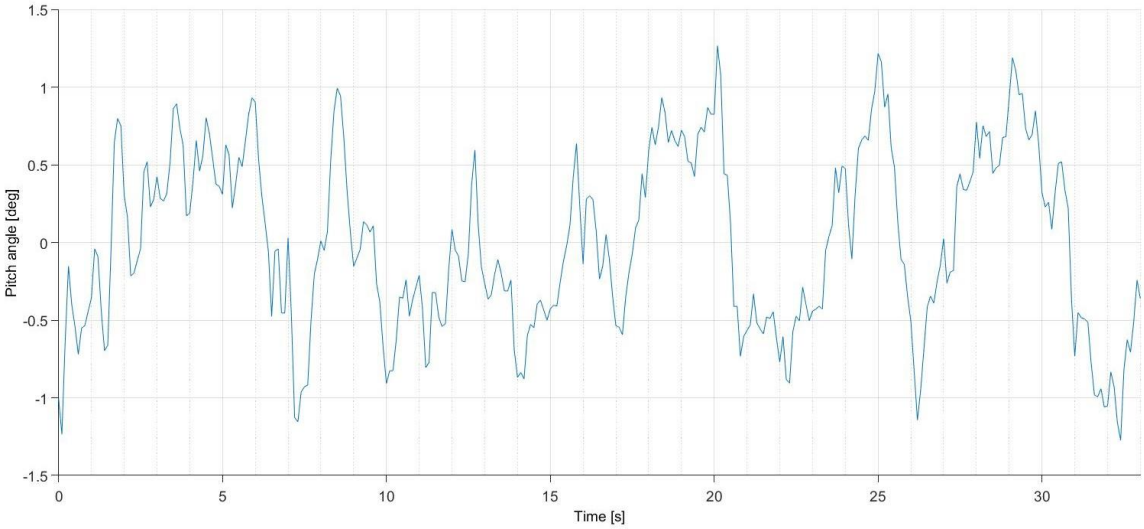


Figure 4.27: Filtered pitch angle of the throttle sweep

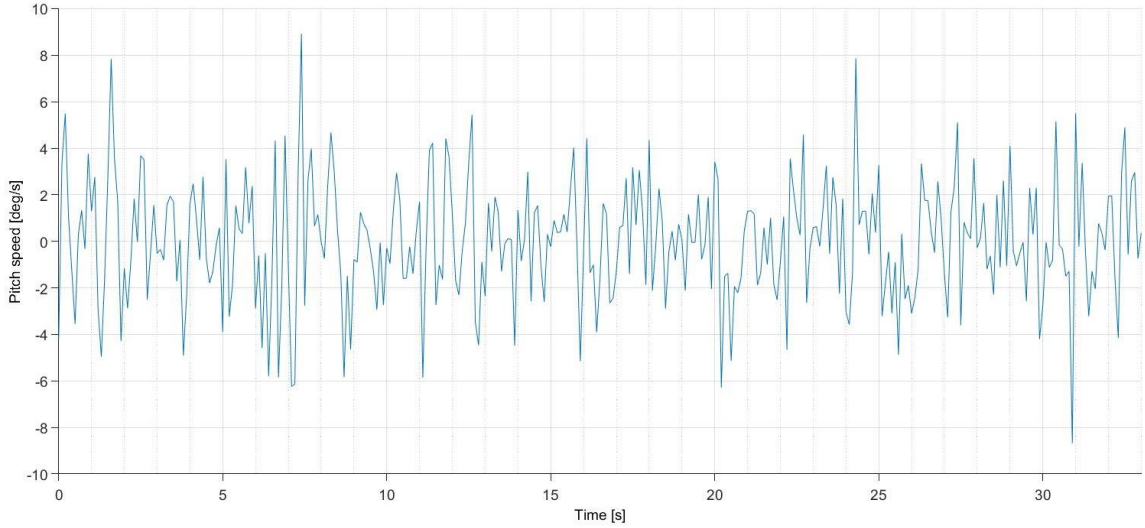


Figure 4.28: Filtered pitch speed of the throttle sweep

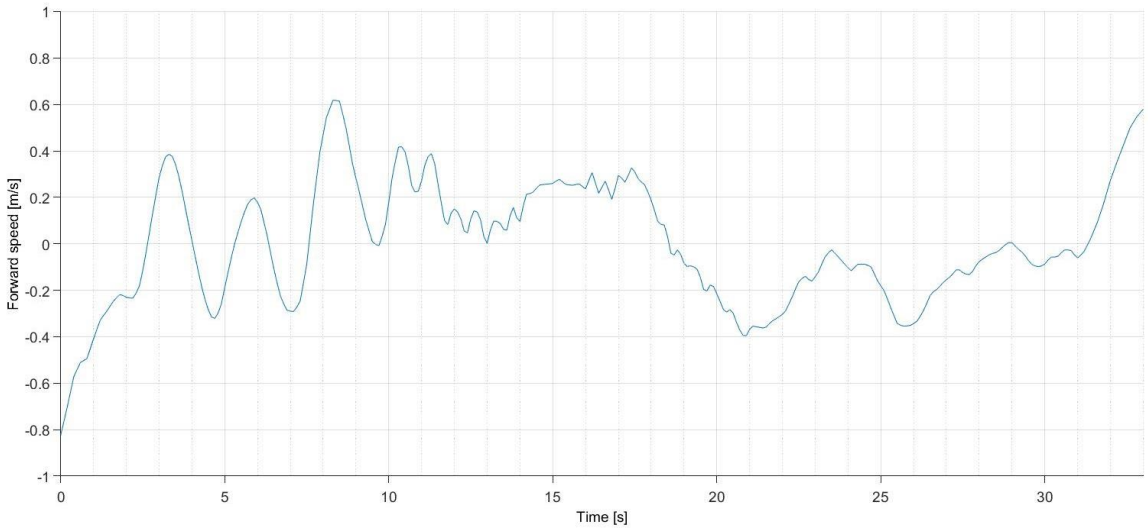


Figure 4.29: Filtered forward speed of the throttle sweep

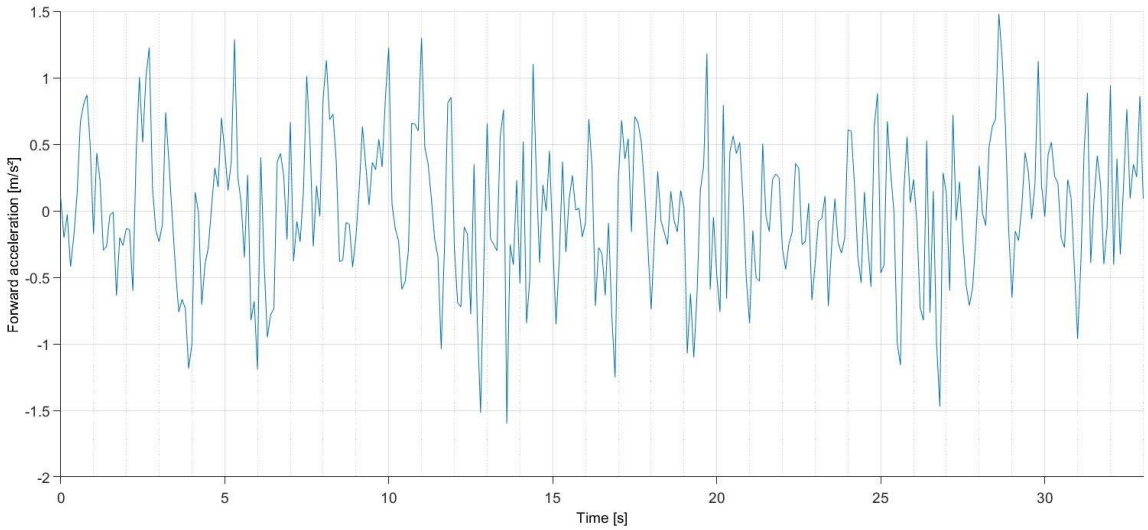


Figure 4.30: Filtered forward acceleration of the throttle sweep

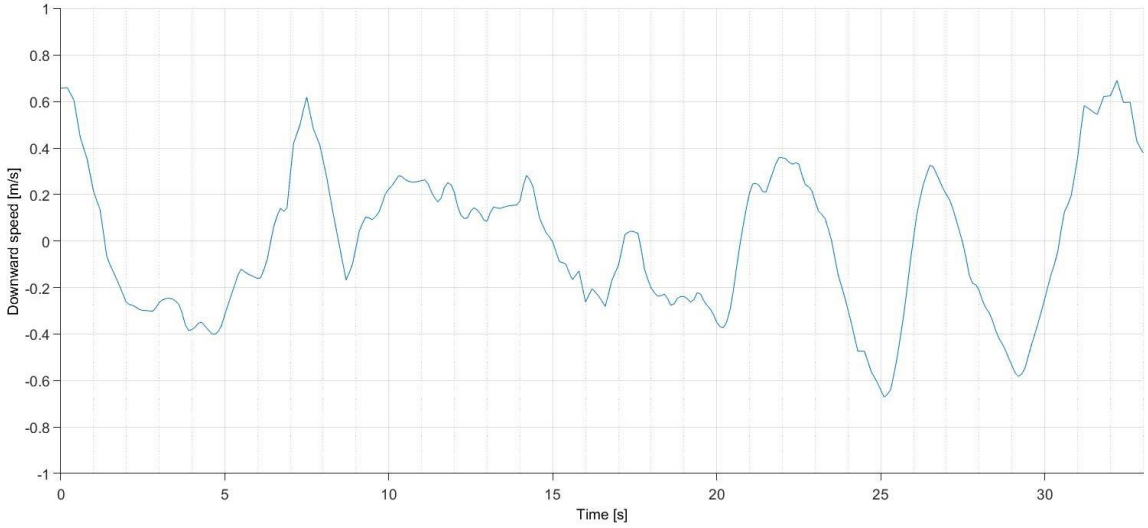


Figure 4.31: Filtered downward speed of the throttle sweep

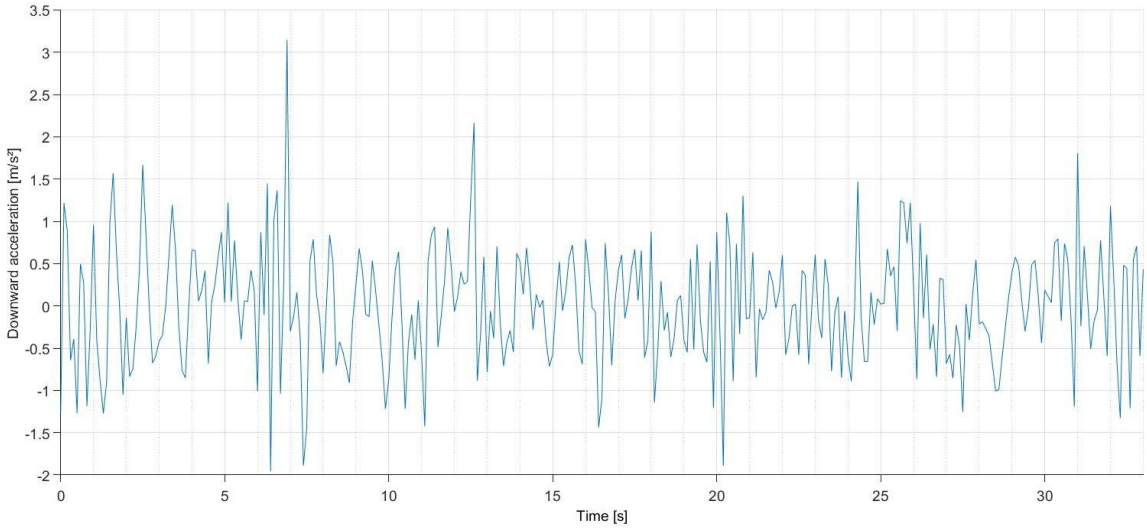


Figure 4.32: Filtered downward acceleration of the throttle sweep

4.2.2 SISO frequency responses

In this section, the bode plots from the combined elevator sweeps are shown in Figure 4.33-Figure 4.38 and the thrust sweep in Figure 4.39-Figure 4.44. Firstly, the response on the pitch angle, secondly, the response on the pitch rate, thirdly, the response on the forward speed, the response of the forward acceleration, the response of the downward speed and at lastly, the response on the downward acceleration are shown. These plots show the frequency response which is calculated FRESPID.

The results for the pitch (Figure 4.36) and pitch rate (Figure 4.35) look very promising. There is a reasonable coherence above 0.6, from 0.25 till 4 a 5 Hz. There is a small dip at 5 Hz and the coherence fluctuates hereafter, which indicates that the frequencies after 5 Hz are not captured accurately, this could be improved by improving the frequency propagation in the frequency sweep.

The response of the forward speed (Figure 4.33), does not contain much reliable information. There is a small region with good coherence at the low frequencies, which makes it possible to use this small part of the results but then the estimated transfer function or state space matrix is only valid for a small frequency region. The response of the downward speed (Figure 4.34), shows a coherence between 0.2-2 Hz. The frequency response of the throttle sweep (Figure 4.39-Figure 4.44) shows undesirable coherence and no reliable results. As expected, the coherence for forward speed and forward acceleration response (Figure 4.39 & Figure 4.43) seem to be slightly better than the rest, but these results can not be used as the region of coherence is small.

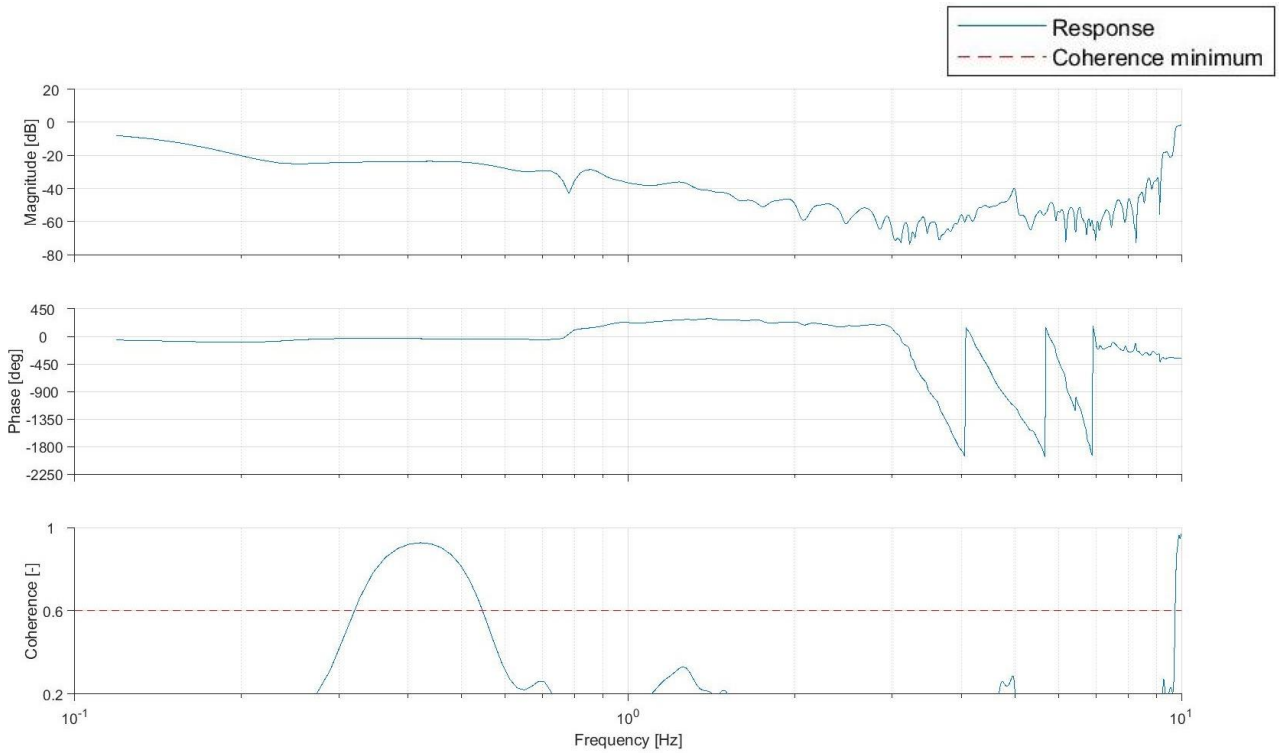


Figure 4.33: Bode plot of the forward speed response relative to the elevator deflection

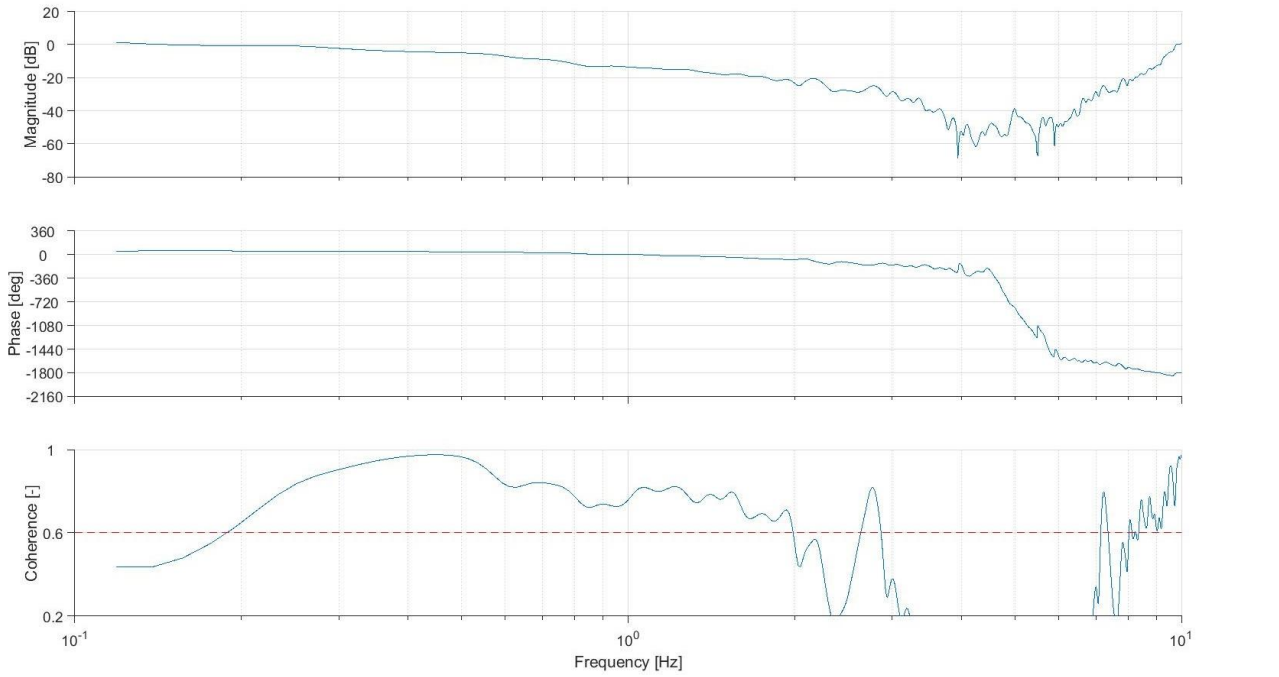


Figure 4.34: Bode plot of the downward speed response relative to elevator deflection

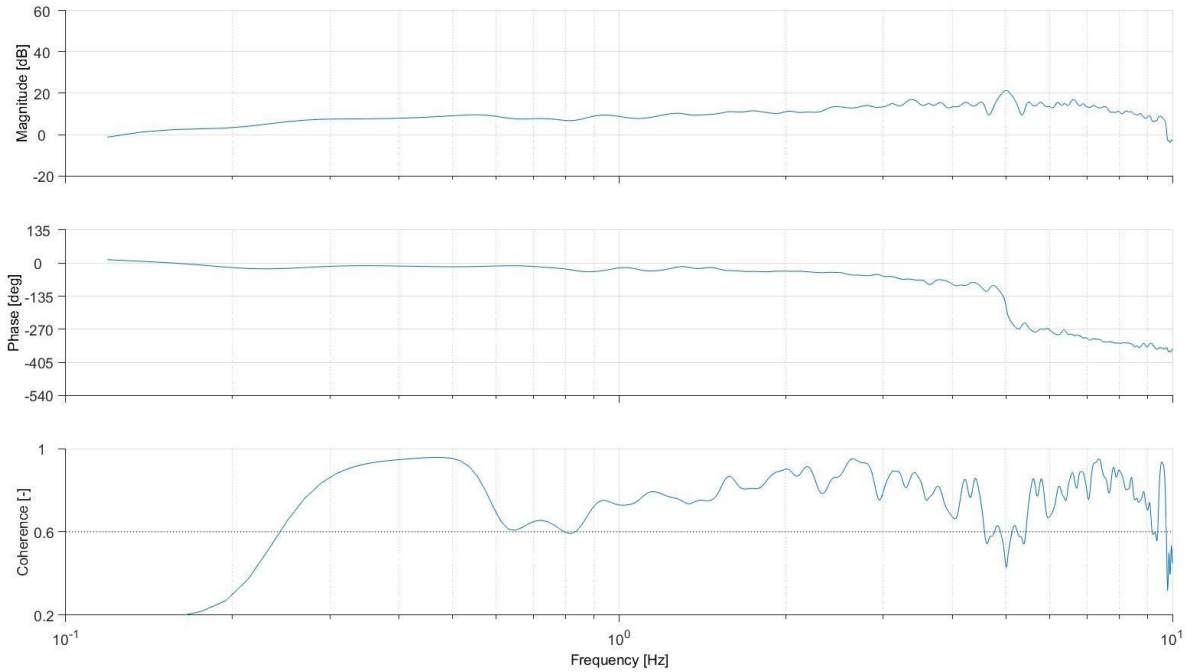


Figure 4.35: Bode plot of the pitch speed response relative to elevator deflection

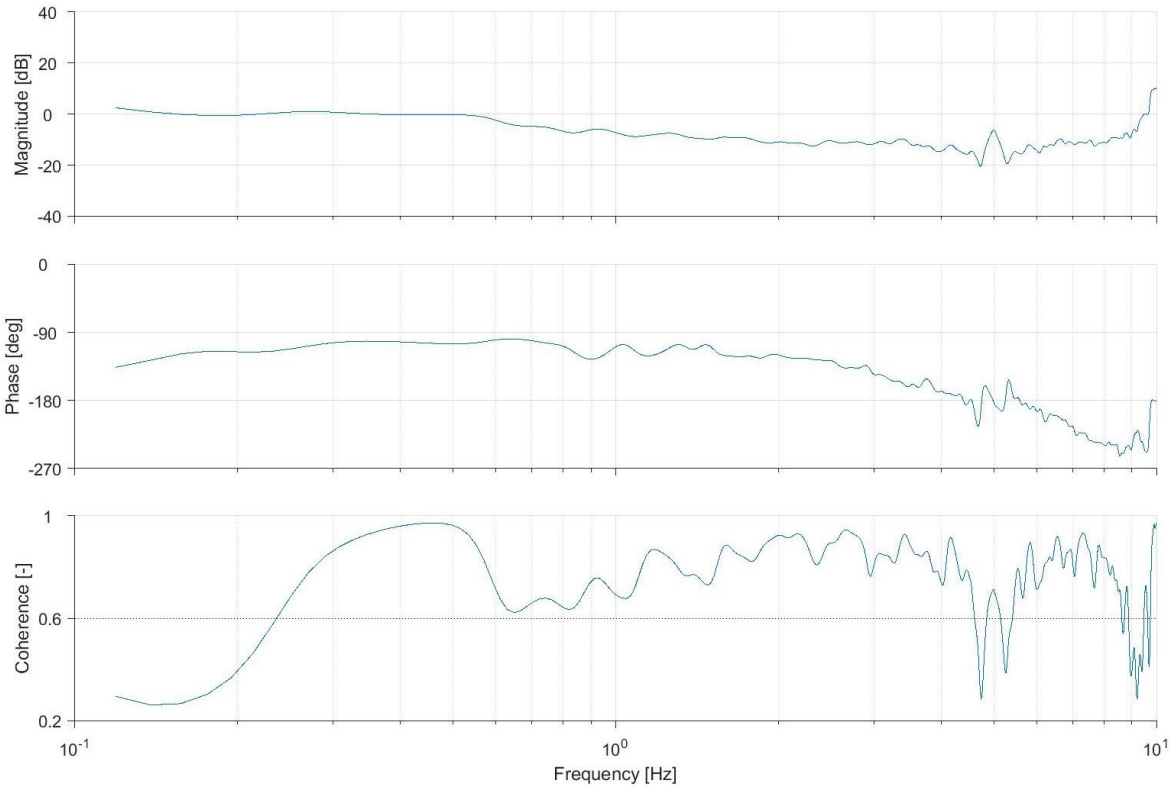


Figure 4.36: Bode plot of the pitch angle response relative to elevator deflection

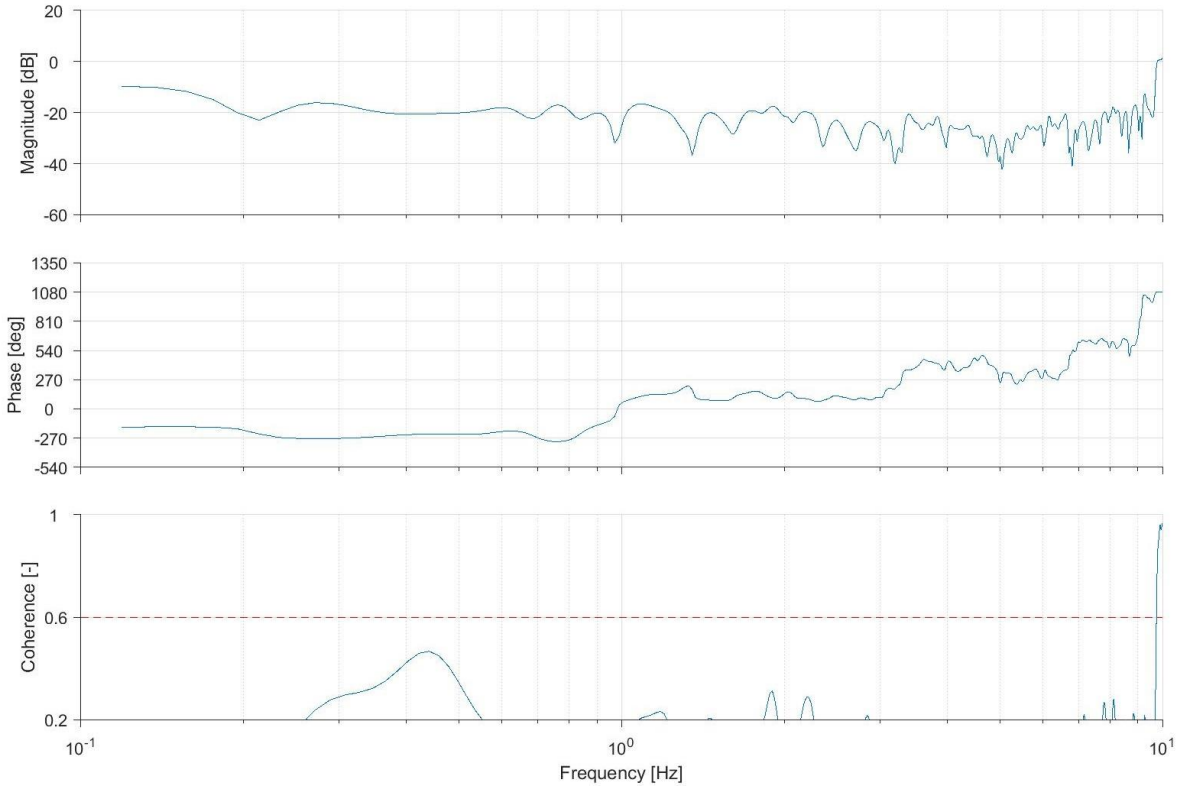


Figure 4.37: Bode plot of the forward acceleration speed response relative to elevator deflection

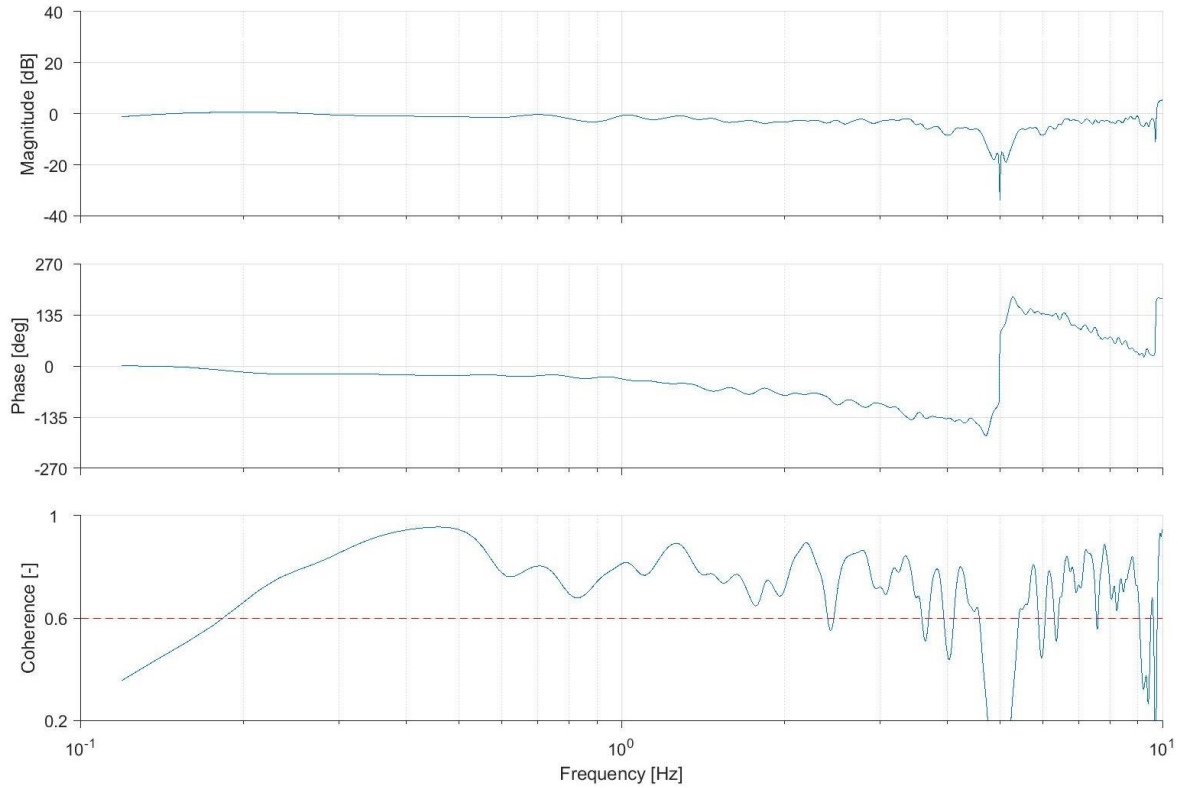


Figure 4.38: Bode plot of the downward acceleration response relative to elevator deflection

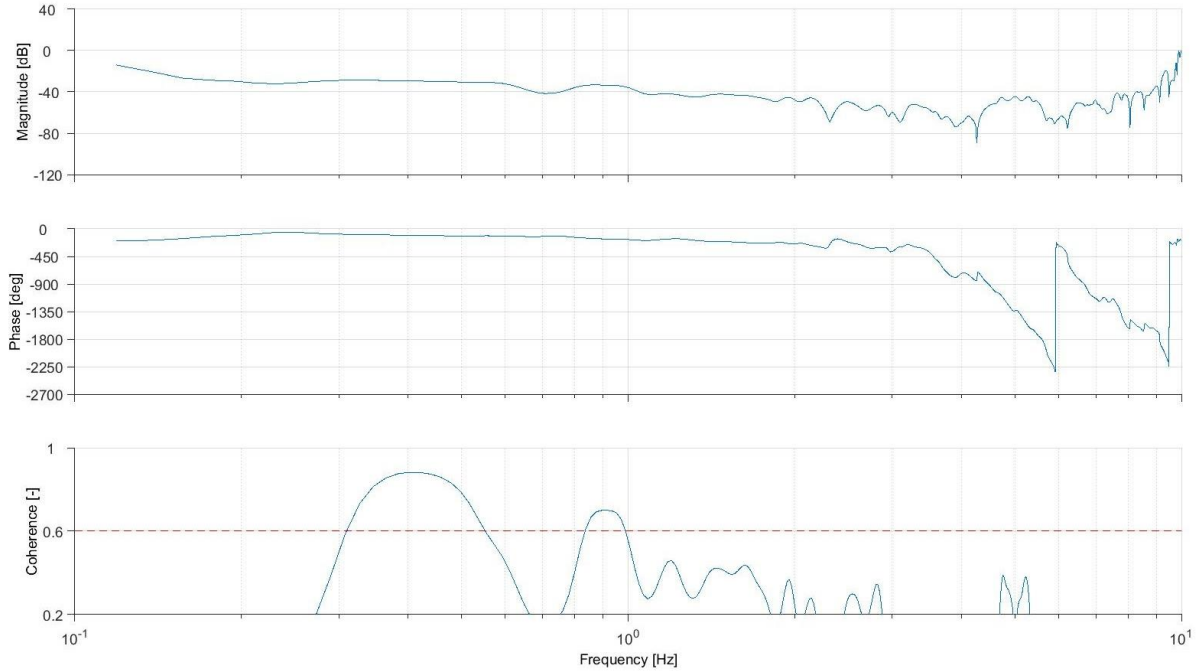


Figure 4.39: Bode plot of the forward speed response relative to throttle setting

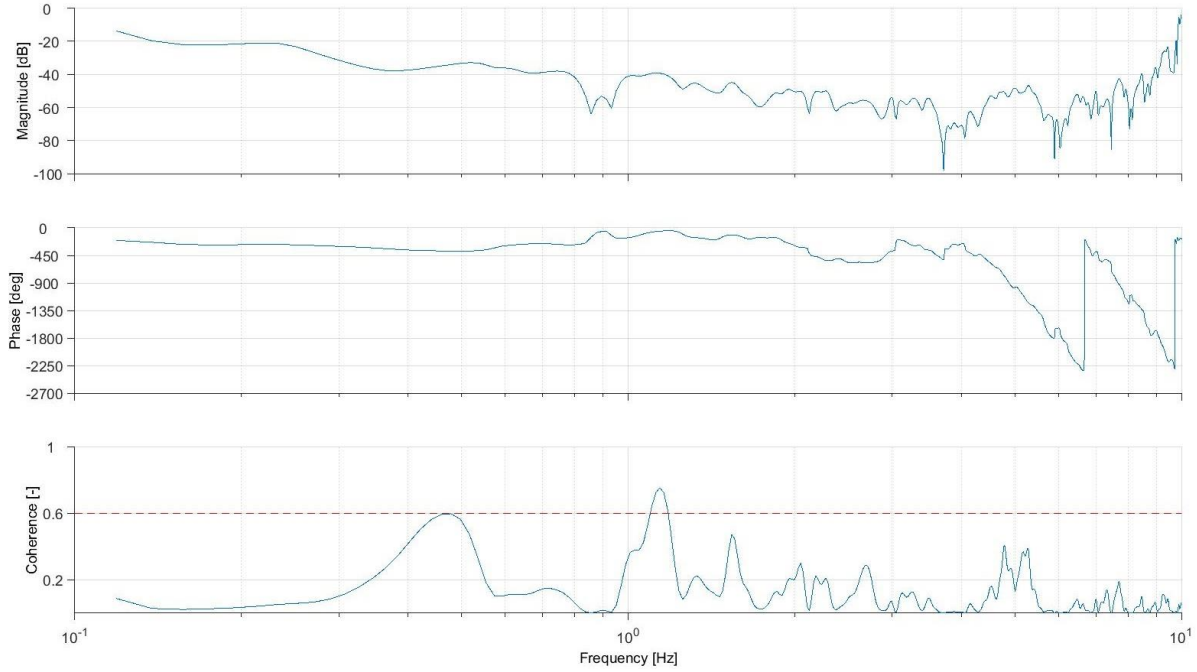


Figure 4.40: Bode plot of downward speed response relative to throttle setting

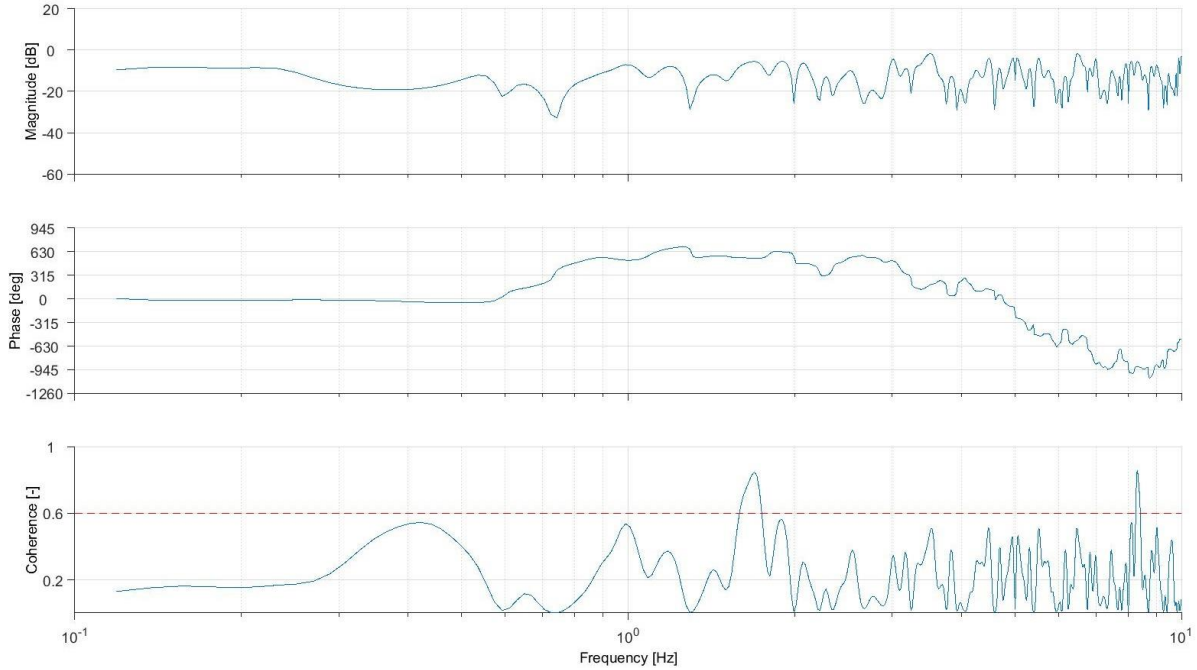


Figure 4.41: Bode plot of pitch speed response relative to throttle setting

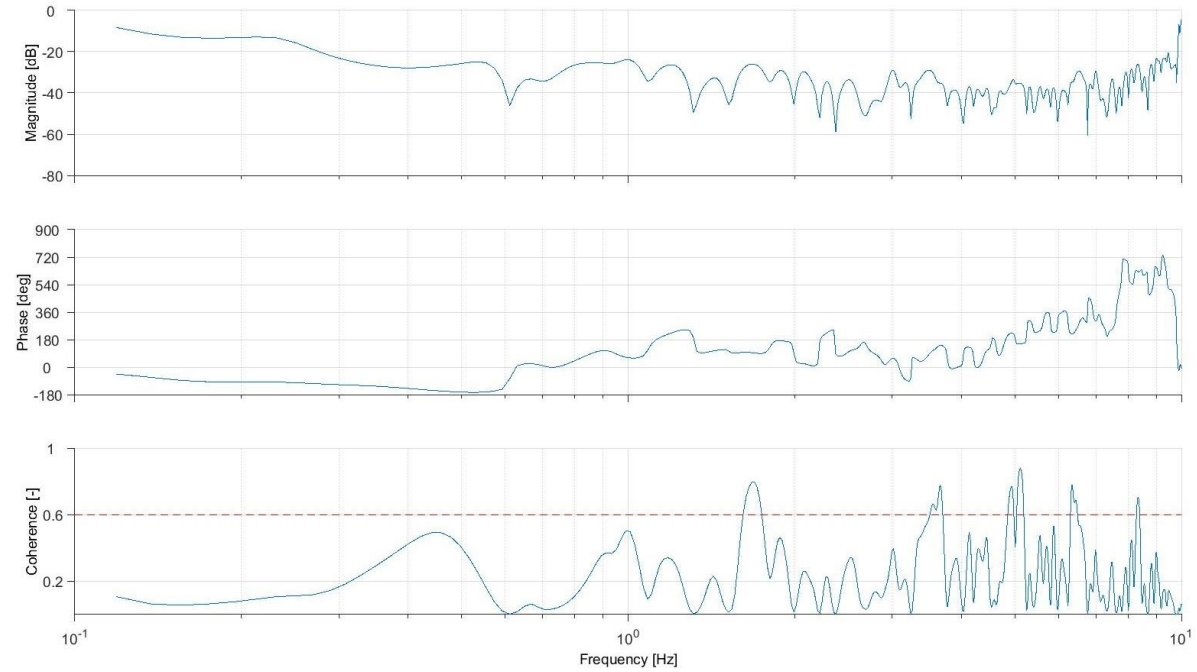


Figure 4.42: Bode plot of pitch angle response relative to throttle setting

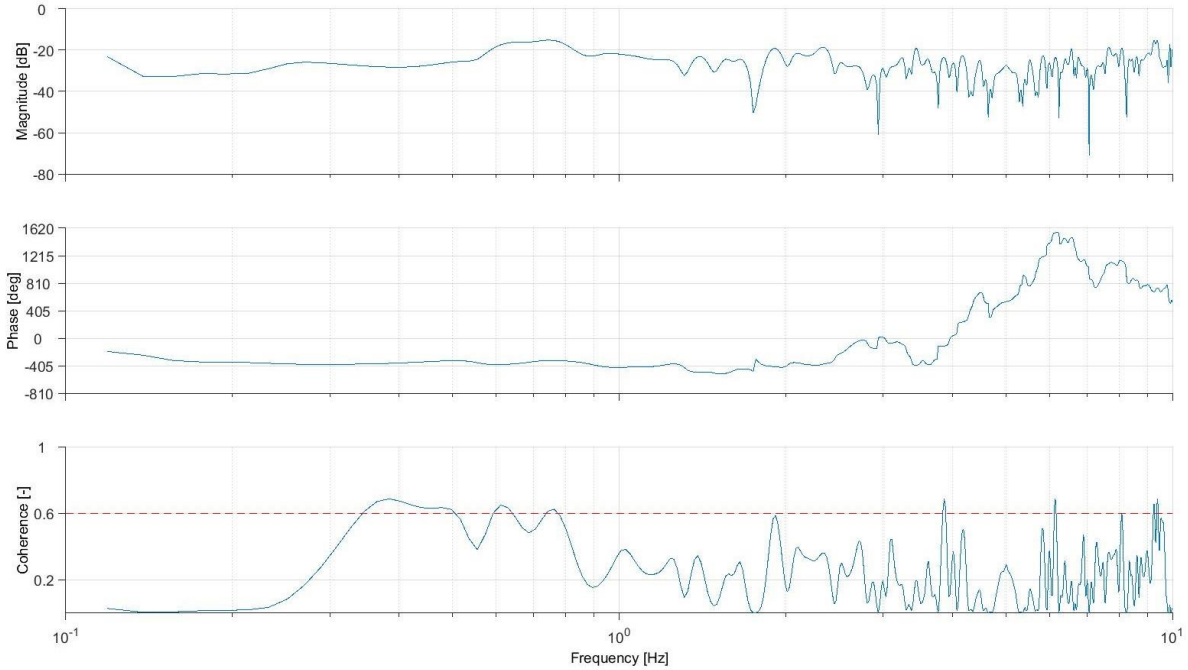


Figure 4.43: Bode plot of forward acceleration response relative to throttle setting

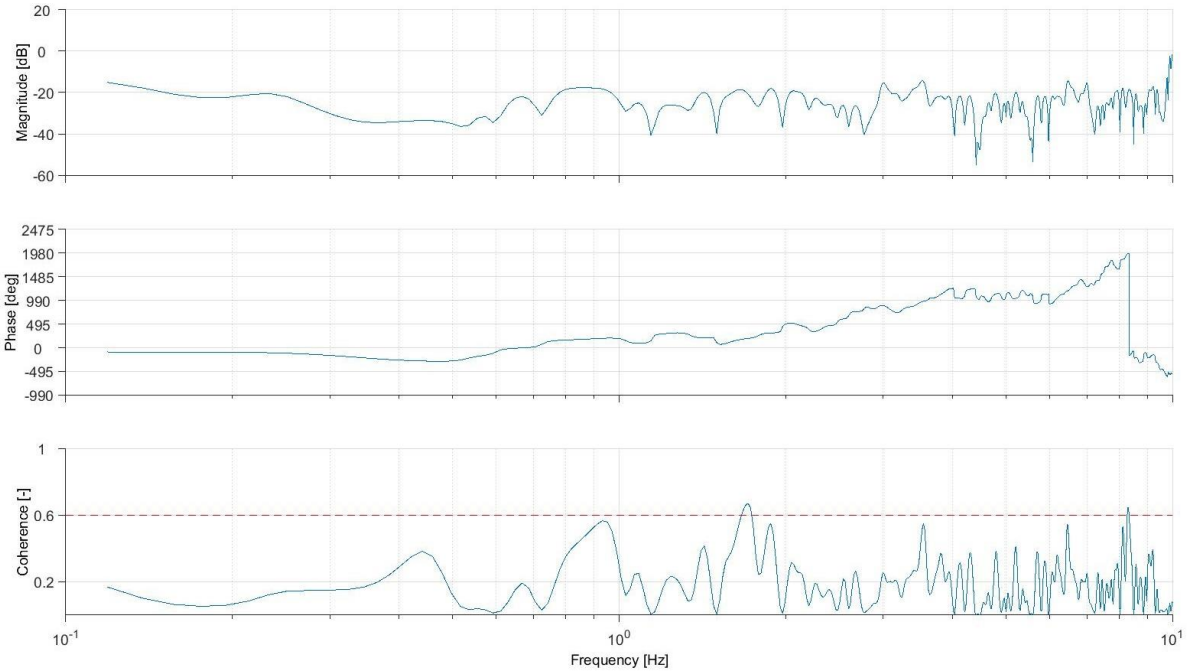


Figure 4.44: Bode plot of downward acceleration response relative to throttle setting

4.2.3 Bandwidth

The bandwidth criterion is examined as explained in method section, this calculation is performed in the Bandwidth module in CIFER. The response of the pitch to the elevator deflection is used for the analysis. The ω_{180} is at 5 Hz, which is shown as the middle line in the phase plot (Figure 4.45) and more closely calculated in Figure 4.46. The corresponding magnitude of that phase is estimated to be around -16 degrees and therefore the $\omega_{WB_{gain}}$ is found at a magnitude of -10 degrees. This is shown in Figure 4.45 as the grey dashed line in the magnitude plot.

The ω_{135} is found following the -135 degrees phase line in the phase plot (Figure 4.45). The $\varphi_{2\omega_{180}}$ is found at twice the ω_{180} which is around 10 Hz. This number is estimated from the slope of Figure 4.46. The normal response is not trusted anymore at these high frequencies and therefore the slope is used instead. With all these variables the bandwidth frequency and time delay are estimated. The phase delay is calculated according to equation 3. The bandwidth is 8.83 rad/s and the phase delay is 0.02s, the criteria can be rated according to Figure 1.4. All of this is summarized in Table 4.1.

Table 4.1: Results bandwidth criterion

Description:	Parameter:	Result:	Unit
6-db gain margin frequency	$\omega_{WB_{gain}}$	1.4	Hz
		8.8	rad/s
Frequency at -180 deg phase	ω_{180}	5.1	Hz
		32	rad/s
Frequency at -135 deg phase	ω_{135}	2.9	Hz
		18	rad/s
Phase at double of the 180 deg phase	$\varphi_{2\omega_{180}}$	-250	deg
Bandwidth frequency	ω_{WB}	1.4	Hz
Phase delay	τ_p	0.02	s

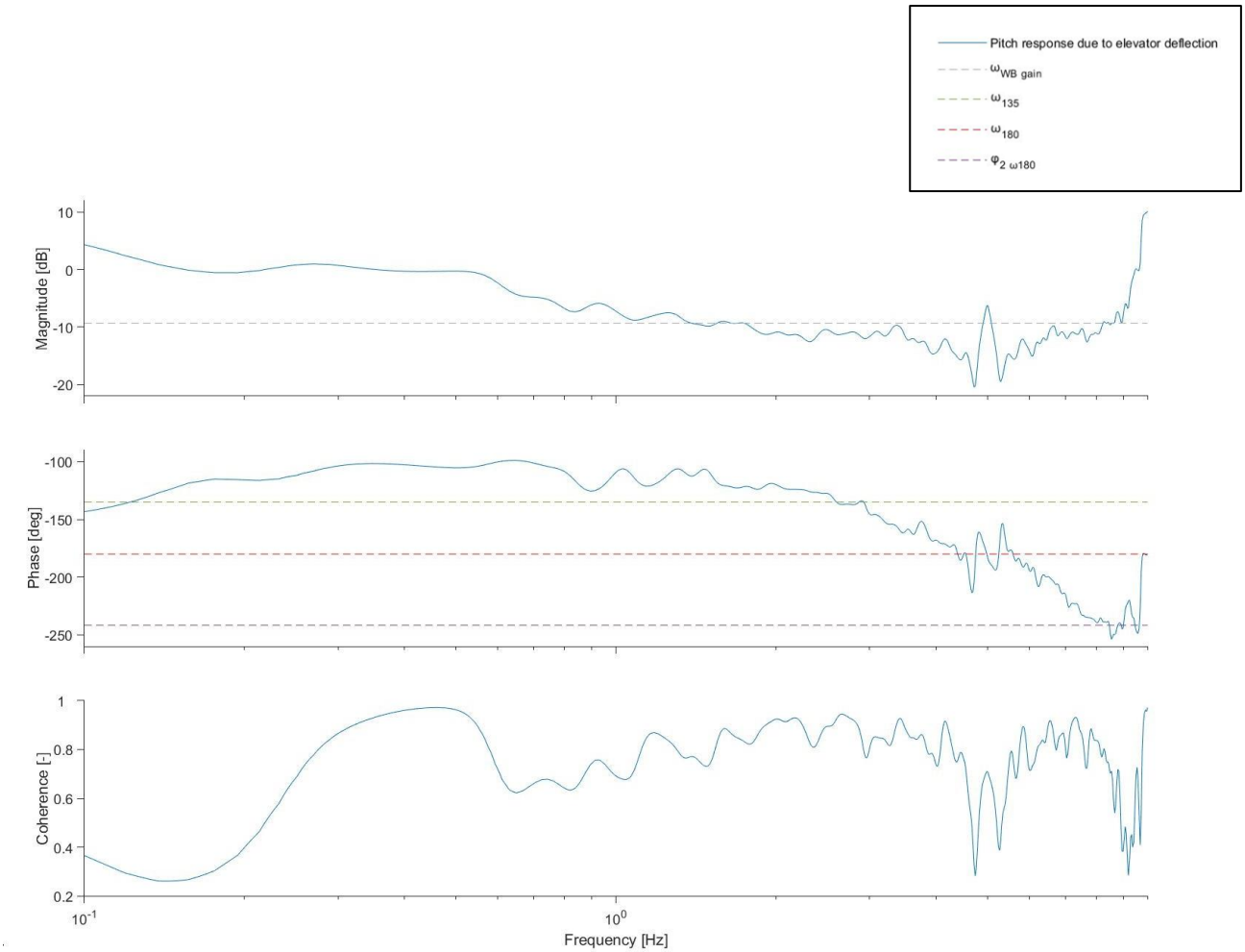


Figure 4.45: Bandwidth criterion

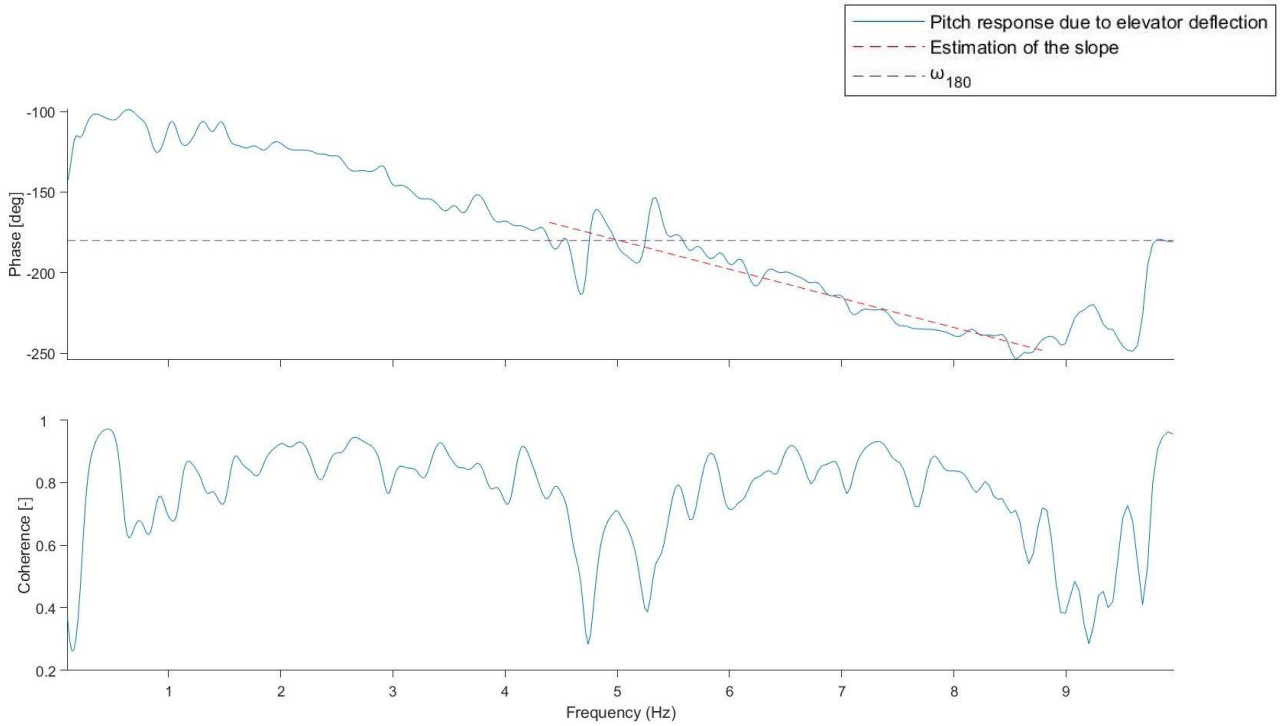


Figure 4.46: Estimation of bandwidth parameter

4.2.4 CAP

Three lower order transfer functions are estimated from the pitch, pitch-rate and normal acceleration response to the elevator input. The short period damping, frequency and the parameters to calculate the CAP can be extracted from these transfer functions.

First, the LOES accelerometer response is investigated. The fit of the transfer function on the bode plot is shown in Figure 4.47. The solid line depicts the flight test results and the dashed line depicts the estimated function. The cost function is below 100, which means that the fit is of acceptable quality. The frequency and damping of the short period are fixed for the other transfer functions because these parameters can not change within the same test setup (Table 4.2). Figure 4.48 shows the fit of the pitch speed response and Figure 4.49 shows the fit of the pitch angle response. The cost functions of these responses are below 100 and therefore these fits are of acceptable quality (Table 4.2). The parameters of the transfer functions can be found in Table 4.2.

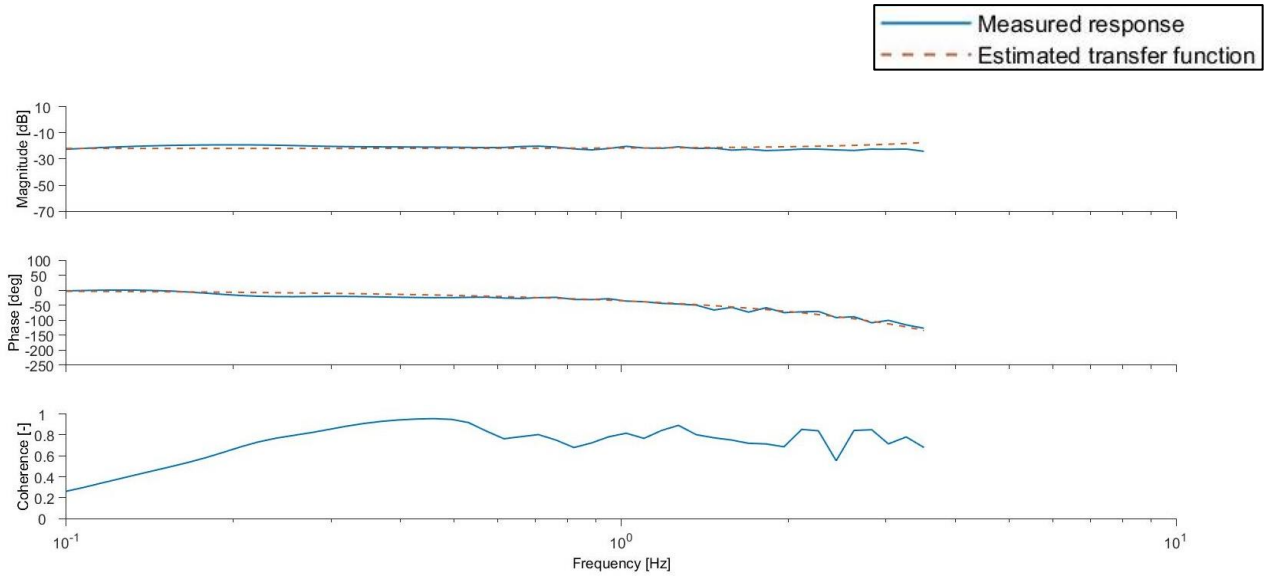


Figure 4.47: Fit of the LOES accelerometer in upward direction response relative to elevator deflection

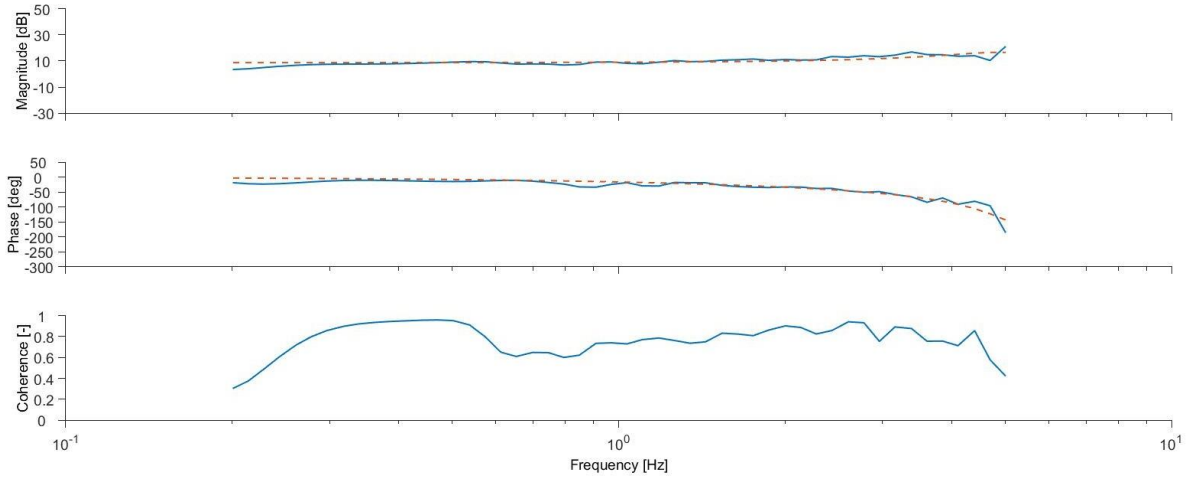


Figure 4.48: Fit of the LOES pitch speed response relative to elevator deflection

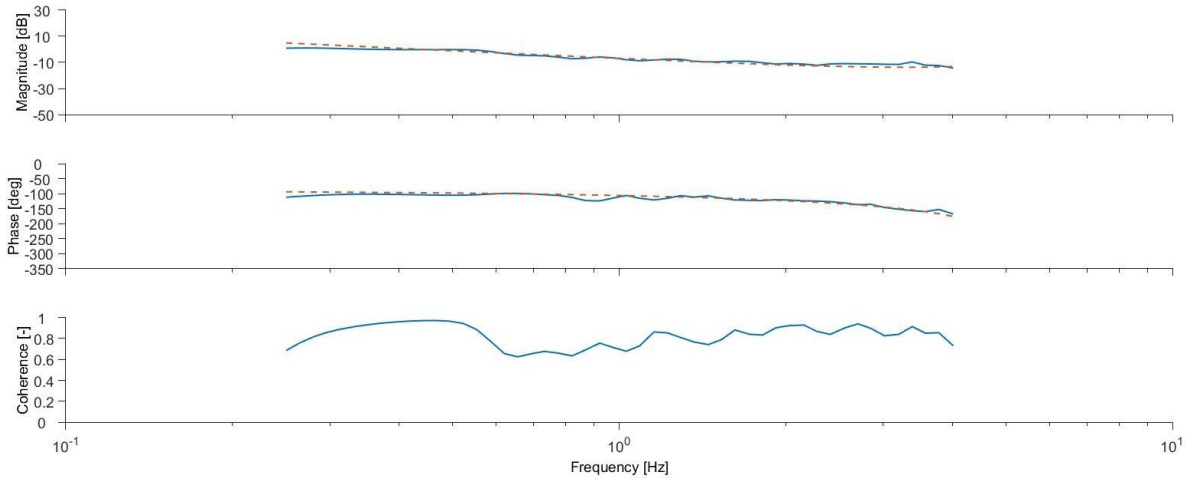


Figure 4.49: Fit of the LOES pitch angle response relative to elevator deflection

The total pitch speed response can be investigated as well (Figure 4.50). The parameters corresponding to the transfer functions are summarized in Table 4.2. Here, only the pitch speed is investigated, the pitch angle could be extracted from this as well, but this will not give more information.

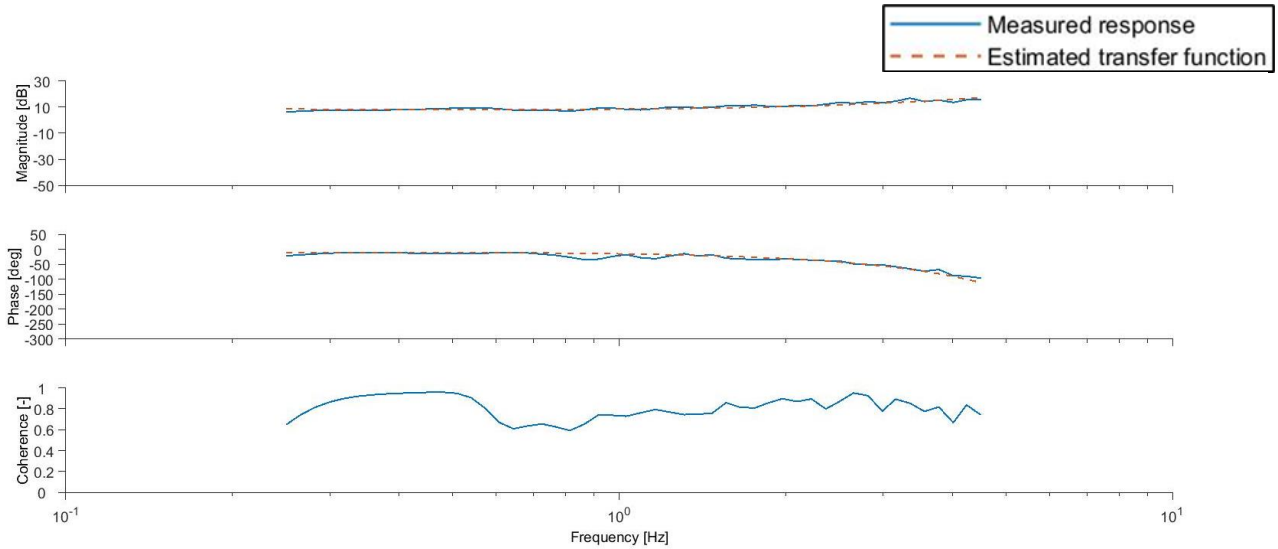


Figure 4.50: Transfer function fit of pitch speed response

Table 4.2: SISO response investigation parameters

Parameter:	LOES results:	Results	Unit:
$N_{z_{\delta_e}}$	81.9	-	g/deg
K_q	-	180	-
T_{θ_1}	-	17	s
T_{θ_2}	0.0025	0.71	s
M_{δ_e}	7.0	-	deg
τ_n	-0.08	-	s
τ_e	-0.03	-0.07	s
ζ_{sp}	0.21	0.32	-
ω_{sp}	32	36	rad/s
ζ_p	-	0.66	-
ω_p	-	0.32	rad/s
J of $\frac{N_z(s)}{\delta_e}$	73	-	-
J of $\frac{q(s)}{\delta_e}$	69	30	-
CAP1	1.40	1.78	rad·g/s^2
CAP2	1.46	-	rad·g/s^2

4.2.5 MIMO results

In this section, the state space system is estimated, this calculation is performed in DERIVID a module in CIFER. The quality of SISO responses for the throttle are not satisfactory, therefore, these are excluded from this analysis and a model without the influence of the throttle is estimated. In the simulation a similar model is estimated.

The response for the forward velocity has a low section of coherence, the frequency range is below 0.3 Hz, which is too low to use in the simulation. Therefore, this response is not used in the optimization problem. This is why the top row of the state space matrix is meaningless and no conclusions should be drawn from that.

The transfer functions which correspond to the state space system are shown in Table 4.3. The complex eigen values, the phugoid and the short period, can be seen clearly. The phugoid is possibly incorrect, because an inadequate number of low frequencies are taken into account in the flight test to see the full frequency spectrum of the phugoid.

The fit of the three acceptable responses are shown in Figure 4.51-Figure 4.53. These fits are overall quite decent, but unfortunately the downward speed response cost function is above 100, although it is not extremely divergent. The forward speed response is taken out of the optimization and therefore it is not included in the results.

The CAP is calculated with the incidence lag constant derived from the LOES model, because this term is often represented inadequately in higher order models [32] and this seems to be the case in this optimization. The incidence lag constant calculated in the previous section is trusted because the CAP is calculated with two methods and both values are close to each other. The comparison of these results to the military criteria, shows that the CAP is level 1, phugoid damping is level 1 and the short period damping is level 2. The derived state space system is shown in Table 4.4.

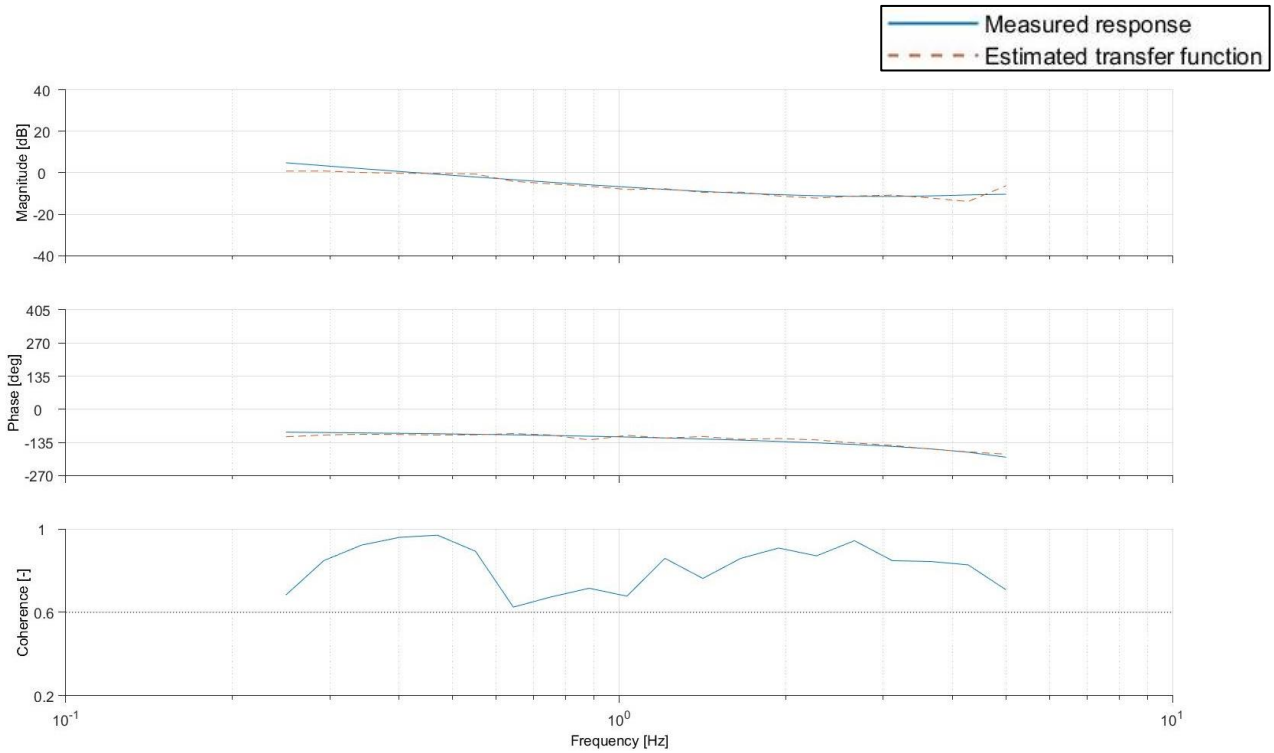


Figure 4.51: Fit over pitch response relative to elevator deflection

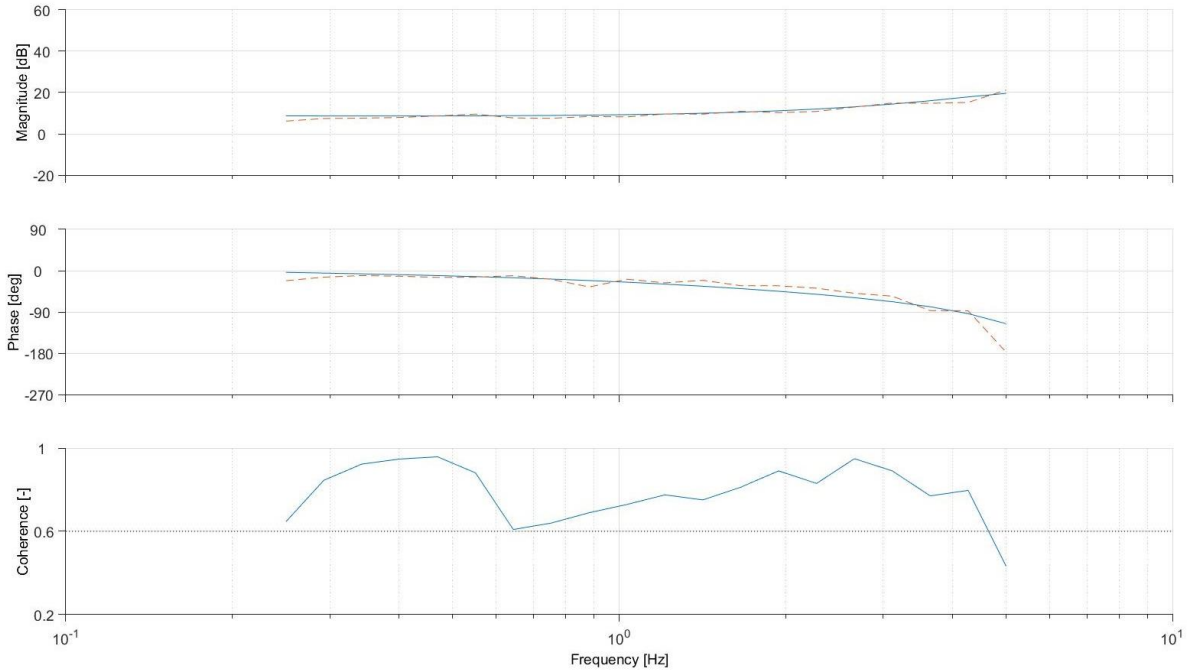


Figure 4.52: Fit of pitch speed response relative to elevator deflection

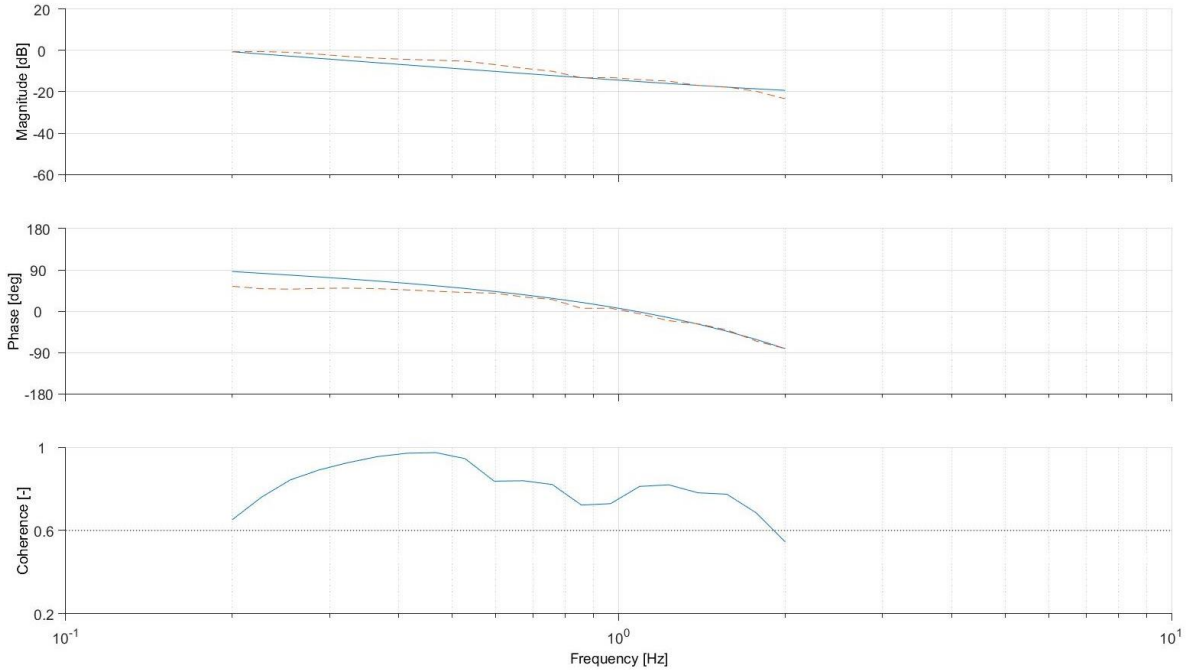


Figure 4.53: Fit of downward speed response relative to elevator deflection

Table 4.3: MIMO transfer function parameters

Parameter:	Results	Unit:
K_w	44.5	-
K_q & K_θ	-209	-
T_{θ_1}	-16.3	s
T_{θ_2}	0.35	s
T_{w_1}	-33.4	s
τ_w	-0.18	s
τ_e	-0.01	s
ς_{sp}	0.32	-
ω_{sp}	36	rad/s
ς_p	0.66	-
ω_p	0.32	rad/s
J of $\frac{q(s)}{\delta_e}$	60	-
J of $\frac{\theta(s)}{\delta_e}$	63	-
J of $\frac{w(s)}{\delta_e}$	143	-
CAP	1.78	rad·g/s^2

Table 4.4: MIMO state space system results

Parameter:	Results:	Unit:	Cramér-Rao:	Insensitivity :
X_u	0	1/s	-	-
X_w	0	1/s	-	-
X_q	0	m/s	-	-
X_θ	0	m/s^2	-	-
X_{δ_e}	68.9	m/s^2	-	-
X_{δ_T}	-	m/s^2	-	-
Z_u	0.14	1/s	-	-
Z_w	0.08	1/s	-	-
Z_q	-0.12	m/s	-	-
Z_θ	0	m/s^2	-	-
Z_{δ_e}	-0.09	m/s^2	-	-
M_u	49.2	1/s^2	-	-
M_w	94.3	1/s^2	-	-
M_q	20.5	1/s	-	-
M_{δ_e}	11.5	1/s^2	-	-
M_{δ_T}	-	1/s^2	-	-
m	16.5	kg	-	-
$Z_{\dot{w}}$	16.2	1/s^2	-	-
$M_{\dot{w}}$	72.4	1/s	-	-
I_{yy}	0.029	kg/m^2	-	-
τ_{e_w}	0.18	s	-	-

4.2.6 Repeatability of the data

The results of the measurement data can be checked by verifying if the frequency responses for the different sweeps are similar. In the case that the separate measurements show the same results, the variable measurement error is low. The frequency responses instead of the time responses are compared to each other, because every sweep is different in length and frequency build-up. Figure 4.54-Figure 4.57 show the bode and coherence plots for all three sweeps and all responses.

It can be seen clearly that for all plots the lines are close together when the coherence is high. When the coherence is lower, the three sweeps start to deviate from each other. This is what is expected from a good measurement. This indicates that the repeatability of the test is good when the coherence is acceptable, despite the fact that the input signal is always a bit different due to the manual frequency increase. The other responses show similar behavior.

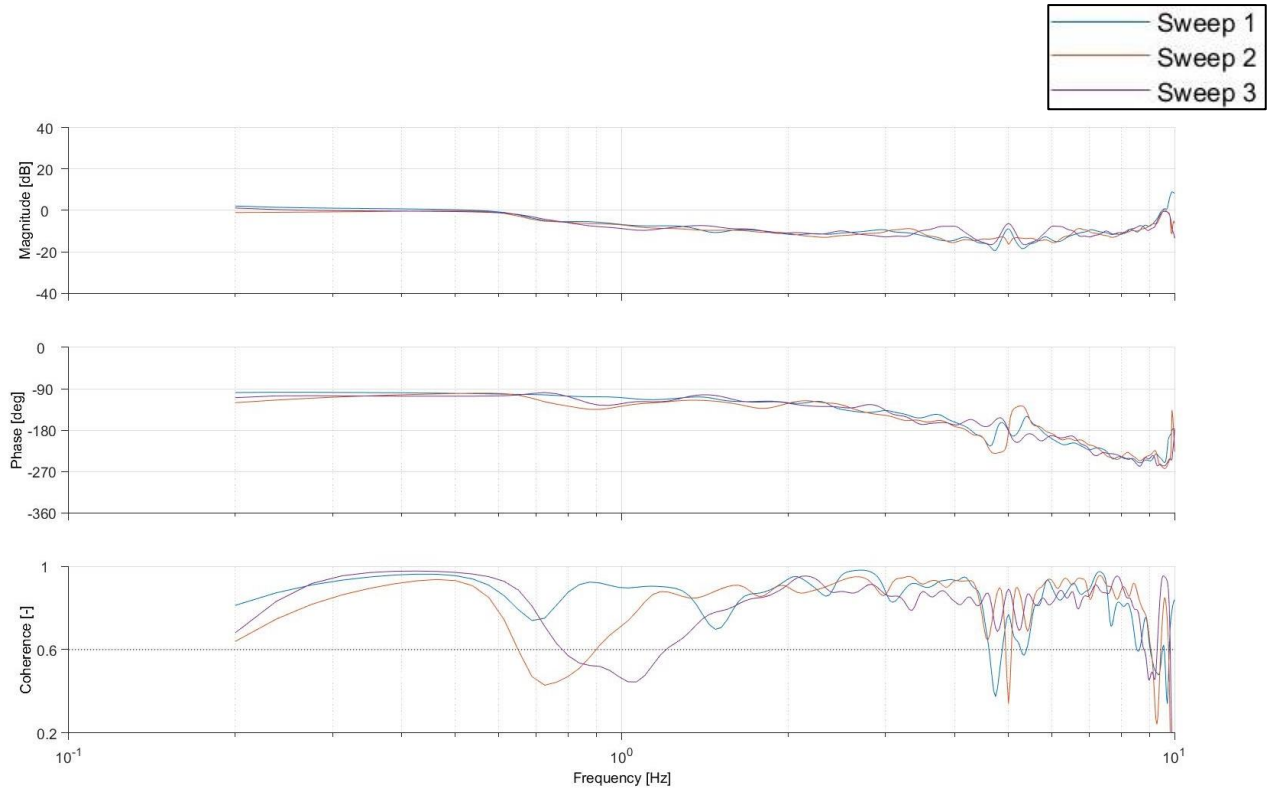


Figure 4.54: Repeatability of sweeps for the pitch response relative to elevator deflection

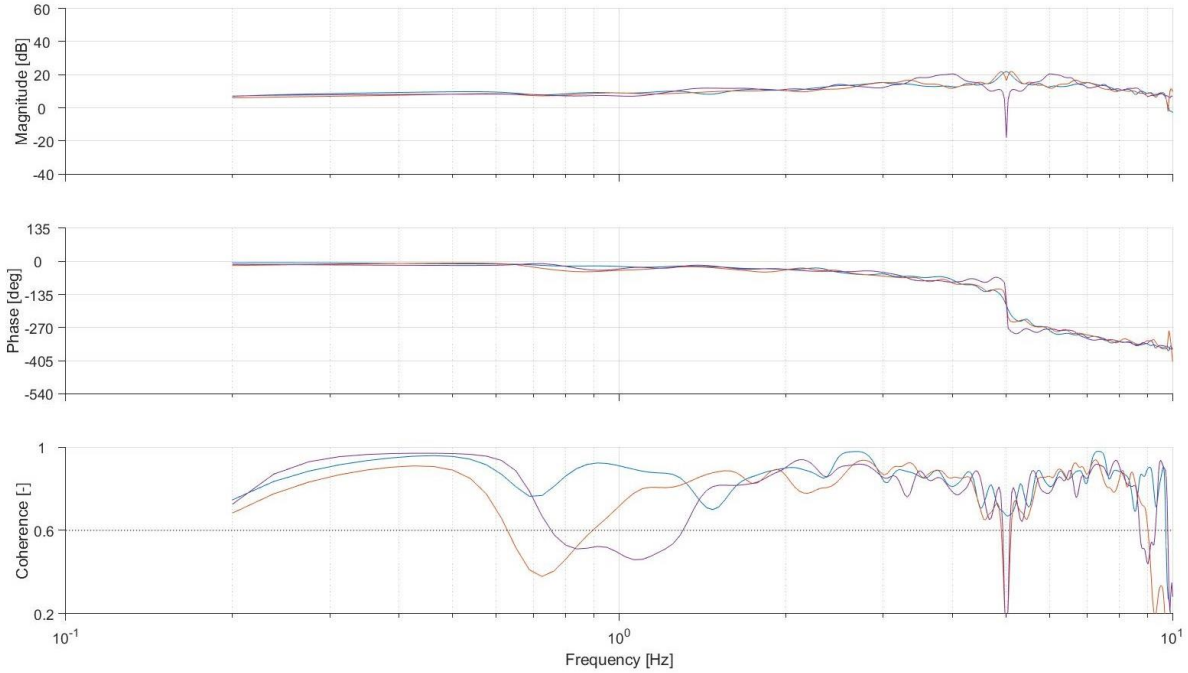


Figure 4.55: Repeatability of sweeps for the pitch speed response relative to elevator deflection

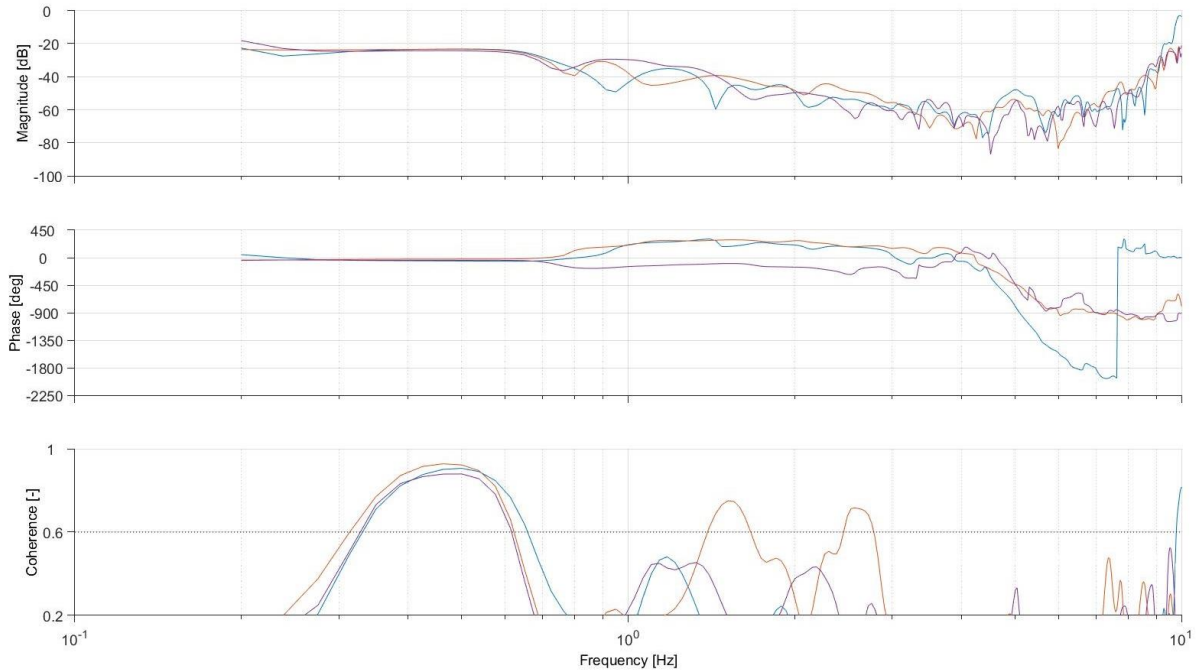


Figure 4.56: Repeatability of sweeps for the forward speed response relative to elevator deflection

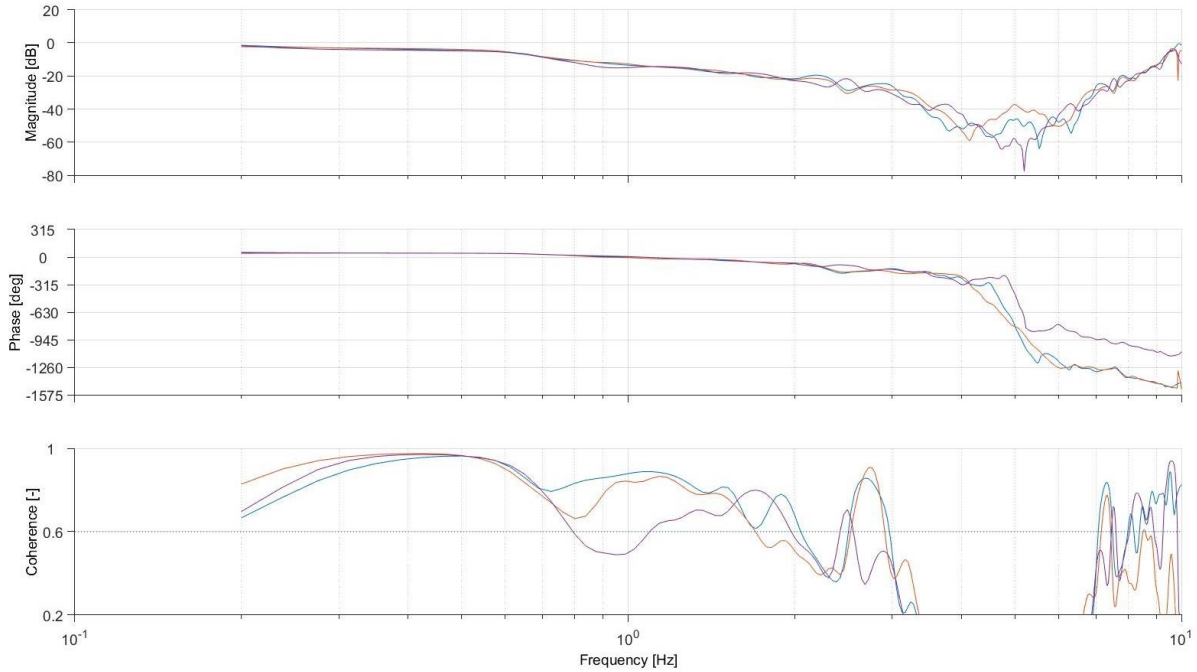


Figure 4.57: Repeatability of sweeps for the downward speed response relative to elevator deflection

4.2.7 Consistency of the data

The consistency of the measurements can be checked in the frequency domain. In the frequency domain the derivative of a transfer function can be calculated with a factor of s. This can be seen in the transfer function of the pitch and pitch speed response, as the factor in between them is 1/s. This is can be done with more parameters as well: the forward speed and the forward acceleration and downward speed and downward acceleration.

Figure 4.58 shows these plots for the pitch and pitch speed. These two functions clearly lay on top of each other at the place where the coherence is acceptable. This means that the measurements are consistent and therefore the results from the pitch and pitch speed can be trusted,

except for low and very high frequencies because the coherence at those frequencies is not acceptable.

The forward speed (Figure 4.59) has a coherence that is too low to be meaningful, therefore these plots are not used in the investigations. The downward speed (Figure 4.60) shows a consistency issue in the phase plot: There is clearly an offset between the two signals. This can have multiple reasons, but an error in the units is unlikely because it would be strange to for example to measure the speed in m/s and the acceleration in ft/s^2. This suggests that the two measurements are not consistent.

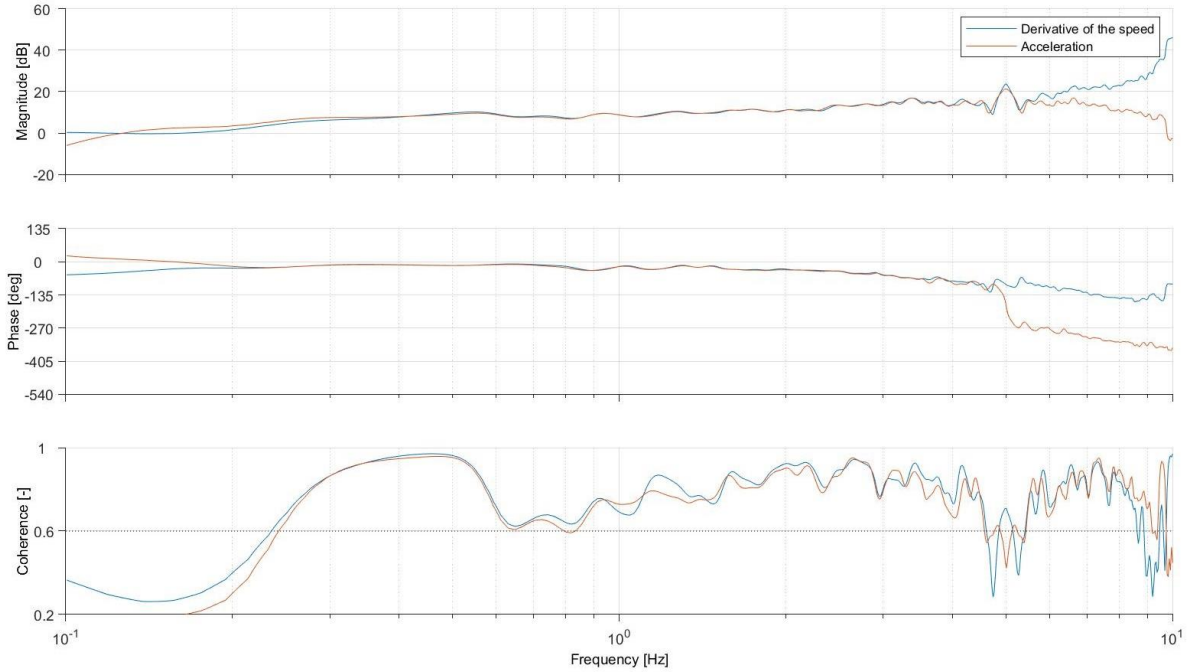


Figure 4.58: Derivative of the pitch angle response and the pitch speed response

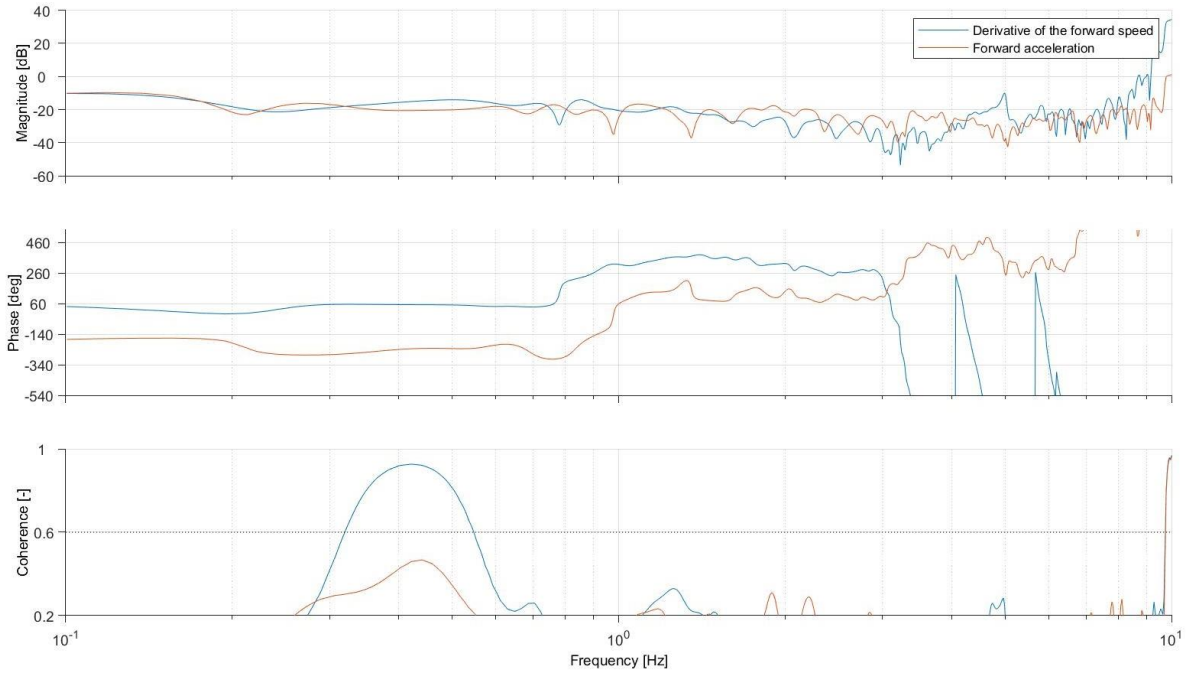


Figure 4.59: Derivative of the forward speed response and the acceleration response

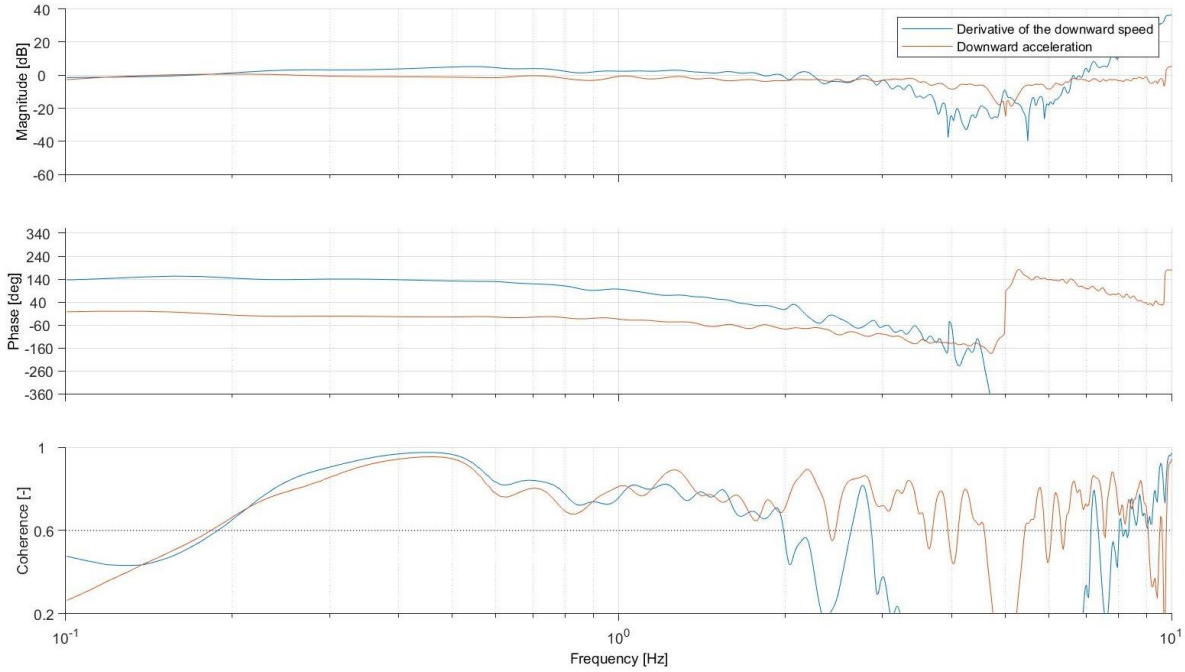


Figure 4.60: Derivative of the downward speed and the acceleration response

4.3 Simulation

The preliminary results of the simulation consist of the four figures below. They are generated for a speed of 18 m/s at the altitude of 30 meters with the corresponding air properties, as explained in the method. These figures are the lift curve slope, lift to drag curve, pitching moment curve and the drag polar.

The lift curve (Figure 4.61a), is expected to be a straight line with a positive slope and a decrease in slope at the higher angles of attack. In this case, the maximum angle of attack is not high enough to see the slope decrease and therefore it is only a straight line. The lift to drag (L/D) curve (Figure 4.61b) is, as expected, a straight line with a positive slope with a maximum whereafter the L/D starts to decrease. The pitching moment (Figure 4.62a) has a negative slope for a stable aircraft and the drag polar (Figure 4.62b) is a c-shaped curve as would be expected as well.

All of this together seems reasonable, also when it is compared to other analyses, as is done in Figure 4.61-Figure 4.62, where the Skysurfer is compared to the Drone Identity[38], Skynet [39] and Fregata [40]. These links show the results of a project at Stanford University in which students model a Bixler aircraft in the same simulation program. The Bixler is very similar to the Skysurfer X8 and therefore this information can be used to check the results of this simulation.

In the simulation, the fuselage and the propeller influence are not included in the analysis. This can be seen in the lift to drag curve which shows a high L/D of above 20, which is unrealistically high for such an aircraft. By neglecting the fuselage and the propeller, the drag is not estimated correctly. Another cause of the low drag estimation is that the simulation is unable to estimate viscous drag, the boundary layer and separation of the flow properly.[41] [58]

[38] https://sites.google.com/a/stanford.edu/droneidentity/problem-set-2
[39] https://skynet241x.wordpress.com/
[40] http://fregata-uav.weebly.com/uploads/7/7/5/8/77584296/problemset2.pdf
[41] http://www.xflr5.tech/docs/Part%20IV:%20Limitations.pdf

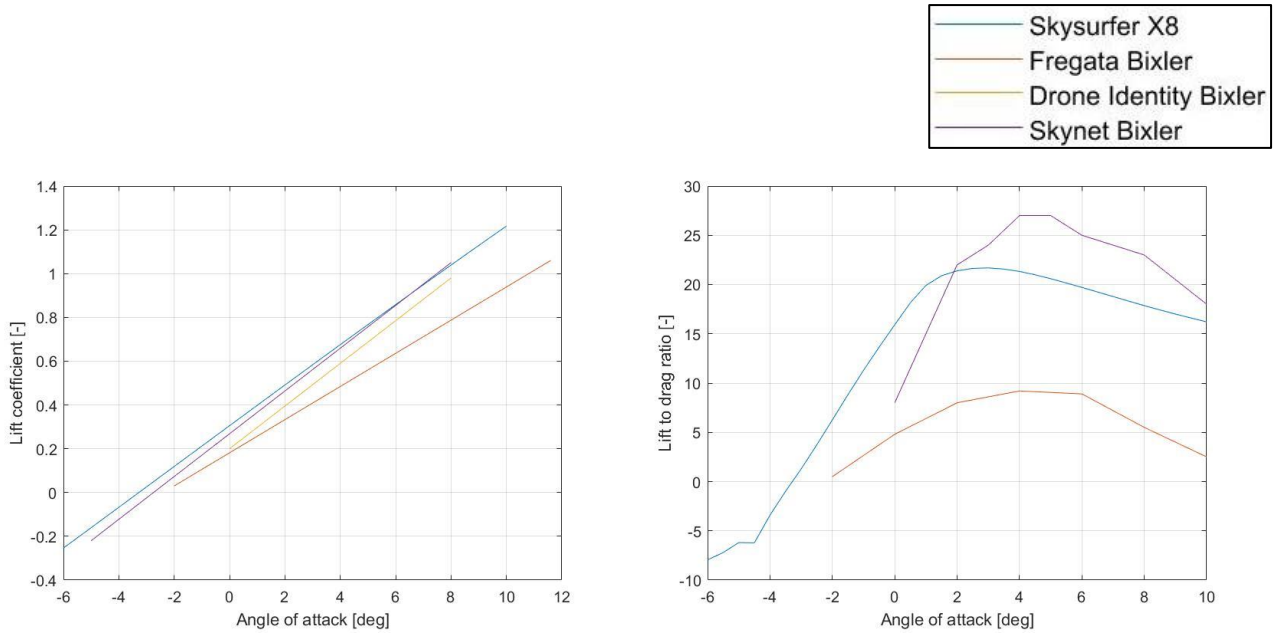


Figure 4.61: a) Lift coefficient vs. angle of attack b) Lift drag ratio vs. angle of attack

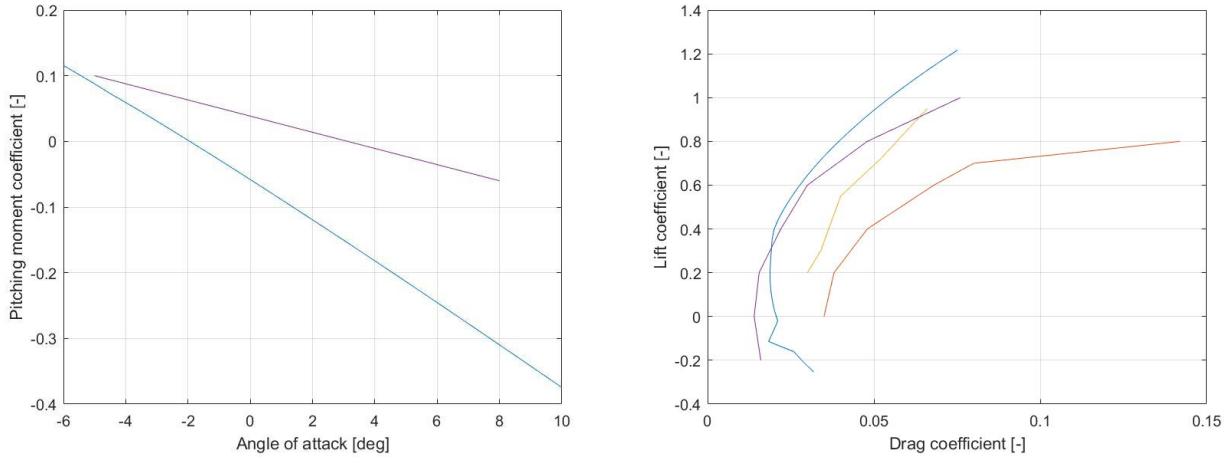


Figure 4.62: a) Pitching moment coefficient vs. angle of attack b) Drag polar

4.3.1 Transfer function and state space function model

The results of the analysis are shown in this section. Table 4.5 contains the phugoid, short period motion and CAP and these parameters found in existing literature for similar aircraft. The state space parameters are summarized in Table 4.6 and the transfer function parameters are summarized in Table 4.7. From this state space system, the transfer functions for the SISO systems can be divided and these systems can be compared with the flight results. Figure 4.63-Figure 4.66 shows the bode plots containing the flight test results and the simulation results.

Table 4.5: Handling characteristics from simulation and other research

Parameter:	Skysurfer:	Fregata Bixler[40]:	Skynet Bixler[39]:	Unit:
ω_p	0.46	0.69	1	rad/s
ς_p	0.09	0.001-0.05	0.02	-
ω_{sp}	35.7	17.8-18.0	33	rad/s
ς_{sp}	0.68	0.65-0.66	1.7	-
CAP	2.60	-	-	rad·g/s^2

Table 4.6: Longitudinal state space parameters of the simulation

Parameter:	Result:	Unit:
X_u	-0.08	1/s
X_w	0.5	1/s
X_q	0	m/s
X_θ	-9.81	m/s^2
X_{δ_e}	2	m/s^2
Z_u	-0.85	1/s
Z_w	-18.3	1/s
Z_q	21.2	m/s
Z_θ	0	m/s^2
Z_{δ_e}	-444	m/s^2
M_u	-0.094	1/s
M_w	-34.1	1/s
M_q	-30.2	1/s
M_{δ_e}	-823	1/s

Table 4.7: Transfer function parameters of the simulation

Parameter:	Result:	Unit:
K_u	1.97	-
K_w	-444	-
$K_q \& K_\theta$	-823	-
T_{u_1}	0.15	s
T_{u_1}	27.4	s
T_{u_1}	-93.1	s
T_{w_1}	69.6	s
T_{θ_1}	0.00365+0.580i	s
T_{θ_2}	0.00365-0.580i	s
ζ_{sp}	0.68	-
ω_{sp}	35.7	rad/s
ζ_p	0.09	-
ω_p	0.46	rad/s
CAP	2.6	rad·g/s^2

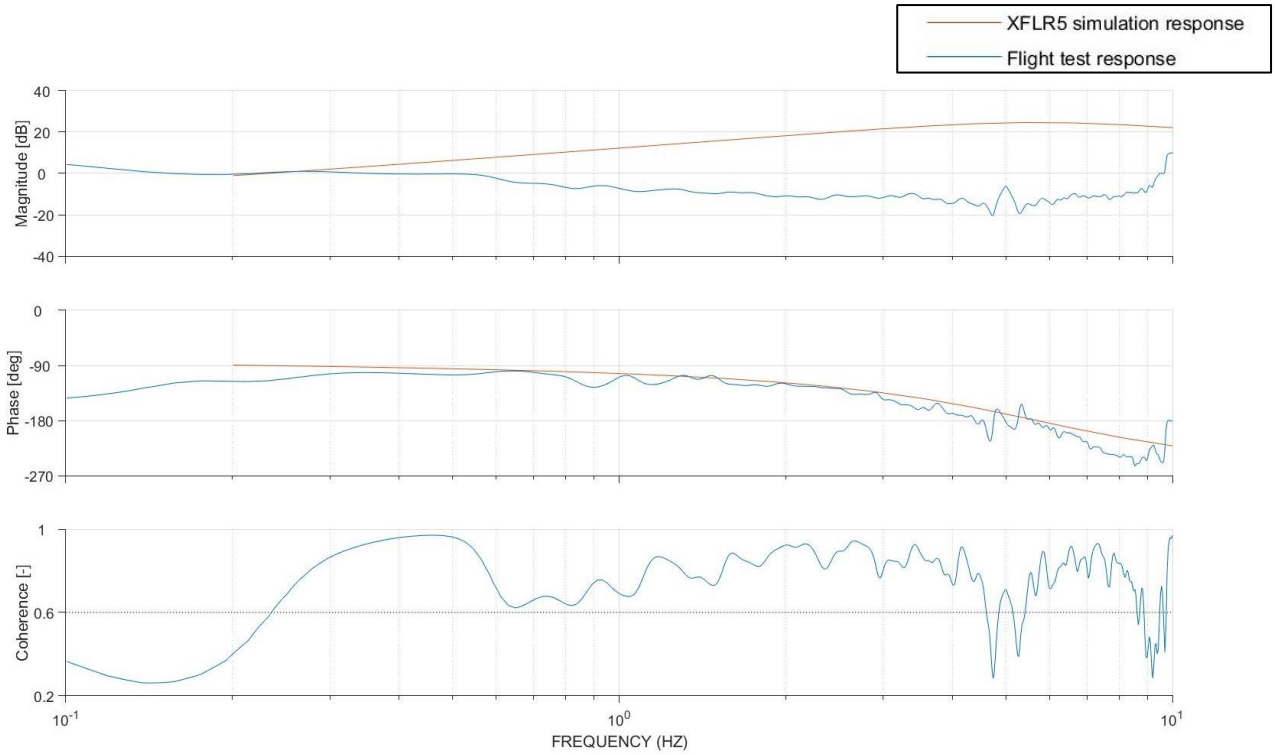


Figure 4.63: Bode plot of pitch response relative to elevator deflection

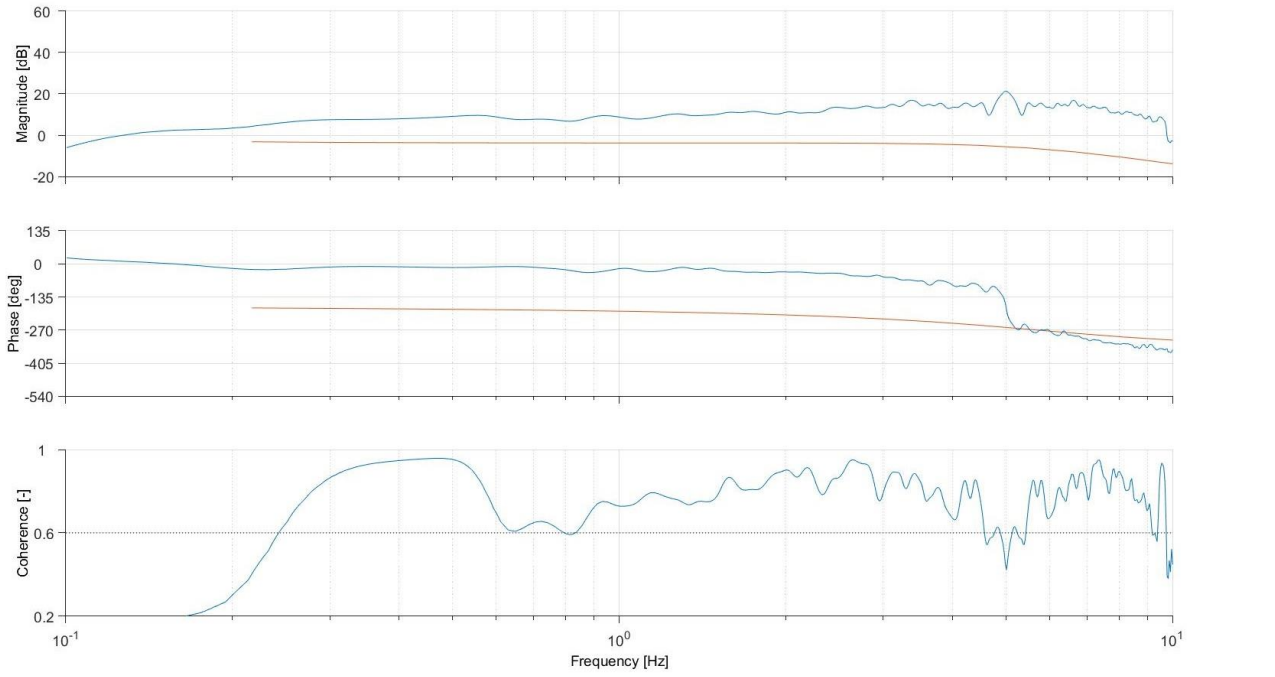


Figure 4.64: Bode plot of pitch speed response relative to elevator deflection

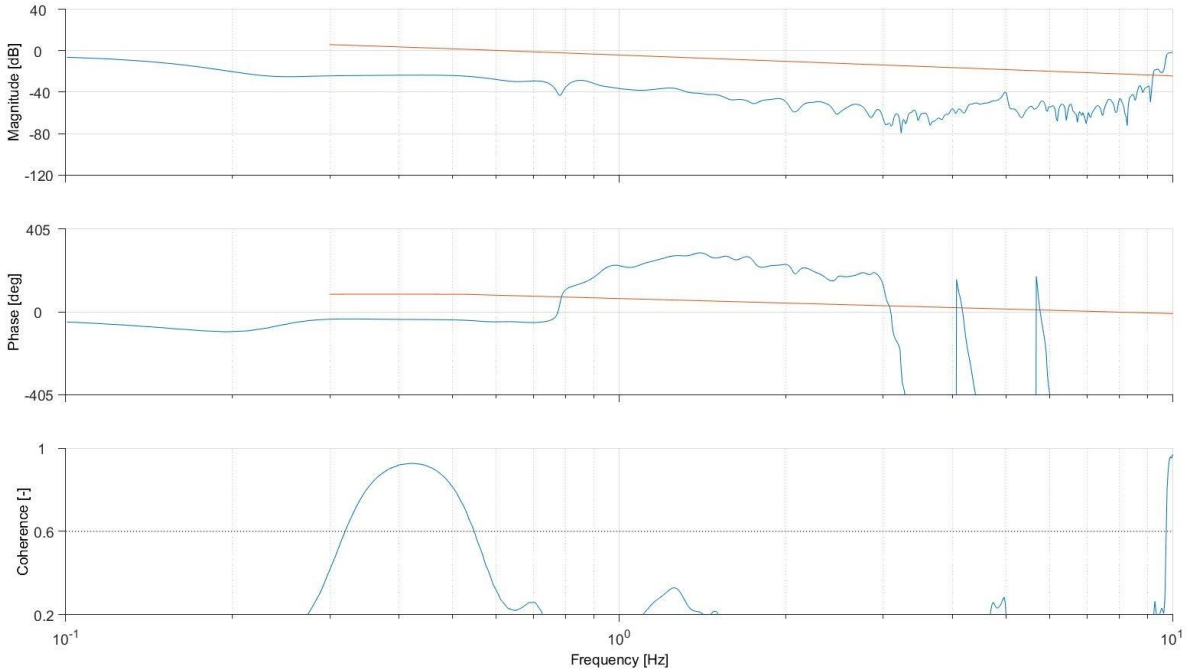


Figure 4.65: Bode plot of forward speed response relative to elevator deflection

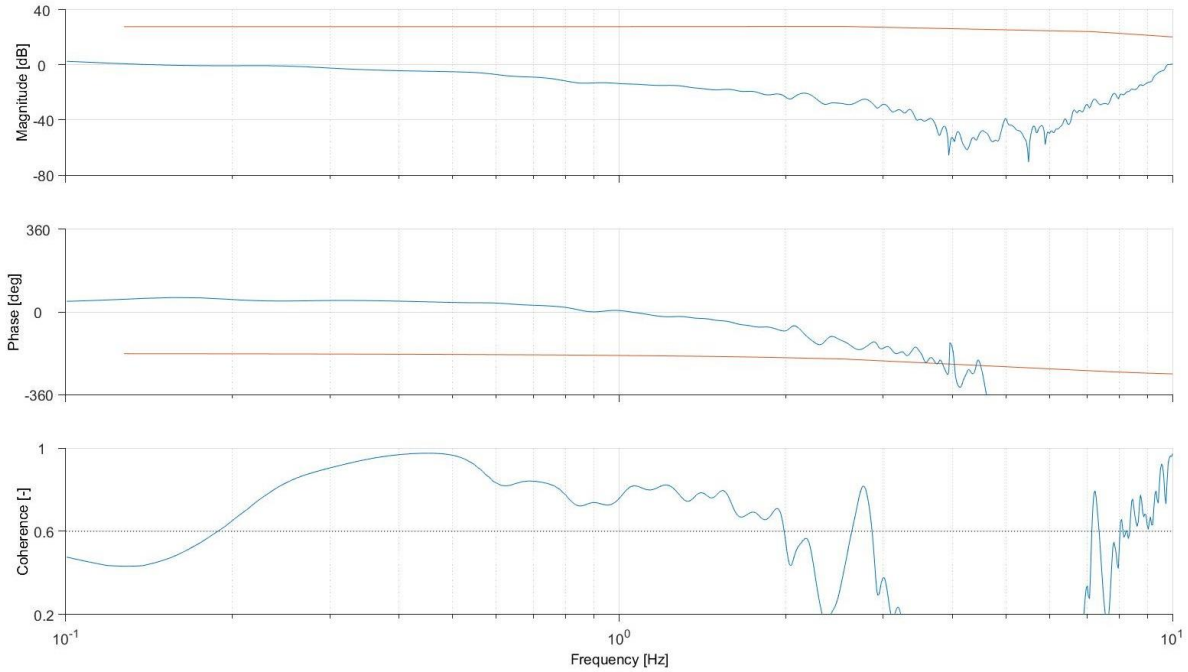


Figure 4.66: Bode plot of downward speed relative to elevator deflection

4.4 Summarized results

The results gathered during these experiments and simulation differ quite a lot from each other as can be seen in Table 4.8. The SISO and MIMO results are partly equal because they are estimated from the same bode plot and have the same transfer function. The LOES approximation differs from these two results because the estimated system is different. The simulation differs from the flight test results because assumptions are made during this process. The center of gravity is estimated, the moments of inertia are estimated, the flight conditions are estimated and the largest influence are the propellers which are not taken into account. A propeller does not only create a thrust force in the forward direction, there is also an upward force generated which is perpendicular to the thrust force.

This force generates a nose upwards moment, which affects the angle of attack response and therefore affects the pitch angle/pitch speed response as well. [59] Because this moment is nose up, this will destabilize the aircraft. This is what can be seen in the short period, this motion is less stable in the flight test than in the simulation, which could explain the lower damping ratio for the flight test compared to the simulation.

The influence on the phugoid motion is different because this mode does not have a major effect on the angle of attack. In this case, the propeller dampens the phugoid motion. During the phugoid motion the thrust generated by the propeller dampens the motion. When the flight speed is decreasing, the thrust force increases. When the speed increases the thrust force decreases.[42][43] This is what is shown in the results, the damping ratio of the phugoid is lower for the simulation than for the flight tests.

However, the frequency of the phugoid motion is low and therefore the coherence at these frequencies is also low because the start frequency of the sweep is above the phugoid frequency. Therefore, motion is not on the bode plots, and these results are not as accurate as the short period motion (which is in the end of the bode plots but has an acceptable coherence). At the MIMO analysis, the signal is used up to 32 rad/s because of the sudden drop in coherence.

Taking these remarks in consideration, the MIMO results are the most trusted, especially the short period data and related handling criteria. The estimation of the phugoid gives an indication of this value, however from a handling characteristic perspective, the most important constraint is that the damping of the phugoid is positive. The results show that it is highly likeable that this is the case.

Table 4.8: Summarized results

Parameter:	LOES:	SISO:	MIMO:	Simulation	Unit:
ω_p	-	0.32	0.32	0.46	rad/s
ζ_p	-	0.66	0.66	0.09	-
ω_{sp}	32.0	36.0	36.0	35.7	rad/s
ζ_{sp}	0.21	0.32	0.32	0.68	-
CAP1	1.40	1.78	1.78	2.60	rad·g/s^2
CAP2	1.46	-	-	-	rad·g/s^2

The results are compared to other aircraft as shown in Table 4.9. For the short period, the common frequency of the other aircraft is 16 rad/s and the damping is between 0.7 and 0.8. This is a huge difference compared to the results from the Skysurfer. The models all share some common features, but also differ in some other features for example: Engine position, size, weight, center of gravity and flight conditions. And despite these differences, the results among these aircraft are still comparable.

When these aircraft are compared to the Skysurfer, there is one big difference that stands out, which is the engine setup. None of the investigated aircraft have a dual engine setup on the main wing and their flight speed is often different as well. This could influence the results, but also the center of gravity would be of great interest in this comparison because that has major influence on the stability of the aircraft. Unfortunately, the center of gravity positions of these aircraft are not stated in their papers and therefore this can not be taken into account during the comparison. Another aspect which could influence the results, is the placement of the measurement units in respect to the center of gravity. If the measurements are not taken in the center of gravity the speed components are affected by an angle replacement. Some researchers took this into account for the analysis, but this

[42] http://www.flightlab.net/Flightlab.net/Download_Course_Notes_files/7_LongitudinalDynami%232BA157.pdf

[43] https://www.electricrcaircraftguy.com/2013/09/propeller-static-dynamic-thrust-equation.html

is not done for the Skysurfer. The distance between the flight controller and the center of gravity in the Skysurfer is 81mm, but this should only have a small or neglectable influence on the results.

Table 4.9: Handling characteristics similar aircraft

Parameter:	Dynam HawkSky [33]:	Cesna model [36]:	Ultrastick 120 [51]:	Bixler 2 [60]:	Unit:
ω_p	-	-	0.51	-	rad/s
ζ_p	-	-	0.38	-	-
ω_{sp}	16.8	12.5	16.3	16.3	rad/s
ζ_{sp}	0.70	0.68	0.83	0.83	-

5 Conclusion

The handling characteristics of the Skysurfer X8, which are investigated in this research, are all level 2 according to the MIMO results and the simulation for the CAP, phugoid damping and short period damping. Therefore, it can be concluded that the handling characteristics are satisfactory, keeping in mind that the handling criteria are very strict for smaller unmanned aircraft compared to manned aircraft. The short period damping results of the LOES system, are level 3. The system derived in the LOES approximation has a higher cost function than the same input-output relation in the MIMO analysis. The short period damping ratio in the MIMO analysis is extracted out of more input-output relations and therefore contains more internal validation. This indicates that the LOES optimization found an optimum which is not the actual optimum and the MIMO analysis is more reliable.

The transfer function models are derived for the LOES model, SISO model and the MIMO model from the flight data and the simulation. In the MIMO analysis and the simulation, the transfer functions for the forward speed, downward speed, pitch angle and pitch speed relative to elevator deflection are derived. In the LOES model the transfer functions for the pitch angle/pitch speed and normal accelerometer in upward direction relative to elevator deflection are derived. In the SISO analysis the transfer function for the pitch angle and pitch speed are derived. In the MIMO analysis and the simulation, the state space system for the longitudinal dynamics is derived.

The accuracy of the transfer functions derived from the flight test is indicated by the cost function of the fit, the coherence of the response and the consistency of the measurements. The pitch angle and pitch speed responses derived from the MIMO analysis are the most accurate and most trusted, because the cost function is low, the coherence is above 0.6 for a large frequency range and the measurements are consistent with each other. The SISO responses show comparable results, this is expected because the same shape of transfer function is estimated from the same response. The LOES results are different because of the simplification in the transfer function. The measurements for the forward and downward speed are not consistent with each other and the coherence range is low and therefore these results are less accurate.

The accuracy of the state space system derived from the flight tests is indicated by the cost function of the fit, Cramér-Rao bounds, insensitivity, coherence of the response and consistency of the measurements. Unfortunately, the coherence of the response of the forward speed was too low, so this response is taken out of the analysis. Because of this the Cramér-Rao bounds and insensitivity could not be calculated. Therefore, the accuracy of the estimation of the parameters for the state space system cannot be proven. The state space results are for this reason not trusted while the cost functions suggests that the state system is a good fit.

The accuracy of the transfer functions and state space system of the simulation are indicated by the applicability of the simulation. There is a considerable discrepancy between the responses of the simulation and the flight test, this is caused by the simplifications in the simulation. The extraction of the influence of the propellers is one of the simplifications which influences the stability of the aircraft due to the normal force generated by the propeller and the thrust lapse. It can also be seen that the lift to drag ratio is optimistic, indicating that these results alone are not accurate. However, when the errors due to the simplifications are taken into account, the results can be compared to the flight test results.

The uncertainty of the measurements is investigated by researching the error and resolution of the systems. These devices show a good resolution and, when available, low errors. The results are improved by the EKF-filter, which is built into PX4 and due to this filter, it was not possible to find the uncertainty of the measurements. The uncertainty of the measurements is also investigated by the repeatability tests. The bode plots of three different sweeps for the same input are plotted, which should all show similar results when the sweeps are performed satisfactory as is the case. Therefore, the uncertainty of the measurements is sufficient for this analysis.

The handling characteristics are derived for the Bandwidth, CAP, short period damping and phugoid damping. The bandwidth is derived from the direct response which was gained from the flight test. The CAP and short period damping are derived from the transfer functions which are gained from a LOES model, SISO model, MIMO model and simulation. The phugoid damping is derived from the transfer functions of the SISO model, MIMO model and the simulation.

When the results of the handling characteristics investigated are compared to the military standards, most of them are satisfactory as they are level 1. Only the LOES results show handling characteristics which are rated below level 2. This value is the short period damping and these results are not trusted because of the simplification of removing the phugoid motion in the transfer function. The transfer functions containing both the phugoid and short period are preferred over the LOES results. The short period damping of the MIMO results is the only value that is level 2 and this value is close to the bound of level 1 and 2. Because of this and because of the scaling issues which arise from the small size of the aircraft.

To conclude, acceptable transfer functions are found by the flight test results for the pitch angle and pitch speed response relative to elevator deflection. Unfortunately, the flight test data and simulation were not sufficient to estimate the full state space system for the longitudinal dynamics. However, the pitch angle or pitch speed response contain all the information that is necessary to calculate the handling characteristics. This made it possible to estimate the required handling characteristics without deriving the whole system. Finally, the handling characteristics of the Skysurfer X8 are satisfactory because they meet or exceed the level 2 handling criteria, according to the SISO analysis, MIMO analysis and simulation.

6 Recommendations

The method in this report can be used as a guide for the evaluation of the handling characteristics (the short period, phugoid and CAP) of other scaled models, for example the Flying-V. With some modifications to the test protocol, the results could be improved to gather the state space system for the longitudinal dynamics of the aircraft and increase the quality of the data for the throttle sweep and the forward speed of the aircraft.

First of all, the flight test protocol should be precisely executed according to the predetermined plan, this makes sure that no maneuvers will be forgotten. In these flight tests only one frequency sweep is executed for the throttle input. But fortunately, subsequent throttle frequency sweeps would not have yielded better results, due to the expected low coherence. However, these extra flights would to the reliability of the results, if the problem of the low coherence would be solved. To improve the results for the throttle input, the input for the sweep should be changed. At low frequencies the coherence is better, lowering the test frequency of the input signal could improve the outcomes as it gives the aircraft more time to react to the signal.

In these tests the doublets were not distinguishable from the inputs on the elevator, because there was no clear trim condition and the maneuvers were executed too quicky after each other. More time should be investigated in finetuning the doublets. A correct amplitude and interval should be chosen to make these maneuvers stand out during a flight test and to make them useful for the verification. This verification would make a great improvement in the reliability of the results.

The reliability of the results could be improved by verifying the measurements of the Pixhawk. Unfortunately, the sensor fusion of EKF-filter in PX4 makes it hard to separately investigate the measurement systems. The speed for example consists of measurements of the pitot sensor, accelerometers and the GPS. From one or multiple of these sources an estimation of the speed of the aircraft is calculated. In the case that something happens to one of the sensors, there is no direct problem with the flight of the aircraft. In the case that the pitot is statically tested, the Pixhawk will know that it is standing still and that the aircraft is not moving and therefore the speed will be adjusted, making it impossible to verify separate measurement systems on the Pixhawk. Therefore, the most efficient way to verify the measurements of the Pixhawk is with a validated device on a moving object, preferably in the air with no distortions around it.

Bibliography

[1] Martinez-Val, R., "Flying Wings: A New Paradigm for Civil Aviation?," *Acta Polytechnica* Vol. 47, No. 1, 2007. https://ojs.cvut.cz/ojs/index.php/ap/article/view/914/746

[2] Jordan, T. L., and Bailey, R. M., "NASA Langley's AirSTAR Testbed – A Subscale Flight Test Capability for Flight Dynamics and Control System Experiments," *AIAA Guidance, Navigation and Control Conference and Exhib*, No. 2008-6660, Honolulu, Hawaii, 2008. https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2008-6660

[3] Kallo, J., Rathke, P., Stephan, T., Thalau, O., Schirmer, J., and Teich, H., "American Institute of Aeronautics and Astronautics Fuel Cell Systems for Aircraft Application & Antares DLR-H2 All-Electric Flying Testbed," *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, No. 2013-0936, Grapevine (Dallas/Ft. Worth Region), Texas, 2013. https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2013-936

[4] Sobron, A., "On Subscale Flight Testing. Applications in Aircraft Conceptual Design," *Department of Management and Engineering*, University of Linköping, 2018. https://www.diva-portal.org/smash/get/diva2:1260642/FULLTEXT01.pdf

[5] Amadori, K., Jouannet, C., and Berry, P., "Development of a subscale flight testing platform for a generic future fighter," *27TH INTERNATIONAL CONGRESS OF THE AERONAUTICAL SCIENCES - ICAS 2010* No., 2010.

[6] Jouannet, C., Lundström, D., Krus, P., Sobron, A., Silva, R. G. A. d., Catalano, F., and Greco, P., "Aerodynamic databased of a subscale demonstrator," *35th AIAA Applied Aerodynamics Conference* No. 2017-4075, Denver, Colorado, 2017. https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2017-4075

[7] Cooper, G. E., and Jr., R. H., "The use of pilot rating in the evaluation of aircraft handling qualities," *Advisory group for aerospace research and development*, 1969. https://apps.dtic.mil/dtic/tr/fulltext/u2/689722.pdf

[8] Benad, J., "The Flying V - A new Aircraft Configuration for Commercial Passenger Transport," *Deutscher Luft-und Raumfahrtkongress*, 2015. https://www.researchgate.net/publication/324150114_The_Flying_V__A_new_Aircraft_Configuration_for_Commercial_Passenger_Transport

[9] Empelen, S. v., "Engine Integration of the Flying V: Quantification of Engine Integration Effects using Wind Tunnel Experiments," *Aerospace Engineering*, Master thesis, Delft University of Technology, 2020. http://resolver.tudelft.nl/uuid:c519caf8-0eba-4633-a4f9-be37684417a8

[10] Palermo, M., "The Longitudinal Static Stability and Control Characteristics of a Flying V Scaled Model," *Aerospace Engineering*, Master thesis, Delft University of Technology, 2019. http://resolver.tudelft.nl/uuid:6286f9e2-c24a-430c-a4fa-9fb67b9558b4

[11] Faggiano, F., "Aerodynamic Design Optimization of a Flying V Aircraft," *Aerospace engineering*, Master thesis, Delft University of Technology, 2016. http://resolver.tudelft.nl/uuid:0b1472a5-3aad-433c-9a64-242c84b114fd

[12] Hillen, M., "Parametrisation of the Flying-V Outer Mould Line," *Aerospace Engineering*, Master thesis, Delft University of Technology, 2020. http://resolver.tudelft.nl/uuid:f4863ae4-2792-4335-b929-ff9dfdb6fed5

[13] Schaft, L. v. d., "Development, Model Generation and Analysis of a Flying V Structure Concept," *Aerospace Engineering*, Master thesis, Delft University of Technology, 2017. http://resolver.tudelft.nl/uuid:d9c9c02f-d67a-4e3c-93a7-eb20ed67cd03

[14] Claeys, M., "Flying V and Reference Aircraft Structural Analysis and Mass Comparison," *Aerospace Engineering*, Master thesis, Delft University of Technology, 2018. http://resolver.tudelft.nl/uuid:ee7f2ecb-cdb6-46de-8b57-d55b89f8c7e6

[15] Rubio Pascual, B., "Engine-Airframe Integration for the Flying-V," *Aerospace Engineering*, Master thesis, Delft University of Technology, 2018. http://resolver.tudelft.nl/uuid:75be27a7-6fd4-4112-a600-45df2999758f

[16] Mitchell, D. G., "Fifty Years of the Cooper-Harper Scale," *AIAA SciTech Forum*, No. 2019-0563 San Diego, California, 2019.
https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2019-0563

[17] Harper, R. P., and Cooper, G. E., "Wright Brothers Lectureship in Aeronautics: Handling Qualities and Pilot Evaluation," *AIAA/AHS/ASEE Aircraft Design Systems and Operations Meeting*, No. 84-2442, San Diego, California 1984.
https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.1984-2442

[18] Kivioja, D. A., "Comparison of the Control Anticipation Parameter and the Bandwidth Criterion During the Landing Task," Master thesis, Air Force Institute of Technology, 1996.
https://apps.dtic.mil/dtic/tr/fulltext/u2/a319509.pdf

[19] Mooij, H. A., "Criteria for low-speed longitudinal handling qualities of transport aircraft with closedloop flight control systems," *Aerospace Engineering*, Doctoral thesis, Delft Univeristy of Technology, 1984.
http://resolver.tudelft.nl/uuid:1fb4367a-0713-40a9-8100-9a58ed67a4eb

[20] Gibson, J. C., " Development of a Methodology for Excellence in Handling Qualities Design for Fly By Wire Aircraft.," *Aerospace Engineering*, Doctoral thesis, Delft University of Technology, 1999.
http://resolver.tudelft.nl/uuid:6b564b35-cb74-436c-8c47-845bfbbb8b4d

[21] Neal, T. P., and Smith, R. E., "An in-flight investigation to develop control system design criteria for fighter airplanes 1," AFFDL-TR-70-74, Cornell Aeronautical Laboratory, 1970.
https://apps.dtic.mil/dtic/tr/fulltext/u2/880426.pdf

[22] Neal, T. P., and Smith, R. E., "An in-flight investigation to develop control system design criteria for fighter airplanes 2," AFFDL-TR-70-74, Cornell Aeronautical Laboratory, 1970.
https://apps.dtic.mil/dtic/tr/fulltext/u2/880252.pdf

[23] Smith, R. H., and Geddes, N. D., "Handling quality requirements for advanced aircraft design: longitudinal mode," AFFDL-TR-78-154, Air Force, 1979.
https://apps.dtic.mil/dtic/tr/fulltext/u2/a077858.pdf

[24] Field, E. J., Rossitto, K. F., and Mitchell, D. G., "Landing Approach Flying Qualities Criteria For Active Control Transport Aircraft," ADPO11128, The Boeing Company and Hoh Aeronautics.
https://apps.dtic.mil/dtic/tr/fulltext/u2/p011128.pdf

[25] Cotting, M. C., "Proposed Longitudinal Flying Qualities Criterion for Unpiloted Autonomous Aircraft, Starting the Conversation," *Atmospheric Flight Mechanics Conference*, No. 7944, Toronto, Ontario Canada, 2010.
https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2010-7944

[26] "MIL-F-8785C5 MILITARY SPECIFICATION: FLYING QUALITIES OF PILOTED AIRPLANES," MIL-F-8785C5, Department of Defence, 1980.
http://www.mechanics.iei.liu.se/edu_ug/tmme50/8785c.pdf

[27] "MIL-HDBK-1797 DEPARTMENT OF DEFENSE HANDBOOK FLYING QUALITIES OF PILOTED AIRCRAFT," MIL-HDBK-1797, Air force, 1994.
http://www.mechanics.iei.liu.se/edu_ug/tmme50/MIL-HDBK-1797.PDF

[28] Klyde, D., McReur, D., and Myers, T., "Unified Pilot-Induced Oscillation Theory. Vol. I: PIO Analysis with Linear and Nonlinear Effective Vehicle Characteristics, Including Rate Limiting," WL-TR 96-3028, Wright Laboratory, Wright-Patterson Air Force Base, 1995.

[29] Greene, K. M., Kunz, D. L., and Cotting, M. C., "Toward a Flying Qualities Standard for Unmanned Aircraft," *AIAA Atmospheric Flight Mechanics Conference*, No. 2014-2194, Atlanta, Georgia, 2014.
https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2014-2194

[30] Cotting, M. C., "Applicability of Human Flying Qualities Requirementsfor UAVs, Finding A Way Forward," *AIAA Atmospheric Flight Mechanics Conference*, No. 2009-6322, Chicago, Illinois, 2009.
https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2009-6322

[31] Foster, T. M., and Bowman, W. J., "DYNAMIC STABILITY AND HANDLING QUALITIES OF SMALL UNMANNED-AERIAL-VEHICLES," *43rd AIAA Aerospace Sciences Meeting and Exhibit*, No. 2005-1023, Reno, Nevada, 2005.

https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2005-1023

[32] Tischler, M. B., and Remple, R. K. "Aircraft and Rotorcraft System Identification Engineerling Methods with Flight Test Examples," *AIAA Education Series*, 2nd ed, AIAA, Reston, Virginia, 2012.

[33] Woodrow, P. M., Tischler, M. B., Hagerott, S. G., Mendoza, G. E., and Hunter, J. M., "Low Cost Flight-Test Platform to Demonstrate Flight Dynamics Concepts using Frequency-Domain System Identification Methods," *AIAA Atmospheric Flight Mechanics Conference* No., Boston Massachusetts, 2013.
https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2013-4739

[34] Smith, G. D., Bixle, B. M., Babcock, J. T., Osteroos, R. K., McLaughlin, T. E., and Tischler, M. B., "System Identification of the ICE/SACCON UAS Aircraft," *AIAA SciTech Forum* No. 2020-0289, Orlando, Florida, 2020.
https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2020-0289

[35] Tischler, M. B., Leung, J. G. M., and Dugan, D. C., "Frequency-Domain Identification of XV-15 Tilt-Rotor Aircraft Dynamics," *AIAA 2nd Flight Testing Conference*, No. 83-2695, Las Vegas, Nevada, 1983.
https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.1983-2695

[36] Sanders, F. C., Tischler, M. B., Berger, T., Berrios, M. G., and Gong, A., "System Identification and Multi-Objective Longitudinal Control Law Design for a Small Fixed-Wing UAV," *AIAA SciTech Forum*, No. 2018-0296, Kissimmee, Florida, 2018.
https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2018-0296

[37] Tischler, M. B., Leungt, J. G. M., and Dugan, D. C., "Identification and Verification of Frequency-Domain Models for XV-15 Tilt-Rotor Aircraft Dynamics in Cruising Flight," *Journal of Guidance* Vol. 9, No. 4, 1985.
https://doi-org.tudelft.idm.oclc.org/10.2514/3.20131

[38] Chowdhary, G., DeBusk, W. M., and Johnson, E. N., "Real-Time System Identification of a Small Multi-Engine Aircraft with Structural Damage " *AIAA Infotech@Aerospace*, No. 2010-3472, Atlanta, Georgia, 2010.
https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2010-3472

[39] DeBusk, W. M., Chowdhary, G., and Johnson, E. N., "Real-Time System Identification of a Small Multi-Engine Aircraft " *AIAA Atmospheric Flight Mechanics Conference*, No. 2009-5935, Chicago, Ilinois, 2009.
https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2009-5935

[40] Scheper, K. Y. W., Chowdhary, G., and Johnson, E. N., "Aerodynamic System Identification of Fixed-wing UAV," *AIAA Atmospheric Flight Mechanics Conference* No. 2013-4920 Boston, MA, 2013.
https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2013-4920

[41] Antoni, J., and Schoukens, J., "FRF measurements with the weighted overlapped segment averaging technique," *IFAC Proceedings Volumes* Vol. 39, No. 1, 2006, pp. 386-391.
https://doi.org/10.3182/20060329-3-AU-2901.00057
(https://www.sciencedirect.com/science/article/pii/S1474667015352939)

[42] Acree, C. W., and Tischler, M. B., "Determining XV-15 Aeroelastic Modes from Flight Data with Frequency-Domain Methods," TP-3330, NASA, 1993.
https://ntrs.nasa.gov/api/citations/19940006463/downloads/19940006463.pdf?attachment=true

[43] Hoh, R. H., Mitchell, D. G., and Hodgkinson, J., "Bandwidth - A Criterion for Highly Augmented Airplanes," *Flight Mechanics Panel Symposium on Criteria for Handling Qualities of Military Aircraft* No. A118596, Fort Worth, 1982.
https://apps.dtic.mil/dtic/tr/fulltext/u2/a118596.pdf

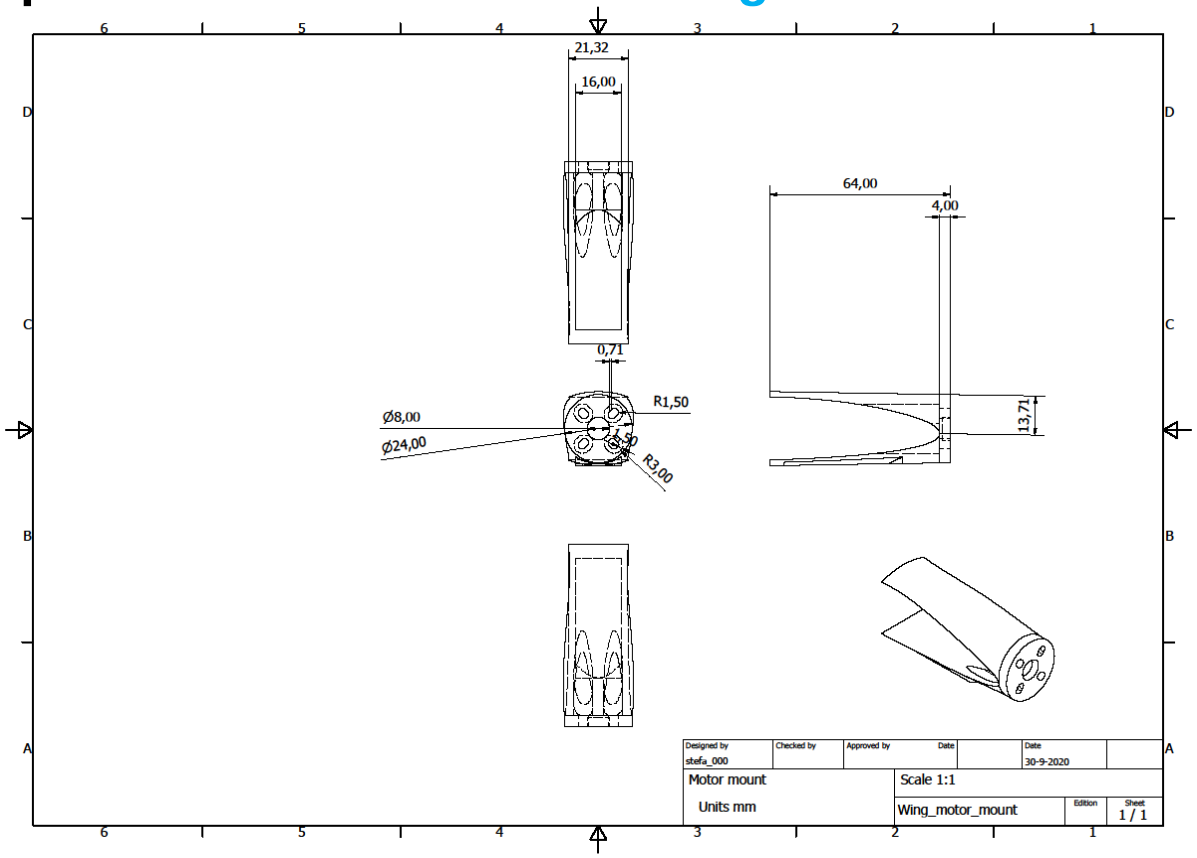
[44] Rosenbrock, H. H., "An Automatic Method for Finding the Greatest or Least Value of a Function," *The computer journal* Vol. 3, No. 3, 1960, pp. 175-184.
https://doi-org.tudelft.idm.oclc.org/10.1093/comjnl/3.3.175

[45] Cook, M. V. "Flight Dynamics Principles: A Linear Systems Approach to Aircraft Stability and Control," 2nd ed, Elsevier, Burlington, MA, 2007.

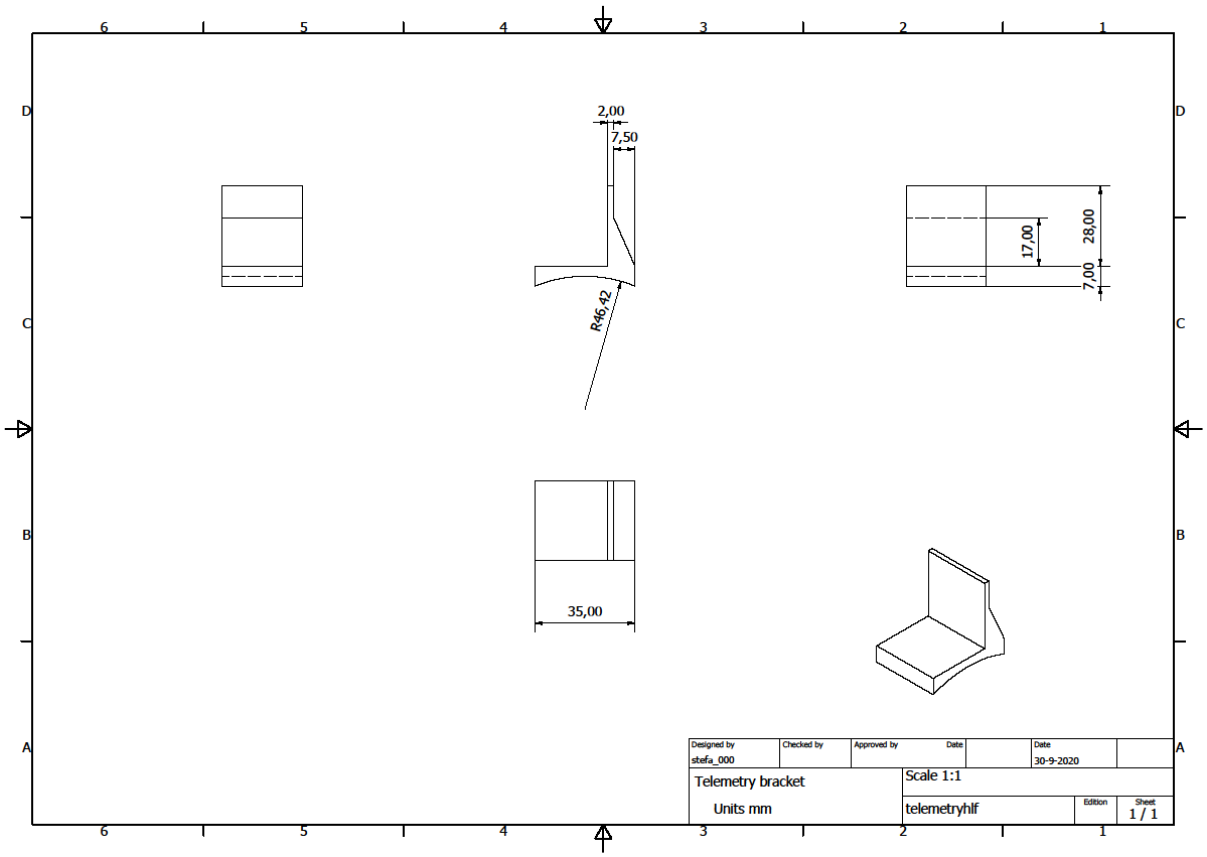
[46] Bos, A. v. d., "A Cramer-Rao Lower Bound for Complex Parameters," *IEEE Transactions on signal processing* Vol. 42, NO. 10, 1994.
[47] Vanreusel, S., "IDENTIFICATION AND MULTIVARIABLE FEEDBACK CONTROL OF THE VIBRATION DYNAMICS OF AN AUTOMOBILE SUSPENSION " *Department of Electrical Engineering*, Master thesis, McGill University, Montreal, 2005.
[48] Bolding, R. L., "Handling Qualities Evaluation of a Variable Stability Navion Handling Qualities Evaluation of a Variable Stability Navion Airplane (N66UT) using Frequency Domain Test Techniques," Master Thesis, University of Tennessee, 1998. https://trace.tennessee.edu/cgi/viewcontent.cgi?article=3022&context=utk_gradthes
[49] Catterall, R. C., "State-Space Modeling of the Rigid-Body Dynamics of a Navion State-Space Modeling of the Rigid-Body Dynamics of a Navion Airplane From Flight Data, Using Frequency-Domain Identification Airplane From Flight Data, Using Frequency-Domain Identification Techniques," Master thesis, University of Tennessee, 2003. https://trace.tennessee.edu/cgi/viewcontent.cgi?article=3269&context=utk_gradthes
[50] Siu, J. M. Y., "Control Law Calculation and Verification Methods for the Variable Control Law Calculation and Verification Methods for the Variable Stability Navion In-Flight Simulation Aircraft ", Master thesis, University of Tennessee, 2013. https://trace.tennessee.edu/cgi/viewcontent.cgi?article=2588&context=utk_gradthes
[51] Dorobantu, A., Murch, A., Mettler, B., and Balas, G., "System Identification for Small, Low-Cost, Fixed-Wing Unmanned Aircraft," *Journal of Aircraft* Vol. 50, No. 4, 2013. https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/1.C032065
[52] Dorobantu, A., Murch, A. M., Mettler, B., and Balas, G. J., "Frequency Domain System Identification for a Small, Low-Cost, Fixed-Wing UAV," *AIAA Guidance, Navigation and Control Conference*, No. 2011-6719, Portland, Oregon, 2011. https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2011-6719
[53] Schulze, P. C., Miller, J. P., Klyde, D. H., Regan, C. D., and Alexandrov, N., "System Identification of a Small UAS in Support of Handling Qualities Evaluations," *AIAA SciTech Forum*, No. 2019-0826, San Diego, California, 2019. https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2019-0826
[54] Simsek, O., and Tekinalp, O., "System Identification and Handling Quality Analysis of a UAV from Flight Test Data," *AIAA Atmospheric Flight Mechanics Conference*, No. 2015-1480 Kissimmee, Florida, 2015. https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdfplus/10.2514/6.2015-1480
[55] Venkataraman, R., and Seiler, P., "System Identification for a Small, Rudderless, Fixed-Wing Unmanned Aircraft," *Journal of Aircraft* Vol. 56, No. 3, 2019. https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/1.C035141
[56] Ahmed, U., "3-DOF Longitudinal Flight Simulation Modeling And Design Using MATLAB/SIMULINK," *Aerospace Engineering*, Master thesis, Ryerson University, 2012. https://digital.library.ryerson.ca/islandora/object/RULA%3A1505
[57] Nelson, R. C. "Flight Stability and Automatic Control," 2nd ed, McGraw-Hill, 1998.
[58] Septiyana, A., Hidayat, K., Rizaldi, A., Ramadiansyah, M. L., Ramadhan, R. A., Suseno, P. A. P., Jayanti, E. B., Atmasari, N., and Rasyadi, A., "Analysis of Aerodynamic Characteristics Using the Vortex Lattice Method on Twin Tail Boom Unmanned Aircraft," *7th International Seminar on Aerospace Science and Technology – ISAST*, No. AIP Conf. Proc. 2226, 2019. https://aip-scitation-org.tudelft.idm.oclc.org/doi/abs/10.1063/5.0002337
[59] Martin, C. A., "Power effects on the longitudinal characteristics of single-engine propeller-driven aircraft," ARL-AERO-REPORT-157, Aeronautical research laboratories, Melbourne, Victoria, Australia, 1984. https://apps.dtic.mil/dtic/tr/fulltext/u2/a137896.pdf
[60] Kumar, P., E.Steck, J., and Hagerott, S. G., "American Institute of Aeronautics and Astronautics1System Identification, HIL and Flight Testing of an Adaptive Controllerona Small Scale Unmanned Aircraft," *AIAA Modeling and Simulation Technologies Conference*, No. 2015-1803, Kissimmee, Florida, 2015. https://arc-aiaa-org.tudelft.idm.oclc.org/doi/pdf/10.2514/6.2015-1803

[61] Gill, P. E., and Murray, W., "NEWTON-TYPE METHODS FOR UNCONSTRAINED AND LINEARLY CONSTRAINED OPTIMIZATION," *Mathematical Programming* Vol. 7, 1972, pp. 311-350.

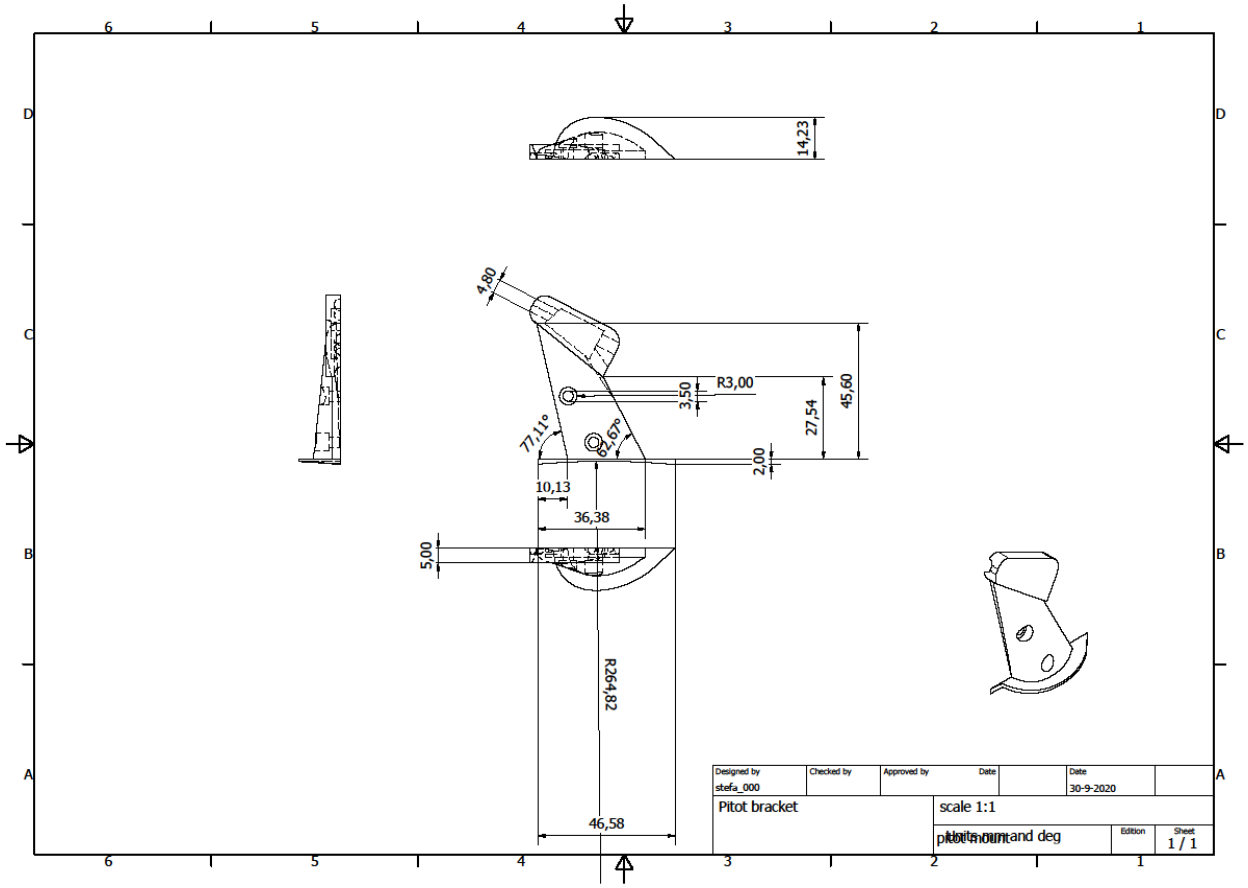
Appendix A Technical drawings of brackets



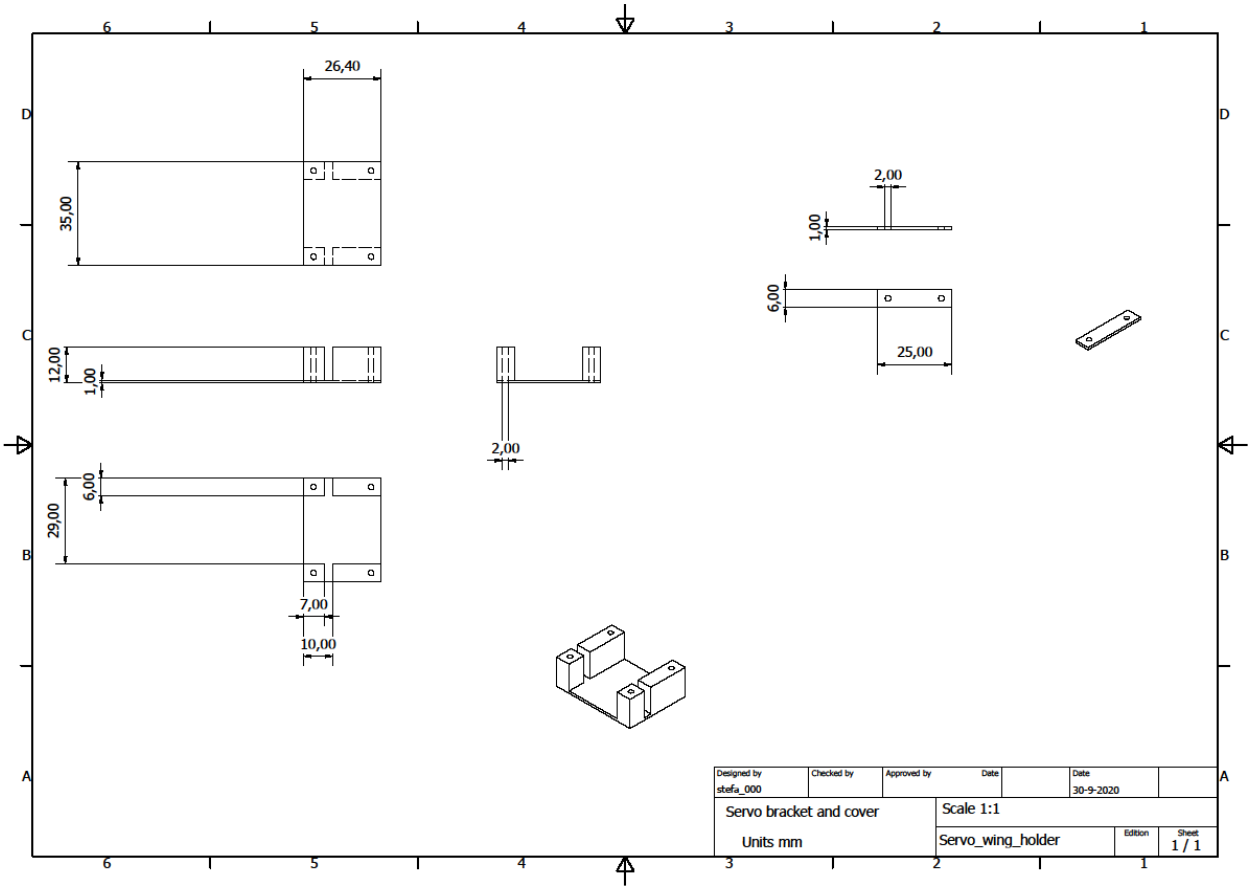
Appendix figure A-1: Technical drawing motor mount



Appendix figure A-2: Technical drawing telemetry mount



Appendix figure A-3: Technical drawing pitot mount



Appendix figure A-4: Technical drawing servo mount

Appendix B Software/firmware changes PX4

Mixerfiles:
Location: Firmware/ROMFS/px4fmu_common/mixers/SKYSURFER.main.mix

Aileron/Elevator/Throttle/Rudder mixer
==========

This file defines mixers suitable for controlling a fixed wing aircraft with aileron, rudder, elevator, throttle, gear, flaps controls. The configuration assumes the aileron servo(s) are connected to output 0, the elevator to output 1, the throttle to output 2 and the rudder to output 3.

Inputs to the mixer come from channel group 0 (vehicle attitude), channels 0 (roll), 1 (pitch), 2 (thrust), 3 (yaw), 4 (flaps), 7 (landing gear)

CH1 CH2: Aileron mixer

Two scalers total (output, roll).

This mixer assumes that the aileron servos are set up correctly mechanically; depending on the actual configuration it may be necessary to reverse the scaling factors (to reverse the servo movement) and adjust the offset, scaling and endpoints to suit.

As there is only one output, if using two servos adjustments to compensate for differences between the servos must be made mechanically. To obtain the correct motion using a Y cable, the servos can be positioned reversed from one another.

M: 1
S: 0 0 -10000 -10000 0 -10000 10000

M: 1
S: 0 0 -10000 -10000 0 -10000 10000

CH3: Elevator mixer

Two scalers total (output, roll).

This mixer assumes that the elevator servo is set up correctly mechanically; depending on the actual configuration it may be necessary to reverse the scaling factors (to reverse the servo movement) and adjust the offset, scaling and endpoints to suit.

M: 1
S: 0 1 -10000 -10000 0 -10000 10000

CH4 CH5: Motor speed mixer

Two scalers total (output, thrust).

This mixer generates a full-range output (-1 to 1) from an input in the (0 - 1) range. Inputs below zero are treated as zero.

```
M: 1
S: 0 3      0  20000  -10000  -10000  10000

M: 1
S: 0 3      0  20000  -10000  -10000  10000

CH6: Rudder mixer
------------
Two scalers total (output, yaw).

This mixer assumes that the rudder servo is set up correctly mechanically;
depending on the actual configuration it may be necessary to reverse the scaling
factors (to reverse the servo movement) and adjust the offset, scaling and
endpoints to suit.

M: 1
S: 0 2   10000  10000      0  -10000  10000
```

Location: Firmware/ROMFS/px4fmu_common/init.d/airframes/2103_skysurfer

```
#!/bin/sh
#
# @name Skysurfer
#
# @type Standard Plane
# @class Plane
#
# @output MAIN1 aileron
# @output MAIN2 elevator2x
# @output MAIN3 throttle2x
# @output MAIN4 rudder
# @output MAIN5 flaps
# @output MAIN6 gear
#
# @output AUX1 feed-through of RC AUX1 channel
# @output AUX2 feed-through of RC AUX2 channel
# @output AUX3 feed-through of RC AUX3 channel
#
#

sh /etc/init.d/rc.fw_defaults

if [ $AUTOCNF = yes ]
then
	param set PWM_AUX_RATE 50
	param set PWM_RATE 50
fi

set MIXER SKYSURFER

# Rate must be set by group (see pwm info).
# Throttle is in the same group as servos.
```

Appendix C Flight test cards

<table>
<tr><td>Test number: 01</td><td>Initial Setup</td><td>Date:-</td><td>Time: -</td><td>Successful/Unsuccessful</td></tr>
<tr><td colspan="2">Pre-flight checks:
1. Check that all electrical connections are secure and that all components are fastened to the airframe
2. Insert and fasten batteries
3. Press safety switch
4. Close off all compartments and secure with screws
5. Verify CG-location
6. Perform calibration of onboard sensors and set tolerances
7. Set geofence in ground station</td><td colspan="2">Goals:
○ Verify that aircraft is flightworthy</td><td>Motivation:

Time end:

Time start:</td></tr>
<tr><td colspan="4">Measurements:

<table><tr><th>Check</th><th>Function</th><th>Status</th><th>CHECK</th></tr><tr><td>Main Battery Voltage above 12V</td><td>Engine/Pixhawk power</td><td></td><td></td></tr></table></td><td>Notes:</td></tr>
</table>

<table>
<tr><td>Test number: 02</td><td>Control deflections test</td><td>Date:-</td><td>Time:-</td><td>Successful/Unsuccessful</td></tr>
<tr><td colspan="2">Pre-flight checks:
1. Make sure the aircraft is disarmed and safely secured
2. Check if the inputs of the remote are according to the table in measurements</td><td colspan="2">Goals:
○ Verify if the control surfaces work accordingly
○ Verify if the control surfaces deflect enough</td><td>Motivation:

Time end:

Time start:</td></tr>
<tr><td colspan="4">Measurements:

<table><tr><th>Control surface</th><th>Function</th><th>Control on radio</th><th>CHECK</th></tr><tr><td>Left Wing</td><td>Aileron</td><td>Right Stick</td><td></td></tr><tr><td>Right Wing</td><td>Aileron</td><td>Right Stick</td><td></td></tr><tr><td>Elevator</td><td>Elevator</td><td>Right Stick</td><td></td></tr><tr><td>Rudder</td><td>Rudder</td><td>Left Stick</td><td></td></tr></table></td><td>Notes:</td></tr>
</table>

<table>
<tr><td>Test number: 03</td><td>Engine test</td><td>Date:</td><td>Time:</td><td>Successful/Unsuccessful</td></tr>
<tr><td colspan="2">Pre-flight checks:
1. Clear the site
2. Two/three persons hold the vehicle to counteract the thrust forces
3. Connect Engine Battery
4. Arm the vehicle</td><td colspan="2">Goals:
o Verify if the engine and temperature sensors are working correctly
o Verify if the engines and batteries do not overheat</td><td>Motivation:

Time end:

Time start:</td></tr>
<tr><td colspan="2">Flight test:
1. Full throttle for 30 seconds
2. Half throttle for 3 minutes
3. Disarm the vehicle</td><td colspan="2">Measurements:
 Full throttle for 5 seconds gave no problems
 Half throttle for 4 minutes gave no problems
All below 100 degrees Celsius:
Max. temperature engines:

Max. temperature ESC:

Max. temperature batteries:

Name datafile:

Location datafile:</td><td>Notes:</td></tr>
</table>

<table>
<tr><td>Test number: 04</td><td>Range test</td><td>Date:</td><td>Time: 08:30-09:00</td><td>Successful/Unsuccessful</td></tr>
<tr><td colspan="2">Pre-flight checks:
□ Pre-flight checklist complete
1. Make sure the aircraft is disarmed and safely secured
2. Create a distance of 1000m between the aircraft and the receiver to check if the connections are sufficient.</td><td colspan="2">Goals:
○ Verify the range of the radio frequency equipment on board for maximum distance during flight.</td><td>Motivation:

Time end:

Time start:</td></tr>
<tr><td colspan="2">Flight test:
1. Check that aircraft responds to control inputs from a large distance
2. Check that onboard telemetry maintains connection with ground station</td><td colspan="2">Measurements:

<table><tr><td></td><td>1000m</td></tr><tr><td>RC Jeti connection RSSI</td><td></td></tr><tr><td>Telemetry RFD868 RSSI</td><td></td></tr></table>
Name datafile:

Location datafile:</td><td>Notes:</td></tr>
</table>

<table>
<tr><td>Test number: 05</td><td>Simple flight test 1: Circle and back</td><td>Date:-</td><td>Time:-</td><td>Successful/Unsuccessful</td></tr>
<tr><td colspan="2">Pre-flight checks:
□ GO NO GO list complete
1. Arm the vehicle</td><td colspan="2">Goals:
o Perform the first flight
o Check the performance of stabilized mode
o Evaluate CG-location
o Estimation for energy consumption/endurance</td><td>Motivation:

Time end:

Time start:</td></tr>
<tr><td colspan="2">Flight test:
1. Set mode to stabilized
2. Set the timer at engines spool up
3. Accelerate to take off speed 12 m/s
4. Rotate the vehicle with full elevator deflections
5. Climb straight out till 20m altitude
6. Rotate 180-degrees: R 30m
7. Fly straight for 400m
8. Rotate 180-degrees: R 30m
9. Rotate 180 degrees: R 30m 2x
10. Approach
11. Land go around if necessary
12. Disarm
13. Stop timer and write time down
14. Check cell voltages and write down
15. If necessary, perform again with different weight distribution.</td><td colspan="2">Measurements:
B= Before A = After
Centre of gravity position:
Voltages: battery :
B
A
Name datafile:
Location datafile:</td><td>Notes:</td></tr>
</table>

<table>
<tr><td>Test number:
06 b</td><td>Simple flight test 1:
Return</td><td>Date:</td><td>Time: 10:30-11:00</td><td>Successful/Unsuccessful</td></tr>
<tr><td colspan="2">Preflight checks:
GO NO GO list complete
2. Arm the vehicle</td><td colspan="2">Goals:
o Evaluate CG-location
o Estimation for energy consumption/endurance
o Check the performance of the return mode</td><td>Motivation:

Time end:

Time start:</td></tr>
<tr><td colspan="2">Flight test:
1. Set mode to stabilized
2. Accelerate to take off speed 12 m/s
3. Rotate the vehicle with full elevator deflections
4. Climb straight out till 20m altitude
5. Rotate 180-degrees: R 30m
6. Fly straight for 400m
7. Rotate 180-degrees: R 30m 2x
8. Fly straight for 400m 2x
9. Rotate 180 degrees: R 30m
10. From a distance of 300m from home turn on return mode
11. Approach
12. Land go around if necessary
13. Disarm
14. Check cell voltages and write down</td><td colspan="2">Measurements:
B= Before A = After
Centre of gravity position:
Voltage:
Battery 1:
B
A
Name datafile:
Location datafile:</td><td>Notes:</td></tr>
</table>

<table>
<tr><td>Test number: 7</td><td>Flight test: Elevator and Thrust Sweep Test</td><td>Date:-</td><td>Time:-</td><td>Successful/Unsuccessful</td></tr>
<tr><td colspan="2">Preflight checks:
□ GO NO GO list complete
1. Arm the vehicle</td><td colspan="2">Goals:
○ Perform the first frequency sweep for the elevator and thrust</td><td>Motivation:

Time end:

Time start:</td></tr>
<tr><td colspan="2">Flight test:
1. Accelerate to take off speed 20 m/s
2. Rotate the vehicle with full deflections
3. Climb straight out till 20m
4. Rotate 180-degrees: R 30m
5. Fly straight
6. Enable thrust frequency sweep 3x
7. Rotate 180-degrees: R 30m 3x
8. Fly straight
9. Enable thrust step input 3x
10. Rotate 180-degrees: R 300m 3x
11. Fly straight
12. Enable elevator frequency sweep 3x
13. Rotate 180-degrees: R 30m 3x
14. Fly straight
15. Enable elevator step input 3x
16. Rotate 180-degrees: R 30m 3x
17. Approach and land (go around if required)
18. Disarm</td><td colspan="2">Measurements:

Name datafile:

Location datafile:</td><td>Notes:</td></tr>
</table>

GO NO GO checklist

- B Vehicle inspection no damage and bolds secure
- B Battery check cell voltage above 4.1V
- B Telemetry link connection, okay?
- B Vehicle closed off and secure
- B Control surface check
- B Check the maneuvers on the ground (if required)
- B Check environment
- B GO NO GO checklist complete

Appendix D MATLAB programs for preprocessing

Preprocess.mat

```
%% Preprocess
% Stefan Juffermans
% 4174658
%% A program to get the data from an already converted .Ulog to .mat log to a dataset containing the:
%  time, aileron deflection, elevator deflection, throttle input, rudder
%  deflection, Vx, Vx, Vy, Vz, ax, ay, az, roll speed, yaw speed, pitch
%  speed, roll, pitch, yaw, true airspeed, temperature, air density,
%  air pressure and height

function [dataset] = preprocess(name)
    datafile= name;

    %% Loading data

    log=load(datafile);                                                         % Import data under the name log

    % Loading input parameters
    timeinputs      = log.log.data.actuator_outputs_0.timestamp/1000000;        % Time stamp in seconds
    ailerons        = (log.log.data.actuator_outputs_0.output_1_-1500)/12.5;    % Aileron deflection in degrees
    elevators       = (log.log.data.actuator_outputs_0.output_2_-1500)*(3/50);  % Elevator deflection in degrees
    throttle        = (log.log.data.actuator_outputs_0.output_3_-1000)/10;      % Throttle in percentage
    rudder          = (log.log.data.actuator_outputs_0.output_5_-1500)/20;      % Rudder deflection in degrees

    % Loading Output parameters
    % Quaternations
    timeoutputs1=log.log.data.estimator_status_0.timestamp/1000000;             % Time stamp in seconds

    q0=log.log.data.estimator_status_0.states_0_;                               % Quaternations low frequency
    q1=log.log.data.estimator_status_0.states_1_;                               % Quaternations low frequency
    q2=log.log.data.estimator_status_0.states_2_;                               % Quaternations low frequency
    q3=log.log.data.estimator_status_0.states_3_;                               % Quaternations low frequency

    % Calculate Attitude
    roll=180/pi*atan2(2*(q3.*q2+q0.*q1),1-2*(q1.*q1+q2.*q2));                   % Roll angle in degrees
    pitch=180/pi*asin(2*(q2.*q0-q3.*q1));                                       % Pitch angle in degrees
    yaw=180/pi*atan2(2*(q3.*q0+q2.*q1),1-2*(q2.*q2+q3.*q3));                    % Yaw angle in degrees

    % Velocity NED
    Vn=log.log.data.estimator_status_0.states_4_;                               % Speed north m/s
    Ve=log.log.data.estimator_status_0.states_5_;                               % Speed east m/s
    Vd=log.log.data.estimator_status_0.states_6_;                               % Speed down m/s

    % Velocity body frame
    Vu=Vn.*cosd(pitch).*cosd(yaw)+Ve.*cosd(pitch).*sind(yaw)+Vd.*sind(pitch);    % Speed U,W and V in m/s
    Vv=Vn.*sind(roll).*-sind(pitch).*cosd(yaw)-Vn.*cosd(roll).*sind(yaw)+Ve.*sind(roll).*-
sind(pitch).*sind(yaw)+Ve.*cosd(roll).*cosd(yaw)+Vd.*sind(roll).*cosd(pitch);
    Vw=Vn.*cosd(roll).*-sind(pitch).*cosd(yaw)+Vn.*sind(roll).*sind(yaw)+Ve.*cosd(roll).*-sind(pitch).*sind(yaw)-
Ve.*sind(roll).*cosd(yaw)+Vd.*cosd(roll).*cosd(pitch);

    % Height
    timeoutputs2 = log.log.data.vehicle_local_position_0.timestamp/1000000;     % Timestamp in seconds
    H    = log.log.data.vehicle_local_position_0.z;                             % Height in meters

    % Accelerations
    ax = log.log.data.vehicle_local_position_0.ax;                              % Acceleration in X direction m/s
    ay = log.log.data.vehicle_local_position_0.ay;                              % Acceleration in Y direction m/s
    az = log.log.data.vehicle_local_position_0.az;                              % Acceleration in Z direction m/s
    timeoutputs3 = log.log.data.vehicle_attitude_0.timestamp/1000000;           % Time stamp in seconds

    pitchspeed      = log.log.data.vehicle_attitude_0.pitchspeed*57.2957795;    % Pitch speed in degrees/second
    yawspeed        = log.log.data.vehicle_attitude_0.yawspeed*57.2957795;      % Yaw speed in degrees/second
    rollspeed       = log.log.data.vehicle_attitude_0.rollspeed*57.2957795;     % Roll speed in degrees/second

    % Quaternations
    q02 = log.log.data.vehicle_attitude_0.q_0_;                                 % Quaternations high sample frequency
    q12 = log.log.data.vehicle_attitude_0.q_1_;                                 % Quaternations high sample frequency
    q22 = log.log.data.vehicle_attitude_0.q_2_;                                 % Quaternations high sample frequency
    q32 = log.log.data.vehicle_attitude_0.q_3_;                                 % Quaternations high sample frequency

    % Calculate Attitude
    roll2=180/pi*atan2(2*(q32.*q22+q02.*q12),1-2*(q12.*q12+q22.*q22));          % Roll angle in degrees high sample frequency
    pitch2=180/pi*asin(2*(q22.*q02-q32.*q12));                                  % Pitch angle in degrees high sample frequency
    yaw2=180/pi*atan2(2*(q32.*q02+q22.*q12),1-2*(q22.*q22+q32.*q32));           % Yaw angle in degrees high sample frequency

    % Air data
    time_airspeed = log.log.data.airspeed_0.timestamp/1000000;                  % Time stamp in seconds
    true_airspeed = log.log.data.airspeed_0.true_airspeed_m_s;                  % True airspeed in m/s
    temperature   = log.log.data.airspeed_0.air_temperature_celsius;            % Air temperature in Celcius

    time_air      = log.log.data.vehicle_air_data_0.timestamp/1000000;          % Time stamp in seconds
    air_density   = log.log.data.vehicle_air_data_0.rho;                        % Density in kg/m^3
    air_pressure  = log.log.data.vehicle_air_data_0.baro_pressure_pa;           % Presure in Pa

    %% Resampling

    % Make same length and start time
    start_time = max([timeinputs(2), timeoutputs1(2), timeoutputs2(2), timeoutputs3(2), time_airspeed(2), time_air(2)]);

    % Let all data start at the same time
    timeinputs = timeinputs-start_time;
    timeoutputs1 = timeoutputs1-start_time;
    timeoutputs2 = timeoutputs2-start_time;
    timeoutputs3 = timeoutputs3-start_time;
    time_airspeed = time_airspeed-start_time;
    time_air      = time_air-start_time;

    timeinputs = timeinputs(timeinputs>=0);
    timeoutputs1 = timeoutputs1(timeoutputs1>=0);
    timeoutputs2 = timeoutputs2(timeoutputs2>=0);
    timeoutputs3 = timeoutputs3(timeoutputs3>=0);
    time_airspeed = time_airspeed(time_airspeed>=0);
    time_air      = time_air(time_air>=0);


    timeinputs(1)   = 0;
    timeoutputs1(1) = 0;
    timeoutputs2(1) = 0;
    timeoutputs3(1) = 0;
    time_airspeed(1) = 0;
    time_air(1)      = 0;

    roll  = roll2( 1 + length(roll2)  - length(timeoutputs3) : end, : );
```

```
yaw   = yaw2(  1 + length(yaw2)   - length(timeoutputs3) : end, : );
pitch = pitch2(1 + length(pitch2) - length(timeoutputs3) : end, : );

rollspeed  = rollspeed(  length(rollspeed) - length(timeoutputs3) +1 : end, : );
yawspeed   = yawspeed(   length(yawspeed)  - length(timeoutputs3) +1 : end, : );
pitchspeed = pitchspeed( length(pitchspeed)- length(timeoutputs3) +1 : end, : );

ax = ax(length(ax)-length(timeoutputs2)+1:end,:);
ay = ay(length(ay)-length(timeoutputs2)+1:end,:);
az = az(length(az)-length(timeoutputs2)+1:end,:);

Vx = Vu(length(Vu)-length(timeoutputs1)+1:end,:);
Vy = Vv(length(Vv)-length(timeoutputs1)+1:end,:);
Vz = Vw(length(Vw)-length(timeoutputs1)+1:end,:);

true_airspeed  = true_airspeed( 1 + length(true_airspeed)  - length(time_airspeed) : end, : );
temperature    = temperature( 1 + length(temperature)  - length(time_airspeed) : end, : );
air_density    = air_density( 1 + length(air_density)  - length(time_air) : end, : );
air_pressure   = air_pressure(1 + length(air_pressure)  - length(time_air) : end, : );

ailerons = ailerons(length(ailerons)-length(timeinputs)+1:end,:);
elevators = elevators(length(elevators)-length(timeinputs)+1:end,:);
throttle = throttle(length(throttle)-length(timeinputs)+1:end,:);
rudder = rudder(length(rudder)-length(timeinputs)+1:end,:);

H      = H(length(H)-length(timeoutputs2)+1:end,:);

% save original signals
original.timeinputs   = timeinputs;
original.timeoutputs1 = timeoutputs1;
original.timeoutputs2 = timeoutputs2;
original.timeoutputs3 = timeoutputs3;
original.time_airspeed = time_airspeed;
original.time_air      = time_air

original.ailerons = ailerons;
original.elevators = elevators;
original.throttle = throttle;
original.rudder = rudder;

original.roll  = roll2;
original.yaw   = yaw2;
original.pitch = pitch2;

original.rollspeed  = rollspeed;
original.yawspeed   = yawspeed;
original.pitchspeed = pitchspeed;

original.ax = ax;
original.ay = ay;
original.az = az;

original.Vx = Vx;
original.Vy = Vy;
original.Vz = Vz;

original.true_airspeed = true_airspeed;
original.temperature   = temperature;
original.air_density   = air_density;
original.air_pressure  = air_pressure;

original.H = H;

% resample the signal at 10 Hz
[Vx,t]=resample(Vx, timeoutputs1, 10,1,1);
[Vy,t]=resample(Vy, timeoutputs1, 10,1,1);
[Vz,t]=resample(Vz, timeoutputs1, 10,1,1);

[ax,t]=resample(ax, timeoutputs2, 10,1,1);
[ay,t]=resample(ay, timeoutputs2, 10,1,1);
[az,t]=resample(az, timeoutputs2, 10,1,1);

[roll,t]=resample(roll, timeoutputs3, 10,1,1);
[pitch,t]=resample(pitch, timeoutputs3, 10,1,1);
[yaw,t]=resample(yaw, timeoutputs3, 10,1,1);

[rollspeed,t]=resample(rollspeed, timeoutputs3, 10,1,1);
[pitchspeed,t]=resample(pitchspeed, timeoutputs3, 10,1,1);
[yawspeed,t]=resample(yawspeed, timeoutputs3, 10,1,1);

[true_airspeed,t] =resample(true_airspeed, time_airspeed, 10,1,1);
[temperature,t]   =resample(temperature, time_airspeed, 10,1,1);
[air_density,t]   =resample(air_density, time_air, 10,1,1);
[air_pressure,t]   =resample(air_pressure, time_air, 10,1,1);

[ailerons,t]=resample(ailerons, timeinputs, 10,1,1);
[elevators,t]=resample(elevators, timeinputs, 10,1,1);
[throttle,t]=resample(throttle, timeinputs, 10,1,1);
[rudder,t]=resample(rudder, timeinputs, 10,1,1);

[height,t]= resample(H, timeoutputs2, 10,1,1);

% Make the same length again

leng = min([length(Vx) length(ax) length(pitch) length(pitchspeed) length(ailerons) length(true_airspeed)]);

t = t(1:leng,:);

Vx = Vx(1:leng,:);
Vy = Vy(1:leng,:);
Vz = Vz(1:leng,:);

ax = ax(1:leng,:);
ay = ay(1:leng,:);
az = az(1:leng,:);

roll = roll(1:leng,:);
pitch = pitch(1:leng,:);
yaw = yaw(1:leng,:);

rollspeed = rollspeed(1:leng,:);
pitchspeed = pitchspeed(1:leng,:);
yawspeed = yawspeed(1:leng,:);

true_airspeed = true_airspeed(1:leng,:);
temperature = temperature(1:leng,:);
air_density = air_density(1:leng,:);
air_pressure = air_pressure(1:leng,:);

ailerons = ailerons(1:leng,:);
elevators = elevators(1:leng,:);
throttle = throttle(1:leng,:);
rudder = rudder(1:leng,:);
```

```
    height = height(1:leng,:);

    % Save the data in one data set
    dataset.time           = t;

    dataset.input.aileron  = ailerons;
    dataset.input.elevator = elevators;
    dataset.input.throttle = throttle;
    dataset.input.rudder   = rudder;

    dataset.output.Vx  = Vx;
    dataset.output.Vy  = Vy;
    dataset.output.Vz  = Vz;

    dataset.output.ax  = ax;
    dataset.output.ay  = ay;
    dataset.output.az  = az;

    dataset.output.r   = rollspeed;
    dataset.output.p  = yawspeed;
    dataset.output.q    = pitchspeed;

    dataset.output.roll   = roll;
    dataset.output.pitch  = pitch;
    dataset.output.yaw    = yaw;

    dataset.output.AOA = atand(-Vz./Vx);
    dataset.output.true_airspeed = true_airspeed;
    dataset.output.temperature = temperature;
    dataset.output.air_density = air_density;
    dataset.output.air_pressure = air_pressure;

    dataset.output.height     = height;

    dataset.original = original;

end
```

Sys_id.mat

```
%% Sys_id

% Stefan Juffermans
% 4174658
%% A program to cut the sweeps out of the databases

clear all
%% Load flight test files
[database1] = preprocess('Stab_sweep_a.mat');                       % Flight test 1
[database2] = preprocess('Stab_sweep_b.mat');                       % Flight test 2

%% Set time slots sweeps and doublets
sweep.throttle.frame   = [240 273 1];                               % Throttle sweep   [start time [s], end time [s], number of test]
doublet.throttle.frame = [388 396 1 ; 398 402 1];                   % Throttle doublet [start time [s], end time [s], number of test]
sweep.elevator.frame   = [315 349 2 ; 118 143 2 ; 81 104 2];        % Elevator sweep   [start time [s], end time [s], number of test]

%% cutting out sweeps and saving them in the sweep parameter
% throttle sweep
for num =  1:length(sweep.throttle.frame(:,1))
    T_start = round(10*sweep.throttle.frame(num,1));               % Number in the array start
    T_end   = round(10*sweep.throttle.frame(num,2));               % Number in the array end

    if sweep.throttle.frame(num,3) == 1
        sweep.throttle.time(:,num)           = database1.time(T_start + 1 :  T_end + 1);                % Cutting out time of sweep
        sweep.throttle.input.aileron(:,num)  = database1.input.aileron(T_start + 1 :  T_end + 1);       % Cutting out aileron deflection of sweep
        sweep.throttle.input.elevator(:,num) = database1.input.elevator(T_start + 1 :  T_end + 1);      % Cutting out elevator deflection of sweep
        sweep.throttle.input.throttle(:,num) = database1.input.throttle(T_start + 1 :  T_end + 1);      % Cutting out throttle input of sweep
        sweep.throttle.input.rudder(:,num)   = database1.input.rudder(T_start + 1 :  T_end + 1);        % Cutting out rudder deflection of sweep

        sweep.throttle.output.roll(:,num)  = database1.output.roll(T_start + 1 :  T_end + 1);          % Cutting out roll angle of sweep
        sweep.throttle.output.pitch(:,num) = database1.output.pitch(T_start + 1 :  T_end + 1);         % Cutting out pitch angle of sweep
        sweep.throttle.output.yaw(:,num)   = database1.output.yaw(T_start + 1 :  T_end + 1);           % Cutting out yaw angle of sweep

        sweep.throttle.output.p(:,num)     = database1.output.p(T_start + 1 :  T_end + 1);             % Cutting out roll speed of sweep
        sweep.throttle.output.q(:,num)     = database1.output.q(T_start + 1 :  T_end + 1);             % Cutting out pitch speed of sweep
        sweep.throttle.output.r(:,num)     = database1.output.r(T_start + 1 :  T_end + 1);             % Cutting out yaw speed of sweep

        sweep.throttle.output.ax(:,num)    = database1.output.ax(T_start + 1 :  T_end + 1);            % Cutting out acceleration in X direction of sweep
        sweep.throttle.output.ay(:,num)    = database1.output.ay(T_start + 1 :  T_end + 1);            % Cutting out acceleration in Y direction of sweep
        sweep.throttle.output.az(:,num)    = database1.output.az(T_start + 1 :  T_end + 1);            % Cutting out acceleration in Z direction of sweep

        sweep.throttle.output.Vx(:,num)    = database1.output.Vx(T_start + 1 :  T_end + 1);            % Cutting out speed in X direction of sweep
        sweep.throttle.output.Vy(:,num)    = database1.output.Vy(T_start + 1 :  T_end + 1);            % Cutting out speed in Y direction of sweep
        sweep.throttle.output.Vz(:,num)    = database1.output.Vz(T_start + 1 :  T_end + 1);            % Cutting out speed in Z direction of sweep

        sweep.throttle.output.flightpath(:,num)      = database1.output.AOA(T_start + 1 :  T_end + 1);     % Cutting out flightpath angle of sweep
        sweep.throttle.output.true_airspeed(:,num)   = database1.output.true_airspeed(T_start + 1 :  T_end + 1);    % Cutting out true airspeed of sweep
        sweep.throttle.output.temperature(:,num)     = database1.output.temperature(T_start+1: T_end+1);   % Cutting out temperature of sweep
        sweep.throttle.output.air_density(:,num)     = database1.output.air_density(T_start+1: T_end+1);   % Cutting out air density of sweep
        sweep.throttle.output.air_pressure(:,num)    = database1.output.air_pressure(T_start+1: T_end+1);  % Cutting out air pressure of sweep
        sweep.throttle.output.height(:,num)          = database1.output.height(T_start+1: T_end+1);        % Cutting out height of sweep
    else

        sweep.throttle.time(:,num)           = database2.time(T_start + 1 :  T_end + 1);                % Cutting out time of sweep
        sweep.throttle.input.aileron(:,num)  = database2.input.aileron(T_start + 1 :  T_end + 1);       % Cutting out aileron deflection of sweep
        sweep.throttle.input.elevator(:,num) = database2.input.elevator(T_start + 1 :  T_end + 1);      % Cutting out elevator deflection of sweep
        sweep.throttle.input.throttle(:,num) = database2.input.throttle(T_start + 1 :  T_end + 1);      % Cutting out throttle input of sweep
        sweep.throttle.input.rudder(:,num)   = database2.input.rudder(T_start + 1 :  T_end + 1);        % Cutting out rudder deflection of sweep

        sweep.throttle.output.roll(:,num)  = database2.output.roll(T_start + 1 :  T_end + 1);
        sweep.throttle.output.pitch(:,num) = database2.output.pitch(T_start + 1 :  T_end + 1);
        sweep.throttle.output.yaw(:,num)   = database2.output.yaw(T_start + 1 :  T_end + 1);

        sweep.throttle.output.p(:,num)     = database2.output.p(T_start + 1 :  T_end + 1);             % Cutting out roll speed of sweep
        sweep.throttle.output.q(:,num)     = database2.output.q(T_start + 1 :  T_end + 1);             % Cutting out pitch speed of sweep
        sweep.throttle.output.r(:,num)     = database2.output.r(T_start + 1 :  T_end + 1);             % Cutting out yaw speed of sweep

        sweep.throttle.output.ax(:,num)    = database2.output.ax(T_start + 1 :  T_end + 1);            % Cutting out acceleration in X direction of sweep
        sweep.throttle.output.ay(:,num)    = database2.output.ay(T_start + 1 :  T_end + 1);            % Cutting out acceleration in Y direction of sweep
        sweep.throttle.output.az(:,num)    = database2.output.az(T_start + 1 :  T_end + 1);            % Cutting out acceleration in Z direction of sweep

        sweep.throttle.output.Vx(:,num)    = database2.output.Vx(T_start + 1 :  T_end + 1);            % Cutting out speed in X direction of sweep
        sweep.throttle.output.Vy(:,num)    = database2.output.Vy(T_start + 1 :  T_end + 1);            % Cutting out speed in Y direction of sweep
        sweep.throttle.output.Vz(:,num)    = database2.output.Vz(T_start + 1 :  T_end + 1);            % Cutting out speed in Z direction of sweep

        sweep.throttle.output.flightpath(:,num)   = database2.output.AOA(T_start + 1 :  T_end + 1);      % Cutting out flightpath angle of sweep
        sweep.throttle.output.true_airspeed(:,num)   = database2.output.true_airspeed(T_start + 1 :  T_end + 1);    % Cutting out true airspeed of sweep
        sweep.throttle.output.temperature(:,num)     = database2.output.temperature(T_start+1: T_end+1);   % Cutting out temperature of sweep
```

```
        sweep.throttle.output.air_density(:,num)     = database2.output.air_density(T_start+1: T_end+1);    % Cutting out air density of sweep
        sweep.throttle.output.air_pressure(:,num)    = database2.output.air_pressure(T_start+1: T_end+1);   % Cutting out air pressure of sweep
        sweep.throttle.output.height(:,num)          = database2.output.height(T_start+1: T_end+1);         % Cutting out height of sweep
    end
end

% Elevator sweep
for num =  1:1%length(sweep.elevator.frame(:,1))
    T_start = round(10*sweep.elevator.frame(num,1));              % Number in the array start
    T_end   = round(10*sweep.elevator.frame(num,2));              % Number in the array end

    if sweep.elevator.frame(num,3) == 1
        sweep.elevator.time(:,num)           = database1.time(T_start + 1 :  T_end + 1);                  % Cutting out time of sweep
        sweep.elevator.input.aileron(:,num)  = database1.input.aileron(T_start + 1 :  T_end + 1);         % Cutting out aileron deflection of sweep
        sweep.elevator.input.elevator(:,num) = database1.input.elevator(T_start + 1 :  T_end + 1);        % Cutting out elevator deflection of sweep
        sweep.elevator.input.throttle(:,num) = database1.input.throttle(T_start + 1 :  T_end + 1);        % Cutting out throttle input of sweep
        sweep.elevator.input.rudder(:,num)   = database1.input.rudder(T_start + 1 :  T_end + 1);          % Cutting out rudder deflection of sweep

        sweep.elevator.output.roll(:,num)  = database1.output.roll(T_start + 1 :  T_end + 1);            % Cutting out roll angle of sweep
        sweep.elevator.output.pitch(:,num) = database1.output.pitch(T_start + 1 :  T_end + 1);           % Cutting out pitch angle of sweep
        sweep.elevator.output.yaw(:,num)   = database1.output.yaw(T_start + 1 :  T_end + 1);             % Cutting out yaw angle of sweep

        sweep.elevator.output.p(:,num)     = database1.output.p(T_start + 1 :  T_end + 1);               % Cutting out roll speed of sweep
        sweep.elevator.output.q(:,num)     = database1.output.q(T_start + 1 :  T_end + 1);               % Cutting out pitch speed of sweep
        sweep.elevator.output.r(:,num)     = database1.output.r(T_start + 1 :  T_end + 1);               % Cutting out yaw speed of sweep

        sweep.elevator.output.ax(:,num)    = database1.output.ax(T_start + 1 :  T_end + 1);              % Cutting out acceleration in X direction of sweep
        sweep.elevator.output.ay(:,num)    = database1.output.ay(T_start + 1 :  T_end + 1);              % Cutting out acceleration in Y direction of sweep
        sweep.elevator.output.az(:,num)    = database1.output.az(T_start + 1 :  T_end + 1);              % Cutting out acceleration in Z direction of sweep

        sweep.elevator.output.Vx(:,num)    = database1.output.Vx(T_start + 1 :  T_end + 1);              % Cutting out speed in X direction of sweep
        sweep.elevator.output.Vy(:,num)    = database1.output.Vy(T_start + 1 :  T_end + 1);              % Cutting out speed in Y direction of sweep
        sweep.elevator.output.Vz(:,num)    = database1.output.Vz(T_start + 1 :  T_end + 1);              % Cutting out speed in Z direction of sweep

        sweep.elevator.output.flightpath(:,num)   = database1.output.AOA(T_start + 1 :  T_end + 1);      % Cutting out flightpath angle of sweep
        sweep.elevator.output.true_airspeed(:,num)   = database1.output.true_airspeed(T_start + 1 :  T_end + 1);    % Cutting out true airspeed of sweep
        sweep.elevator.output.temperature(:,num)     = database1.output.temperature(T_start+1: T_end+1);   % Cutting out temperature of sweep
        sweep.elevator.output.air_density(:,num)     = database1.output.air_density(T_start+1: T_end+1);   % Cutting out air density of sweep
        sweep.elevator.output.air_pressure(:,num)    = database1.output.air_pressure(T_start+1: T_end+1);  % Cutting out air pressure of sweep
        sweep.elevator.output.height(:,num)          = database1.output.height(T_start+1: T_end+1);        % Cutting out height of sweep
    else

        sweep.elevator.time(:,num)           = database2.time(T_start + 1 :  T_end + 1);                  % Cutting out time of sweep
        sweep.elevator.input.aileron(:,num)  = database2.input.aileron(T_start + 1 :  T_end + 1);         % Cutting out aileron deflection of sweep
        sweep.elevator.input.elevator(:,num) = database2.input.elevator(T_start + 1 :  T_end + 1);        % Cutting out elevator deflection of sweep
        sweep.elevator.input.throttle(:,num) = database2.input.throttle(T_start + 1 :  T_end + 1);        % Cutting out throttle input of sweep
        sweep.elevator.input.rudder(:,num)   = database2.input.rudder(T_start + 1 :  T_end + 1);          % Cutting out rudder deflection of sweep

        sweep.elevator.output.roll(:,num)  = database2.output.roll(T_start + 1 :  T_end + 1);            % Cutting out roll angle of sweep
        sweep.elevator.output.pitch(:,num) = database2.output.pitch(T_start + 1 :  T_end + 1);           % Cutting out pitch angle of sweep
        sweep.elevator.output.yaw(:,num)   = database2.output.yaw(T_start + 1 :  T_end + 1);             % Cutting out yaw angle of sweep

        sweep.elevator.output.p(:,num)     = database2.output.p(T_start + 1 :  T_end + 1);               % Cutting out roll speed of sweep
        sweep.elevator.output.q(:,num)     = database2.output.q(T_start + 1 :  T_end + 1);               % Cutting out pitch speed of sweep
        sweep.elevator.output.r(:,num)     = database2.output.r(T_start + 1 :  T_end + 1);               % Cutting out yaw speed of sweep

        sweep.elevator.output.ax(:,num)    = database2.output.ax(T_start + 1 :  T_end + 1);              % Cutting out acceleration in X direction of sweep
        sweep.elevator.output.ay(:,num)    = database2.output.ay(T_start + 1 :  T_end + 1);              % Cutting out acceleration in Y direction of sweep
        sweep.elevator.output.az(:,num)    = database2.output.az(T_start + 1 :  T_end + 1);              % Cutting out acceleration in Z direction of sweep

        sweep.elevator.output.Vx(:,num)    = database2.output.Vx(T_start + 1 :  T_end + 1);              % Cutting out speed in X direction of sweep
        sweep.elevator.output.Vy(:,num)    = database2.output.Vy(T_start + 1 :  T_end + 1);              % Cutting out speed in Y direction of sweep
        sweep.elevator.output.Vz(:,num)    = database2.output.Vz(T_start + 1 :  T_end + 1);              % Cutting out speed in Z direction of sweep

        sweep.elevator.output.flightpath(:,num)   = database2.output.AOA(T_start + 1 :  T_end + 1);      % Cutting out flightpath angle of sweep
        sweep.elevator.output.true_airspeed(:,num)   = database2.output.true_airspeed(T_start + 1 :  T_end + 1);    % Cutting out true airspeed of sweep
        sweep.elevator.output.temperature(:,num)     = database2.output.temperature(T_start+1: T_end+1);   % Cutting out temperature of sweep
        sweep.elevator.output.air_density(:,num)     = database2.output.air_density(T_start+1: T_end+1);   % Cutting out air density of sweep
        sweep.elevator.output.air_pressure(:,num)    = database2.output.air_pressure(T_start+1: T_end+1);  % Cutting out air pressure of sweep
        sweep.elevator.output.height(:,num)          = database2.output.height(T_start+1: T_end+1);        % Cutting out height of sweep
    end
end

for num =  2:2%length(sweep.elevator.frame(:,1))
    T_start = round(10*sweep.elevator.frame(num,1));              % Number in the array start
    T_end   = round(10*sweep.elevator.frame(num,2));              % Number in the array end

    if sweep.elevator.frame(num,3) == 1
        sweep.elevator.time2(:,1)           = database1.time(T_start + 1 :  T_end + 1);                  % Cutting out time of sweep
        sweep.elevator.input2.aileron(:,1)  = database1.input.aileron(T_start + 1 :  T_end + 1);         % Cutting out aileron deflection of sweep
        sweep.elevator.input2.elevator(:,1) = database1.input.elevator(T_start + 1 :  T_end + 1);        % Cutting out elevator deflection of sweep
        sweep.elevator.input2.throttle(:,1) = database1.input.throttle(T_start + 1 :  T_end + 1);        % Cutting out throttle input of sweep
        sweep.elevator.input2.rudder(:,1)   = database1.input.rudder(T_start + 1 :  T_end + 1);          % Cutting out rudder deflection of sweep

        sweep.elevator.output2.roll(:,1)  = database1.output.roll(T_start + 1 :  T_end + 1);            % Cutting out roll angle of sweep
        sweep.elevator.output2.pitch(:,1) = database1.output.pitch(T_start + 1 :  T_end + 1);           % Cutting out pitch angle of sweep
        sweep.elevator.output2.yaw(:,1)   = database1.output.yaw(T_start + 1 :  T_end + 1);             % Cutting out yaw angle of sweep

        sweep.elevator.output2.p(:,1)     = database1.output.p(T_start + 1 :  T_end + 1);               % Cutting out roll speed of sweep
        sweep.elevator.output2.q(:,1)     = database1.output.q(T_start + 1 :  T_end + 1);               % Cutting out pitch speed of sweep
        sweep.elevator.output2.r(:,1)     = database1.output.r(T_start + 1 :  T_end + 1);               % Cutting out yaw speed of sweep

        sweep.elevator.output2.ax(:,1)    = database1.output.ax(T_start + 1 :  T_end + 1);              % Cutting out acceleration in X direction of sweep
        sweep.elevator.output2.ay(:,1)    = database1.output.ay(T_start + 1 :  T_end + 1);              % Cutting out acceleration in Y direction of sweep
        sweep.elevator.output2.az(:,1)    = database1.output.az(T_start + 1 :  T_end + 1);              % Cutting out acceleration in Z direction of sweep

        sweep.elevator.output2.Vx(:,1)    = database1.output.Vx(T_start + 1 :  T_end + 1);              % Cutting out speed in X direction of sweep
        sweep.elevator.output2.Vy(:,1)    = database1.output.Vy(T_start + 1 :  T_end + 1);              % Cutting out speed in Y direction of sweep
        sweep.elevator.output2.Vz(:,1)    = database1.output.Vz(T_start + 1 :  T_end + 1);              % Cutting out speed in Z direction of sweep

        sweep.elevator.output2.flightpath(:,1)   = database1.output.AOA(T_start + 1 :  T_end + 1);      % Cutting out flightpath angle of sweep
        sweep.elevator.output2.true_airspeed(:,1)   = database1.output.true_airspeed(T_start + 1 :  T_end + 1);    % Cutting out true airspeed of sweep
        sweep.elevator.output2.temperature(:,1)     = database1.output.temperature(T_start+1: T_end+1);   % Cutting out temperature of sweep
        sweep.elevator.output2.air_density(:,1)     = database1.output.air_density(T_start+1: T_end+1);   % Cutting out air density of sweep
        sweep.elevator.output2.air_pressure(:,1)    = database1.output.air_pressure(T_start+1: T_end+1);  % Cutting out air pressure of sweep
        sweep.elevator.output2.height(:,1)          = database1.output.height(T_start+1: T_end+1);        % Cutting out height of sweep
    else

        sweep.elevator.time2(:,1)           = database2.time(T_start + 1 :  T_end + 1);                  % Cutting out time of sweep
        sweep.elevator.input2.aileron(:,1)  = database2.input.aileron(T_start + 1 :  T_end + 1);         % Cutting out aileron deflection of sweep
        sweep.elevator.input2.elevator(:,1) = database2.input.elevator(T_start + 1 :  T_end + 1);        % Cutting out elevator deflection of sweep
        sweep.elevator.input2.throttle(:,1) = database2.input.throttle(T_start + 1 :  T_end + 1);        % Cutting out throttle input of sweep
        sweep.elevator.input2.rudder(:,1)   = database2.input.rudder(T_start + 1 :  T_end + 1);          % Cutting out rudder deflection of sweep

        sweep.elevator.output2.roll(:,1)  = database2.output.roll(T_start + 1 :  T_end + 1);            % Cutting out roll angle of sweep
        sweep.elevator.output2.pitch(:,1) = database2.output.pitch(T_start + 1 :  T_end + 1);           % Cutting out pitch angle of sweep
        sweep.elevator.output2.yaw(:,1)   = database2.output.yaw(T_start + 1 :  T_end + 1);             % Cutting out yaw angle of sweep

        sweep.elevator.output2.p(:,1)     = database2.output.p(T_start + 1 :  T_end + 1);               % Cutting out roll speed of sweep
        sweep.elevator.output2.q(:,1)     = database2.output.q(T_start + 1 :  T_end + 1);               % Cutting out pitch speed of sweep
        sweep.elevator.output2.r(:,1)     = database2.output.r(T_start + 1 :  T_end + 1);               % Cutting out yaw speed of sweep

        sweep.elevator.output2.ax(:,1)    = database2.output.ax(T_start + 1 :  T_end + 1);              % Cutting out acceleration in X direction of sweep
```

```
        sweep.elevator.output2.ay(:,1)    = database2.output.ay(T_start + 1 :  T_end + 1);                % Cutting out acceleration in Y direction of sweep
        sweep.elevator.output2.az(:,1)    = database2.output.az(T_start + 1 :  T_end + 1);                % Cutting out acceleration in Z direction of sweep

        sweep.elevator.output2.Vx(:,1)    = database2.output.Vx(T_start + 1 :  T_end + 1);                % Cutting out speed in X direction of sweep
        sweep.elevator.output2.Vy(:,1)    = database2.output.Vy(T_start + 1 :  T_end + 1);                % Cutting out speed in Y direction of sweep
        sweep.elevator.output2.Vz(:,1)    = database2.output.Vz(T_start + 1 :  T_end + 1);                % Cutting out speed in Z direction of sweep

        sweep.elevator.output2.flightpath(:,1)   = database2.output.AOA(T_start + 1 :  T_end + 1);         % Cutting out flightpath angle of sweep
        sweep.elevator.output2.true_airspeed(:,1)   = database2.output.true_airspeed(T_start + 1 :  T_end + 1);    % Cutting out true airspeed of sweep
        sweep.elevator.output2.temperature(:,1)     = database2.output.temperature(T_start+1: T_end+1);    % Cutting out temperature of sweep
        sweep.elevator.output2.air_density(:,1)     = database2.output.air_density(T_start+1: T_end+1);    % Cutting out air density of sweep
        sweep.elevator.output2.air_pressure(:,1)    = database2.output.air_pressure(T_start+1: T_end+1);   % Cutting out air pressure of sweep
        sweep.elevator.output2.height(:,1)          = database2.output.height(T_start+1: T_end+1);         % Cutting out height of sweep
    end
end

for num =  3:3%length(sweep.elevator.frame(:,1))
    T_start = round(10*sweep.elevator.frame(num,1));              % Number in the array start
    T_end   = round(10*sweep.elevator.frame(num,2));              % Number in the array end

    if sweep.elevator.frame(num,3) == 1
        sweep.elevator.time3(:,1)           = databasel.time(T_start + 1 :  T_end + 1);                  % Cutting out time of sweep
        sweep.elevator.input3.aileron(:,1)  = databasel.input.aileron(T_start + 1 :  T_end + 1);         % Cutting out aileron deflection of sweep
        sweep.elevator.input3.elevator(:,1) = databasel.input.elevator(T_start + 1 :  T_end + 1);        % Cutting out elevator deflection of sweep
        sweep.elevator.input3.throttle(:,1) = databasel.input.throttle(T_start + 1 :  T_end + 1);        % Cutting out throttle input of sweep
        sweep.elevator.input3.rudder(:,1)   = databasel.input.rudder(T_start + 1 :  T_end + 1);          % Cutting out rudder deflection of sweep

        sweep.elevator.output3.roll(:,1)  = databasel.output.roll(T_start + 1 :  T_end + 1);             % Cutting out roll angle of sweep
        sweep.elevator.output3.pitch(:,1) = databasel.output.pitch(T_start + 1 :  T_end + 1);            % Cutting out pitch angle of sweep
        sweep.elevator.output3.yaw(:,1)   = databasel.output.yaw(T_start + 1 :  T_end + 1);              % Cutting out yaw angle of sweep

        sweep.elevator.output3.p(:,1)     = databasel.output.p(T_start + 1 :  T_end + 1);                % Cutting out roll speed of sweep
        sweep.elevator.output3.q(:,1)     = databasel.output.q(T_start + 1 :  T_end + 1);                % Cutting out pitch speed of sweep
        sweep.elevator.output3.r(:,1)     = databasel.output.r(T_start + 1 :  T_end + 1);                % Cutting out yaw speed of sweep

        sweep.elevator.output3.ax(:,1)    = databasel.output.ax(T_start + 1 :  T_end + 1);               % Cutting out acceleration in X direction of sweep
        sweep.elevator.output3.ay(:,1)    = databasel.output.ay(T_start + 1 :  T_end + 1);               % Cutting out acceleration in Y direction of sweep
        sweep.elevator.output3.az(:,1)    = databasel.output.az(T_start + 1 :  T_end + 1);               % Cutting out acceleration in Z direction of sweep

        sweep.elevator.output3.Vx(:,1)    = databasel.output.Vx(T_start + 1 :  T_end + 1);               % Cutting out speed in X direction of sweep
        sweep.elevator.output3.Vy(:,1)    = databasel.output.Vy(T_start + 1 :  T_end + 1);               % Cutting out speed in Y direction of sweep
        sweep.elevator.output3.Vz(:,1)    = databasel.output.Vz(T_start + 1 :  T_end + 1);               % Cutting out speed in Z direction of sweep

        sweep.elevator.output3.flightpath(:,1)   = databasel.output.AOA(T_start + 1 :  T_end + 1);        % Cutting out flightpath angle of sweep
        sweep.elevator.output3.true_airspeed(:,1)   = databasel.output.true_airspeed(T_start + 1 :  T_end + 1);    % Cutting out true airspeed of sweep
        sweep.elevator.output3.temperature(:,1)     = databasel.output.temperature(T_start+1: T_end+1);    % Cutting out temperature of sweep
        sweep.elevator.output3.air_density(:,1)     = databasel.output.air_density(T_start+1: T_end+1);    % Cutting out air density of sweep
        sweep.elevator.output3.air_pressure(:,1)    = databasel.output.air_pressure(T_start+1: T_end+1);   % Cutting out air pressure of sweep
        sweep.elevator.output3.height(:,1)          = databasel.output.height(T_start+1: T_end+1);         % Cutting out height of sweep
    else

        sweep.elevator.time3(:,1)           = database2.time(T_start + 1 :  T_end + 1);                  % Cutting out time of sweep
        sweep.elevator.input3.aileron(:,1)  = database2.input.aileron(T_start + 1 :  T_end + 1);         % Cutting out aileron deflection of sweep
        sweep.elevator.input3.elevator(:,1) = database2.input.elevator(T_start + 1 :  T_end + 1);        % Cutting out elevator deflection of sweep
        sweep.elevator.input3.throttle(:,1) = database2.input.throttle(T_start + 1 :  T_end + 1);        % Cutting out throttle input of sweep
        sweep.elevator.input3.rudder(:,1)   = database2.input.rudder(T_start + 1 :  T_end + 1);          % Cutting out rudder deflection of sweep

        sweep.elevator.output3.roll(:,1)  = database2.output.roll(T_start + 1 :  T_end + 1);             % Cutting out roll angle of sweep
        sweep.elevator.output3.pitch(:,1) = database2.output.pitch(T_start + 1 :  T_end + 1);            % Cutting out pitch angle of sweep
        sweep.elevator.output3.yaw(:,1)   = database2.output.yaw(T_start + 1 :  T_end + 1);              % Cutting out yaw angle of sweep

        sweep.elevator.output3.p(:,1)     = database2.output.p(T_start + 1 :  T_end + 1);                % Cutting out roll speed of sweep
        sweep.elevator.output3.q(:,1)     = database2.output.q(T_start + 1 :  T_end + 1);                % Cutting out pitch speed of sweep
        sweep.elevator.output3.r(:,1)     = database2.output.r(T_start + 1 :  T_end + 1);                % Cutting out yaw speed of sweep

        sweep.elevator.output3.ax(:,1)    = database2.output.ax(T_start + 1 :  T_end + 1);               % Cutting out acceleration in X direction of sweep
        sweep.elevator.output3.ay(:,1)    = database2.output.ay(T_start + 1 :  T_end + 1);               % Cutting out acceleration in Y direction of sweep
        sweep.elevator.output3.az(:,1)    = database2.output.az(T_start + 1 :  T_end + 1);               % Cutting out acceleration in Z direction of sweep

        sweep.elevator.output3.Vx(:,1)    = database2.output.Vx(T_start + 1 :  T_end + 1);               % Cutting out speed in X direction of sweep
        sweep.elevator.output3.Vy(:,1)    = database2.output.Vy(T_start + 1 :  T_end + 1);               % Cutting out speed in Y direction of sweep
        sweep.elevator.output3.Vz(:,1)    = database2.output.Vz(T_start + 1 :  T_end + 1);               % Cutting out speed in Z direction of sweep

        sweep.elevator.output3.flightpath(:,1)   = database2.output.AOA(T_start + 1 :  T_end + 1);        % Cutting out flightpath angle of sweep
        sweep.elevator.output3.true_airspeed(:,1)   = database2.output.true_airspeed(T_start + 1 :  T_end + 1);    % Cutting out true airspeed of sweep
        sweep.elevator.output3.temperature(:,1)     = database2.output.temperature(T_start+1: T_end+1);    % Cutting out temperature of sweep
        sweep.elevator.output3.air_density(:,1)     = database2.output.air_density(T_start+1: T_end+1);    % Cutting out air density of sweep
        sweep.elevator.output3.air_pressure(:,1)    = database2.output.air_pressure(T_start+1: T_end+1);   % Cutting out air pressure of sweep
        sweep.elevator.output3.height(:,1)          = database2.output.height(T_start+1: T_end+1);         % Cutting out height of sweep
    end
end

%% Cut out doublets



%% Calculating reference values
mean_temperature  =
(mean(sweep.elevator.output.temperature)+mean(sweep.elevator.output2.temperature)+mean(sweep.elevator.output3.temperature)+mean(sweep.throttle.output.temperature)
)/4;              % Average of the temperature
mean_air_density  =
(mean(sweep.elevator.output.air_density)+mean(sweep.elevator.output2.air_density)+mean(sweep.elevator.output3.air_density)+mean(sweep.throttle.output.air_density)
)/4;              % Average of the air density
mean_air_pressure =
(mean(sweep.elevator.output.air_pressure)+mean(sweep.elevator.output2.air_pressure)+mean(sweep.elevator.output3.air_pressure)+mean(sweep.throttle.output.air_press
ure))/4;          % Average of the air pressure
mean_height       =
(mean(sweep.elevator.output.height)+mean(sweep.elevator.output2.height)+mean(sweep.elevator.output3.height)+mean(sweep.throttle.output.height)/4);
% Average of the height
```

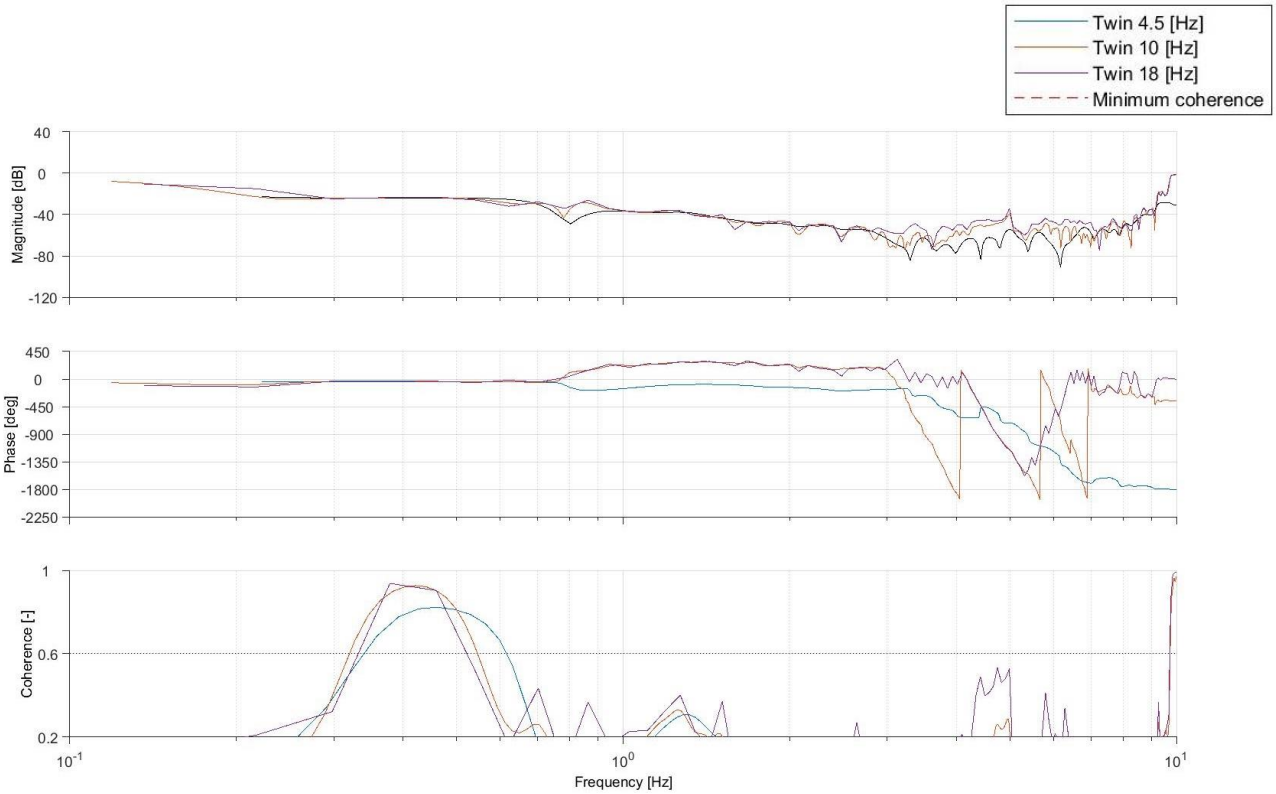
Freput.mat

```
%% Freput
% Stefan Juffermans
% 4174658

%% A program to make the files which are the input for the frequency analysis

%% Creating the elevator sweep file:

time = sweep.elevator.time;                    % Saving time [s]
elevator = sweep.elevator.input.elevator;      % Saving elevator deflection [deg]
throttle = sweep.elevator.input.throttle;      % Saving throttle setting [%]
pit = sweep.elevator.output.pitch;             % Saving pitch [deg]
q = sweep.elevator.output.q;                   % Saving pitch speed [deg/s]
u = sweep.elevator.output.Vx;                  % Saving speed in x direction [m/s]
w = sweep.elevator.output.Vz;                  % Saving speed in z direction [m/s]
ax = sweep.elevator.output.ax;                 % Saving acceleration in x direction [m/s^2]
az = sweep.elevator.output.az;                 % Saving acceleration in z direction [m/s^2]
```

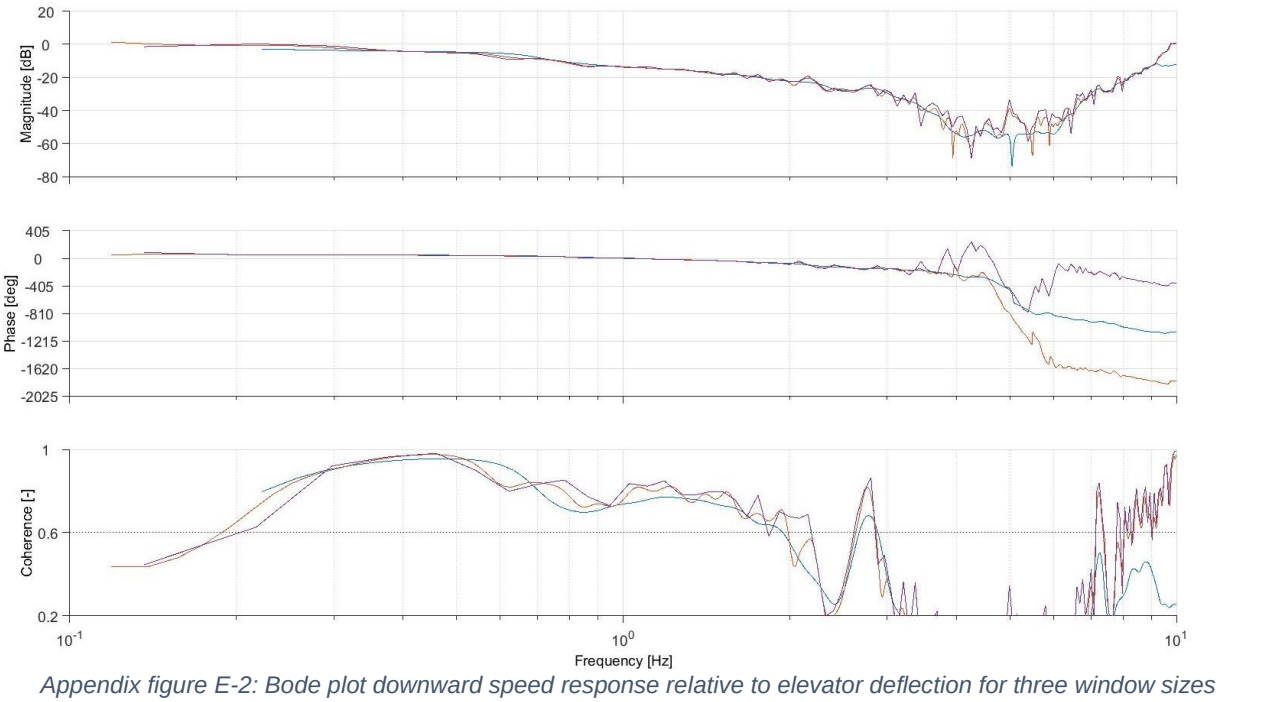
```
height = sweep.elevator.output.height;
temp = sweep.elevator.output.temperature;


save('elevator_sweep_skysurfer3.mat','time','elevator','pit','q','throttle','u','w','ax','az')  % Save all variables in one file for the analysis

time = sweep.elevator.time2;                    % Saving time [s]
elevator = sweep.elevator.input2.elevator;      % Saving elevator deflection [deg]
throttle = sweep.elevator.input2.throttle;      % Saving throttle setting [%]
pit = sweep.elevator.output2.pitch;             % Saving pitch [deg]
q = sweep.elevator.output2.q;                   % Saving pitch speed [deg/s]
u = sweep.elevator.output2.Vx;                  % Saving speed in x direction [m/s]
w = sweep.elevator.output2.Vz;                  % Saving speed in z direction [m/s]
ax = sweep.elevator.output2.ax;                 % Saving acceleration in x direction [m/s^2]
az = sweep.elevator.output2.az;                 % Saving acceleration in z direction [m/s^2]

save('elevator_sweep_skysurfer2.mat','time','elevator','pit','q','throttle','u','w','ax','az')  % Save all variables in one file for the analysis

time = sweep.elevator.time3;                    % Saving time [s]
elevator = sweep.elevator.input3.elevator;      % Saving elevator deflection [deg]
throttle = sweep.elevator.input3.throttle;      % Saving throttle setting [%]
pit = sweep.elevator.output3.pitch;             % Saving pitch [deg]
q = sweep.elevator.output3.q;                   % Saving pitch speed [deg/s]
u = sweep.elevator.output3.Vx;                  % Saving speed in x direction [m/s]
w = sweep.elevator.output3.Vz;                  % Saving speed in z direction [m/s]
ax = sweep.elevator.output3.ax;                 % Saving acceleration in x direction [m/s^2]
az = sweep.elevator.output3.az;                 % Saving acceleration in z direction [m/s^2]

save('elevator_sweep_skysurfer.mat','time','elevator','pit','q','throttle','u','w','ax','az')  % Save all variables in one file for the analysis

time = sweep.throttle.time;                     % Saving time [s]
elevator = sweep.throttle.input.elevator;       % Saving elevator deflection [deg]
throttle = sweep.throttle.input.throttle;       % Saving throttle setting [%]
pit = sweep.throttle.output.pitch;              % Saving pitch [deg]
q = sweep.throttle.output.q;                    % Saving pitch speed [deg/s]
u = sweep.throttle.output.Vx;                   % Saving speed in x direction [m/s]
w = sweep.throttle.output.Vz;                   % Saving speed in z direction [m/s]
ax = sweep.throttle.output.ax;                  % Saving acceleration in x direction [m/s^2]
az = sweep.throttle.output.az;                  % Saving acceleration in z direction [m/s^2]

save('throttle_sweep_skysurfer.mat','time','throttle','pit','q','elevator','u','w','ax','az')  % Save all variables in one file for the analysis
```

Appendix E Window length

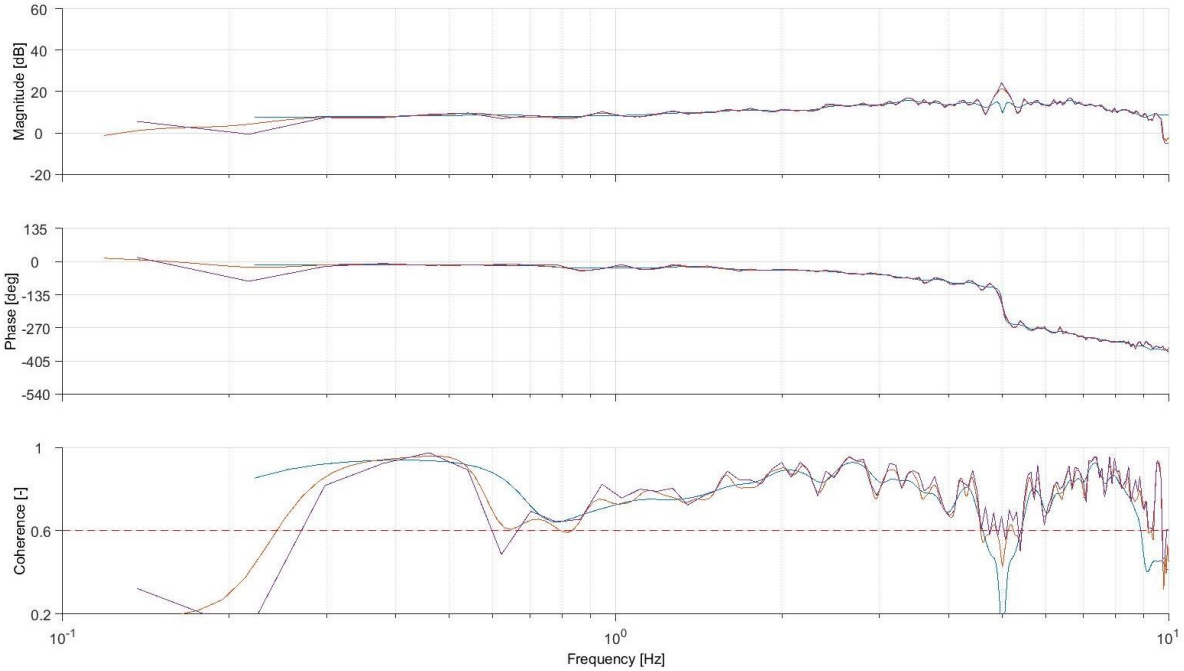
In Appendix figure E-1-Appendix figure E-4 are the responses calculated for three different window sizes. With increasing window size, the responses become more fluctuating while the coherence shows some improvement. Because of this, the window size of 10 Hz is chosen to increase the coherence slightly while keeping the signal stable. It can also be seen that the responses lay on top of each other where the coherence is high regardless of which window size is used.



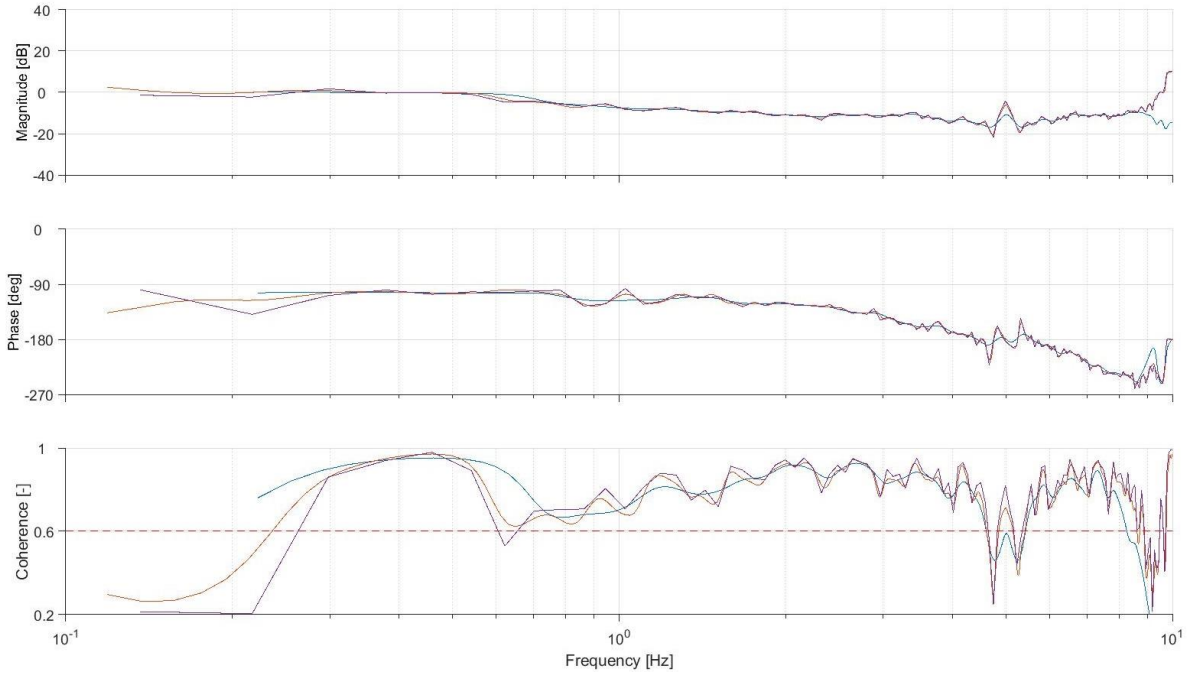
Appendix figure E-1: Bode plot forward speed response relative to elevator deflection for three window sizes



Appendix figure E-2: Bode plot downward speed response relative to elevator deflection for three window sizes



Appendix figure E-3: Bode plot pitch speed response relative to elevator deflection for three window sizes

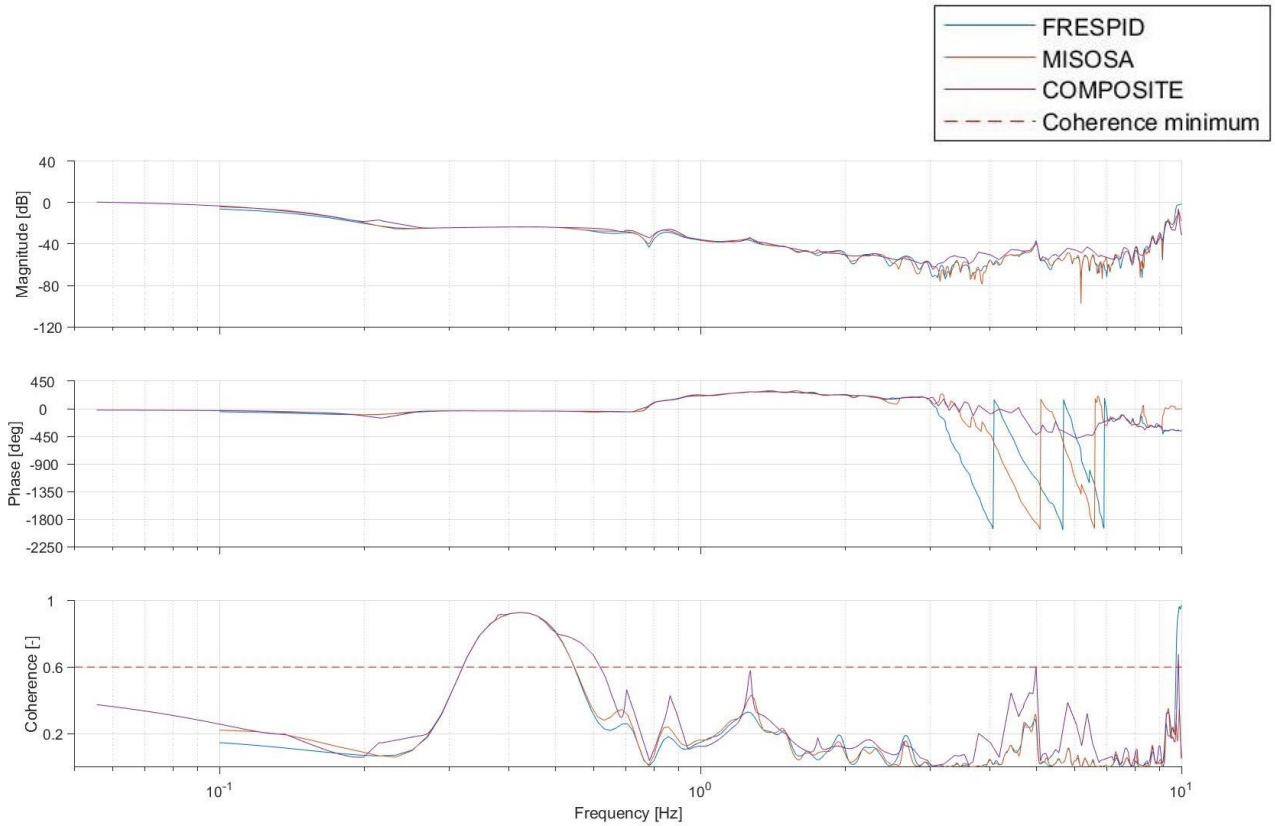


Appendix figure E-4: Bode plot pitch angle response relative to elevator deflection for three window sizes

Appendix F MIMOSA & COMPOSITE

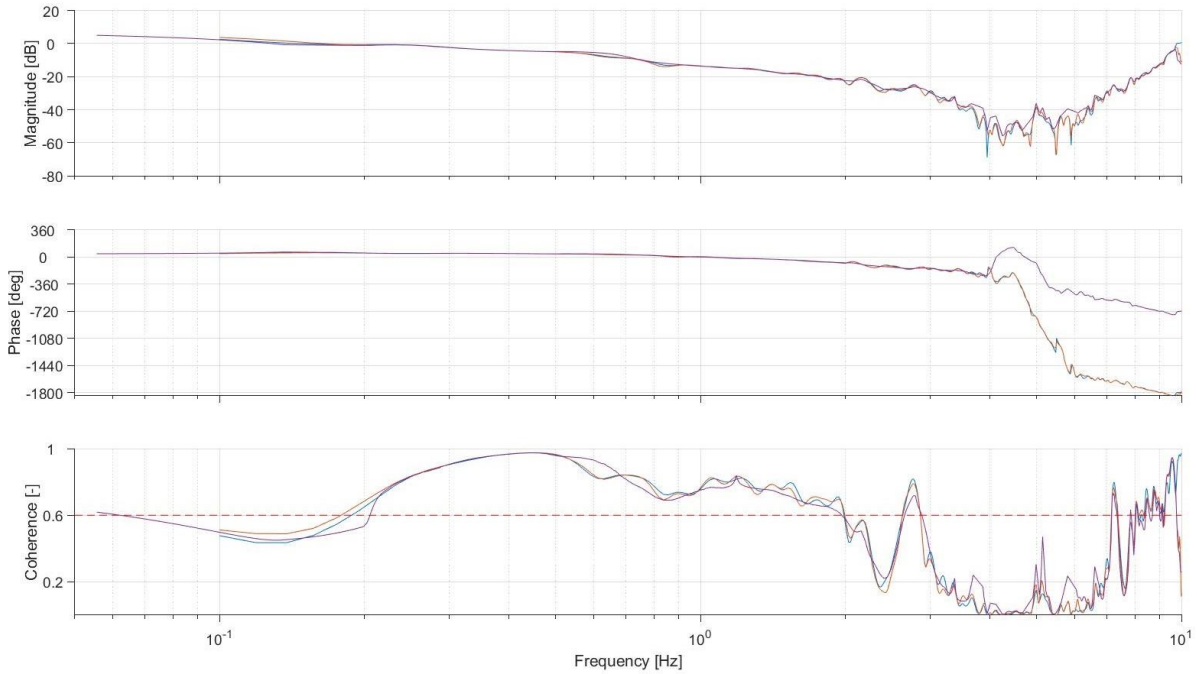
MISOSA ensures that the influence of the secondary inputs are extracted. If you take for example the elevator sweep, the influence of the throttle is eliminated from the main signal. This produces a response without the influence of other inputs. "In the COMPOSITE function, a unique windowing algorithm that generates various combinations selects the best results using an embedded coherence and error function calculation and integrates the optimal combination".[44] This unique windowing algorithm is based on an unconstrained optimization using quasi-Newton-Raphson methods[61].

Appendix figure F-1-Appendix figure F-4 shows the four elevator responses for FRESPID, MISOSA and COMPOSITE. The improvements of the programs is minimal in the magnitude and phase plot. The coherence therefore does show some improvement for the COMPOSITE case.

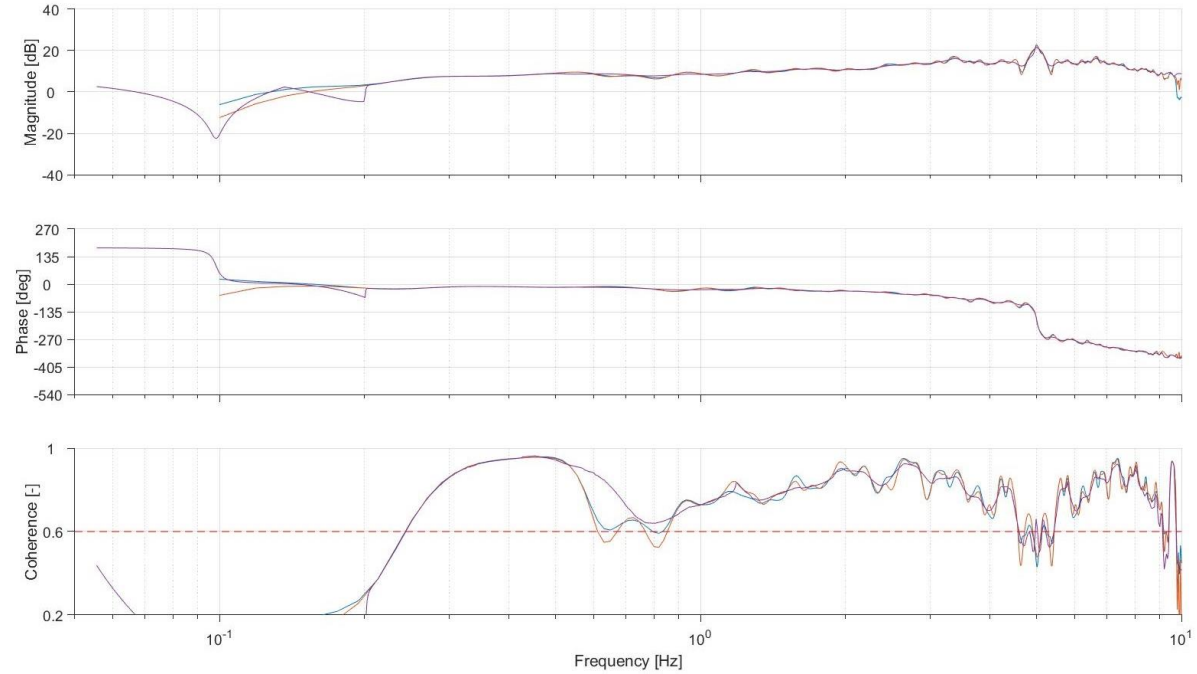


Appendix figure F-1: Forward speed response for FRESPID, MISOSA and COMPOSITE

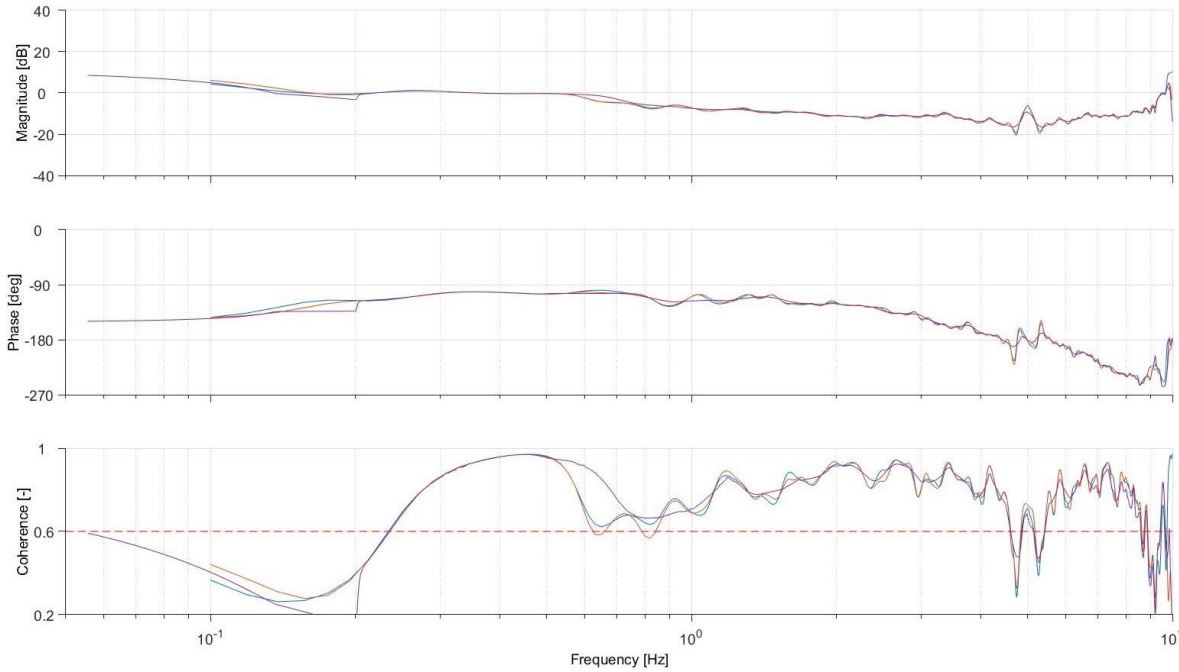
[44] http://casalcorp.com/images/CIFER_Users_Manual_V4-2-00_part_one_.pdf



Appendix figure F-2: Downward speed response for FRESPID, MISOSA and COMPOSITE



Appendix figure F-3: Pitch speed response for FRESPID, MISOSA and COMPOSITE



Appendix figure F-4: Pitch angle response for FRESPID, MISOSA and COMPOSITE

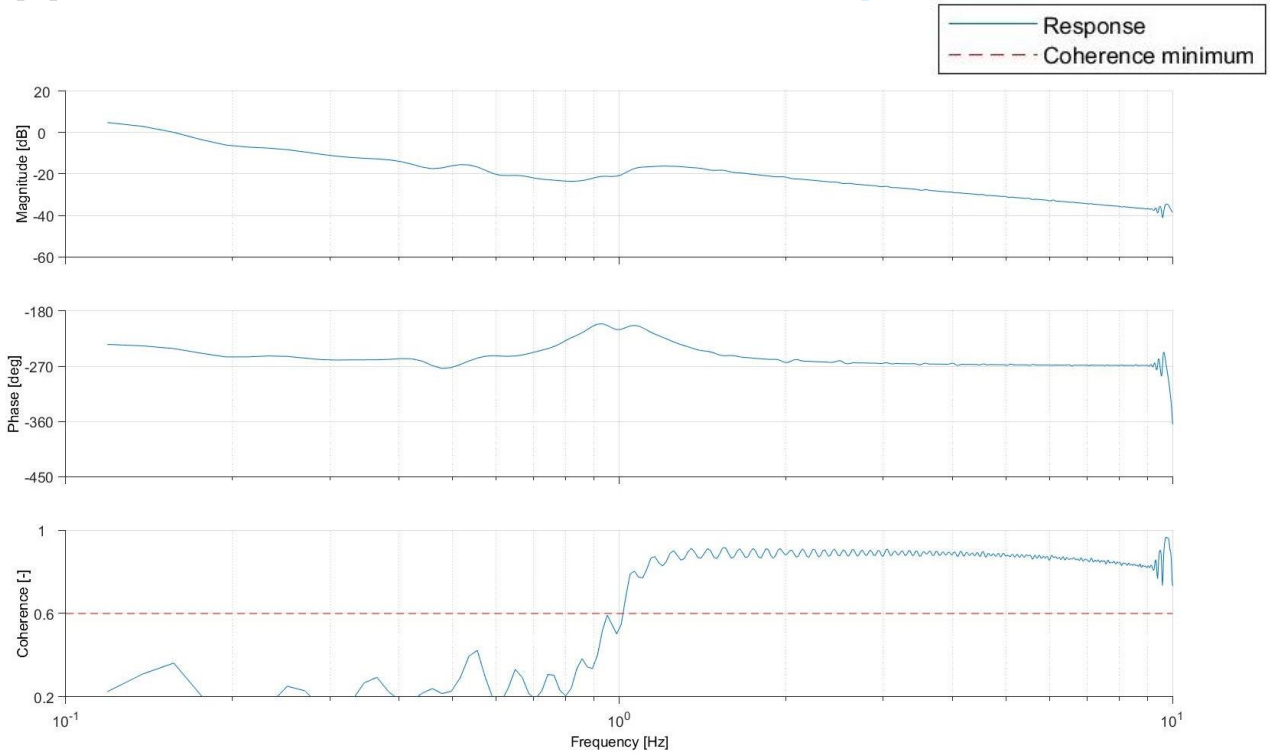
Appendix G MATLAB verification script

Verification.mat:

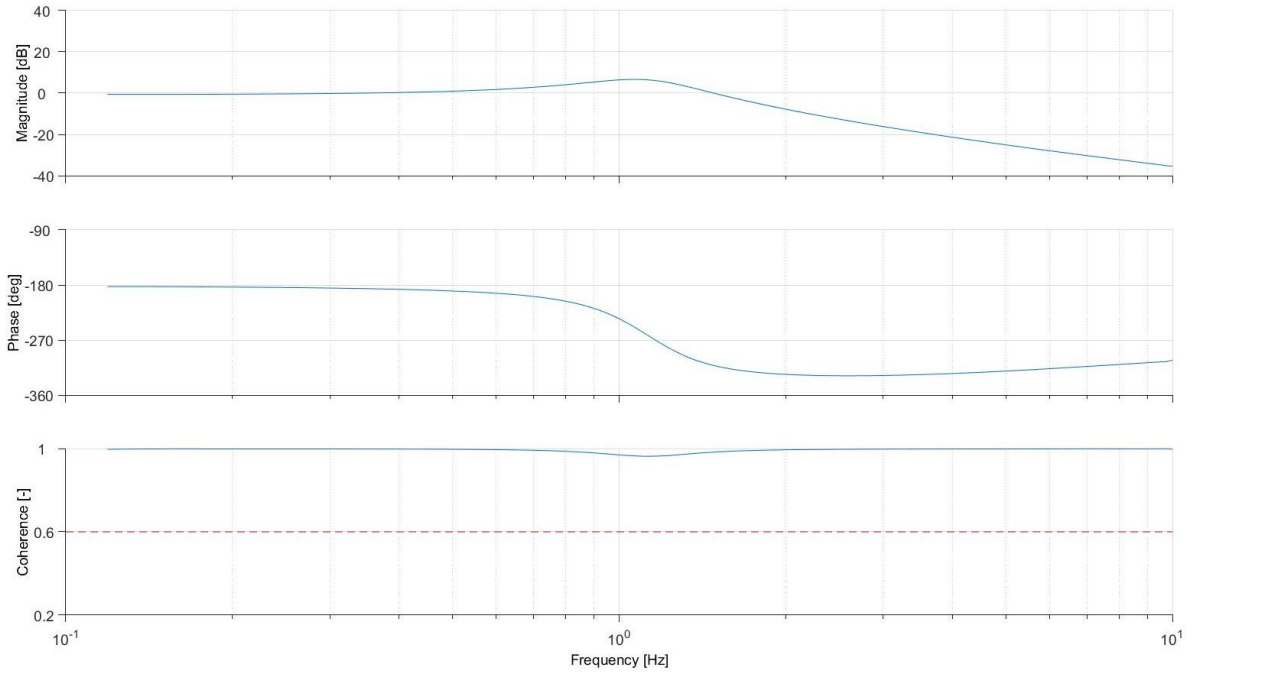
```
%% Verification
% Stefan Juffermans
% 4174658
% A program to generate frequency sweeps and get the input-output data in
% time domain from a predefined state space system

%% Constants for Sweep

f0 = 0.05;                % Start frequency [Hz]
f1 = 10;                  % End frequency [Hz]
t0  = 0:1/50:200;         % Time series [s]
t1 = 200;                 % End time of end frequency [s]

%% Generate sweep
y  = chirp(t0,f0,t1,f1,'quadratic');     % Sweep generations
t2=0:1/50:45;                            % Time series for constant part sweep
y2 = cos(2*pi*0.05*t2+2*pi*3.75);        % Constant part sweep

t=0:1/50:245;                            % New time series
y2(end) = [];                            % Delete double number

y= [y2,y];                               % Add constant part to sweep

%% State space variables

Xu   = 0.0028;
Xw   = -0.3088;
Xq   = -2.5278;
Xth  = -9.8088;
Xde  = -0.8951;
Xdt  = 0.2567;

Zu   = -0.1025;
Zw   = -2.8342;
Zq   = 132.3746;
Zth  = -0.1563;
Zde  = -0.9673;
Zdt  = 0;

Mu   = 0.0059;
Mw   = -0.3712;
Mq   = -0.1567;
Mth  =  0;
Mde  = -0.3406;
Mdt  = 0;

%% Generate state space system
A = [Xu, Xw, Xq, Xth;
     Zu, Zw, Zq, Zth;
     Mu, Mw, Mq, Mth;
     0 , 0 , 1 , 0];

B = [Xde, Xdt;
     Zde, Zdt;
     Mde, Mdt;
     0 ,  0 ];

C= [1,0,0,0;
    0,1,0,0;
    0,0,1,0;
    0,0,0,1];

D= [0,0;
```

```
    0,0;
    0,0;
    0,0];

sys = ss(A,B,C,D);

%% Generate elevator sweep input-output signals

elevator_sweep = y*5;                                   % Make sweep of 5 deg
elevator_sweep = elevator_sweep.';                      % Make it a column
throttle_constant = zeros(length(elevator_sweep),1);    % Constant  throttle sweep
throttle_constant = throttle_constant(:,1);             % Make it a column
time = t.';                                             % Make it a column
u= [elevator_sweep, throttle_constant];                 % Input for analysis

res_ele = lsim(sys,u,time,[0;0;0;0]);                   % Calculate the response

%% Generate the input file from the results

elevator = elevator_sweep;
throttle = throttle_constant;
time     = time;

u     = res_ele(:,1);
w     = res_ele(:,2);
theta = res_ele(:,4);
q     = res_ele(:,3);

save('elevator_sweep_stol.mat','time','elevator','throttle','u','w','theta','q');

%% Generate throttle sweep

throttle_sweep = y*0.25;                        % Throttle sweep of 25%
throttle_sweep = throttle_sweep.';              % Make a column
elevator_constant = throttle_constant;          % Constant elevator

u= [elevator_constant, throttle_sweep];         % Input signals

res_thr = lsim(sys,u,time,[0;0;0;0]);           % Calculate results

%% Generate input file from the results

elevator = elevator_constant;
throttle = throttle_sweep;
time     = time;

u     = res_thr(:,1);
w     = res_thr(:,2);
theta = res_thr(:,4);
q     = res_thr(:,3);

save('throttle_sweep_stol.mat','time','elevator','throttle','u','w','theta','q','Nz');
```

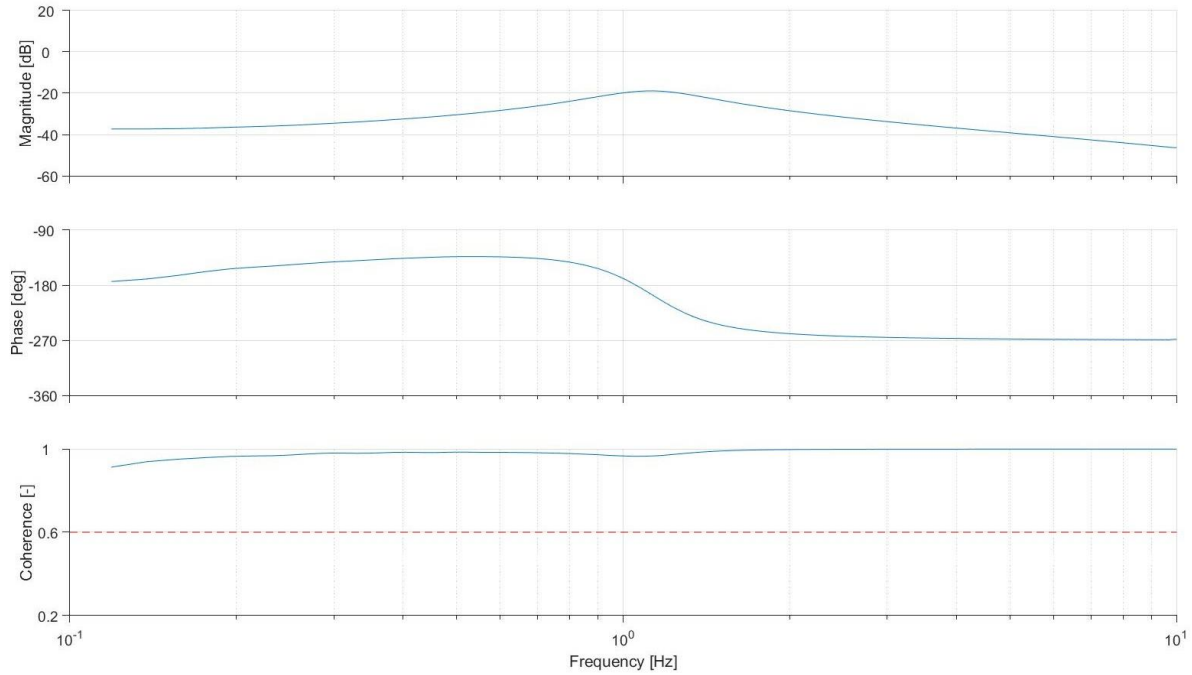
Appendix H Verification SISO responses



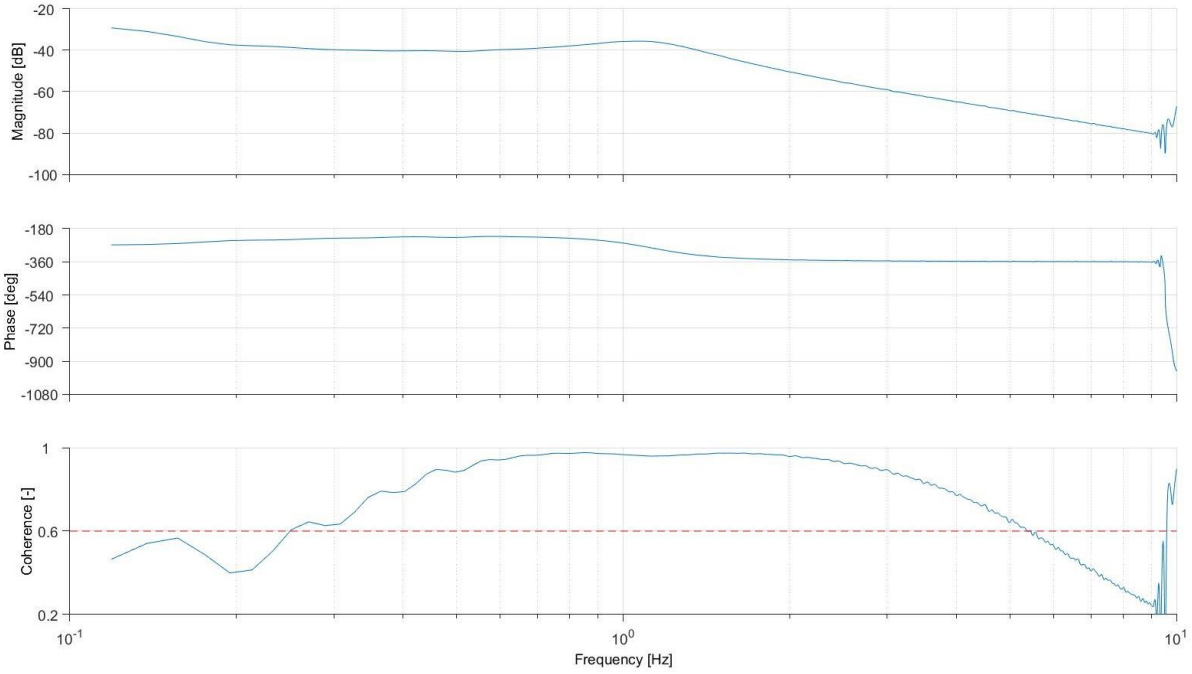
Appendix figure H-1: Forward speed response relative to elevator deflection



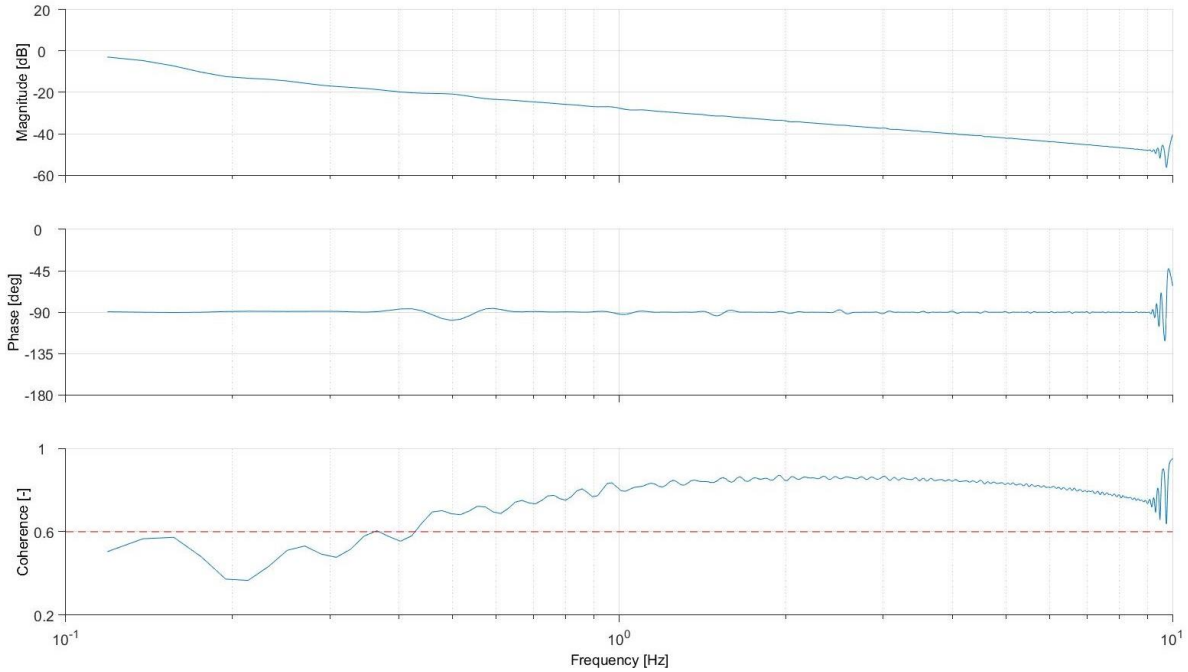
Appendix figure H-2: Downward speed response relative to elevator deflection



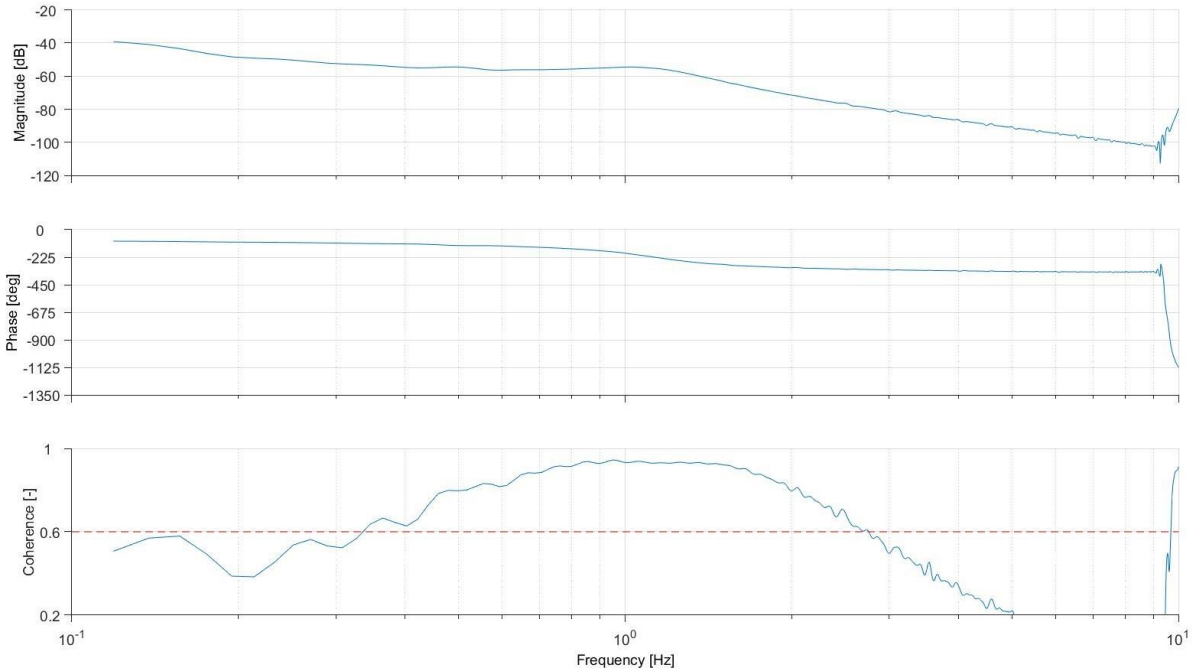
Appendix figure H-3: Pitch speed response relative to elevator deflection



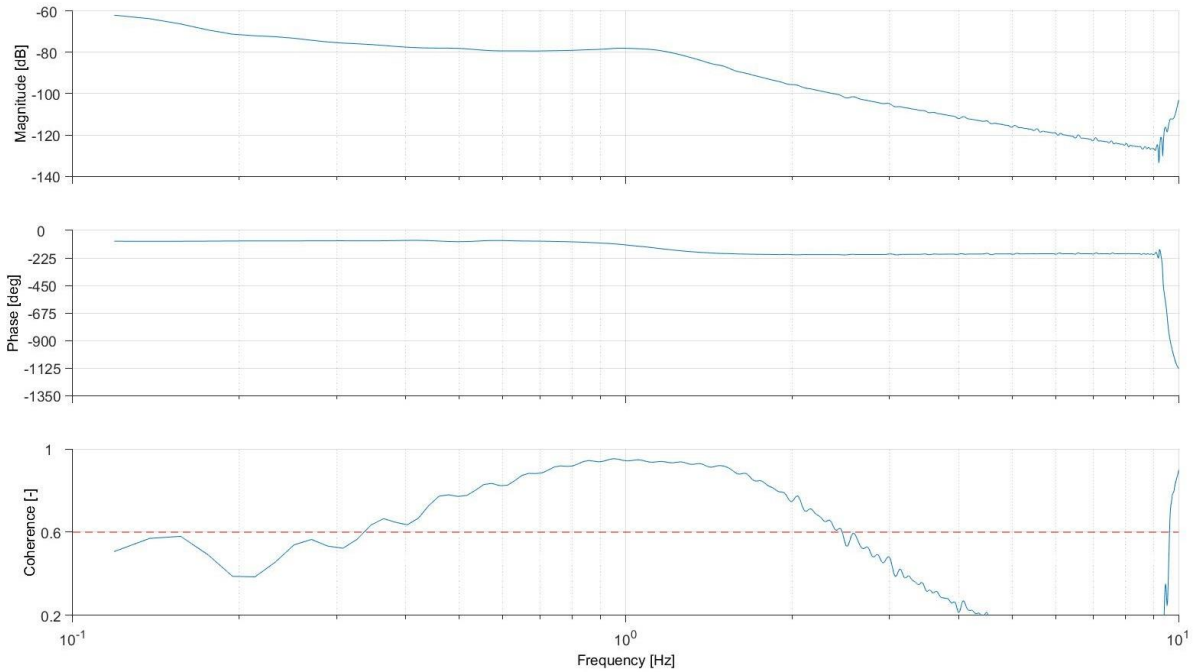
Appendix figure H-4: Pitch angle response relative to elevator deflection



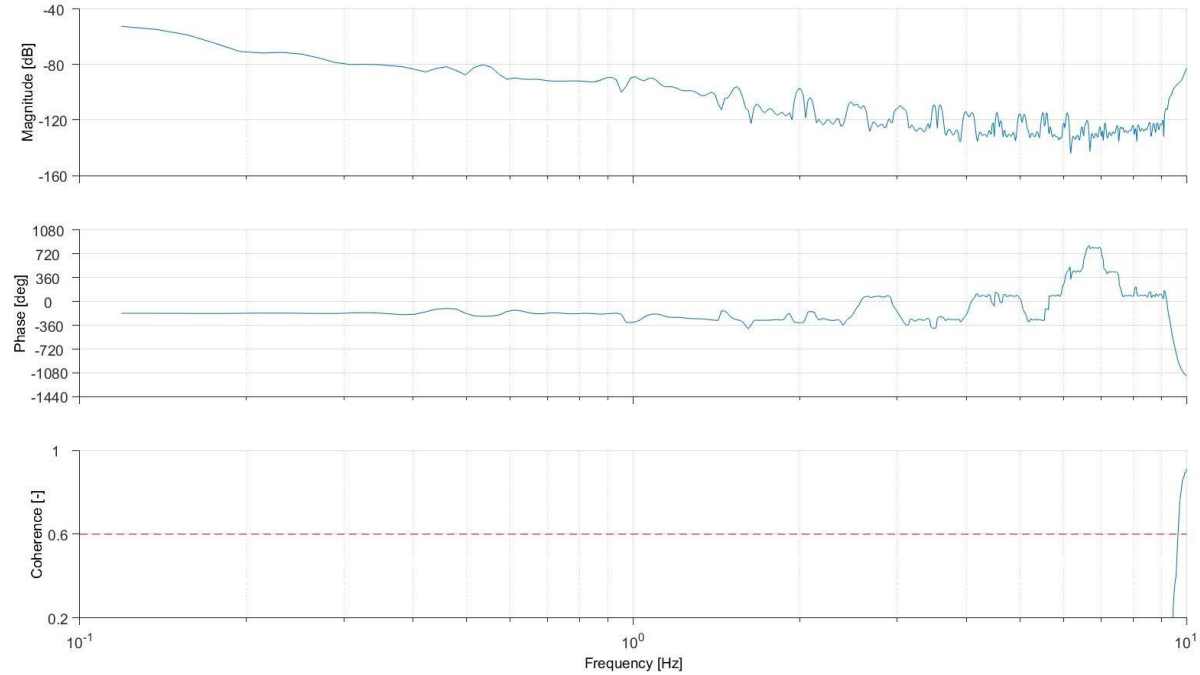
Appendix figure H-5: Forward speed response relative to throttle variation



Appendix figure H-6: Downward speed response relative to throttle variation

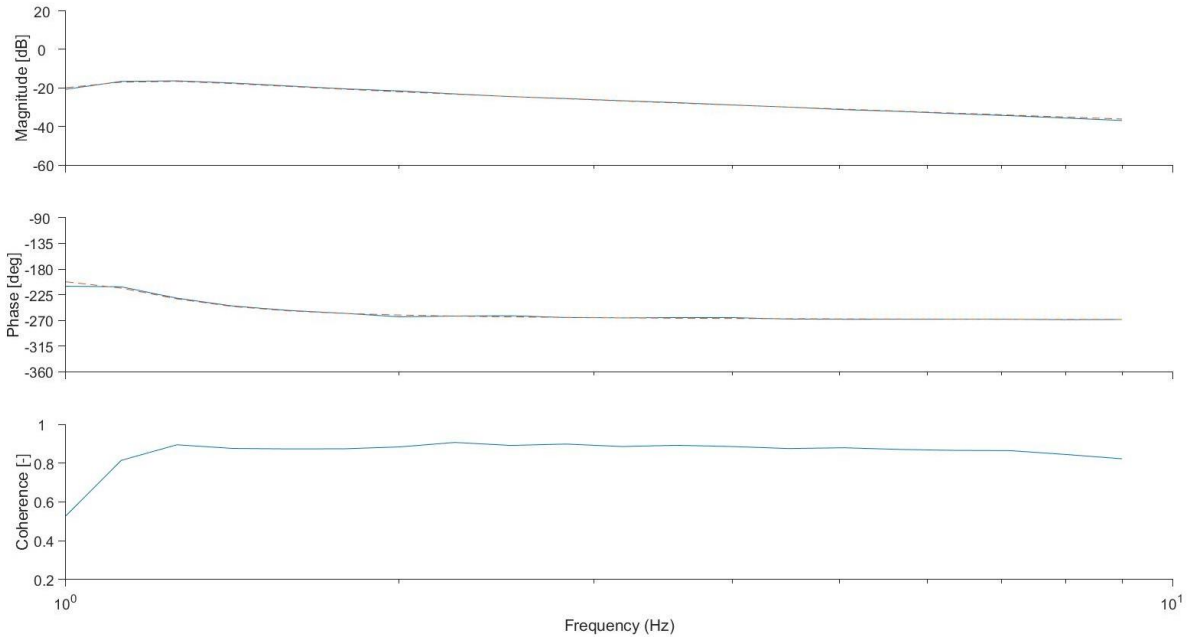


Appendix figure H-7: Pitch speed response relative to throttle variation

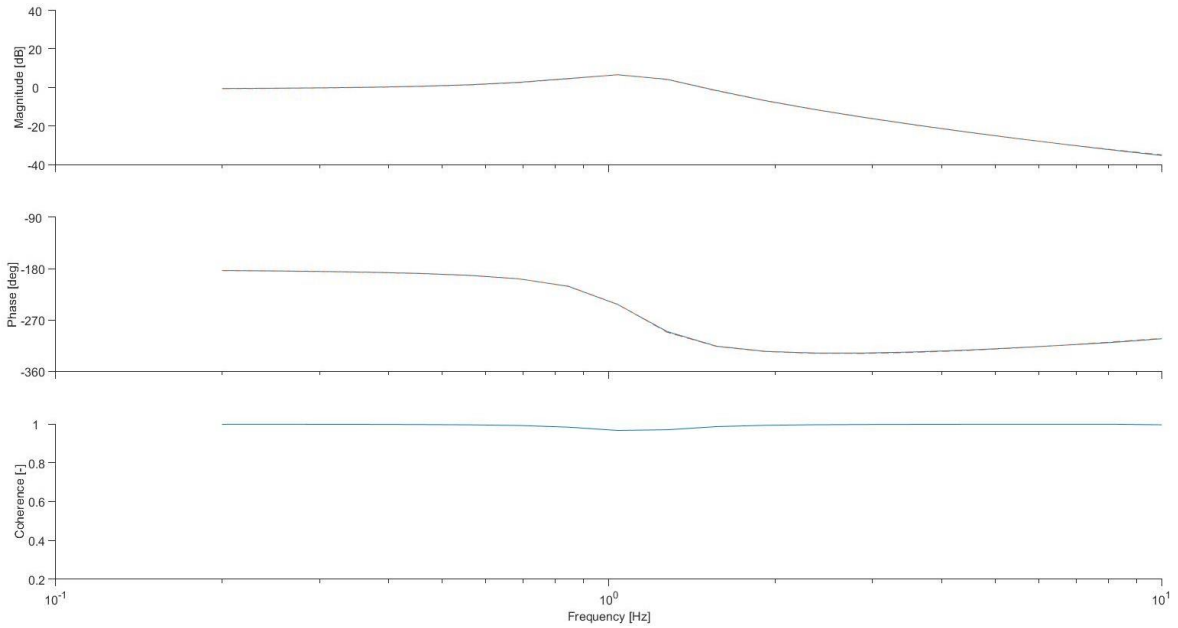


Appendix figure H-8: Pitch angle response relative to throttle variation

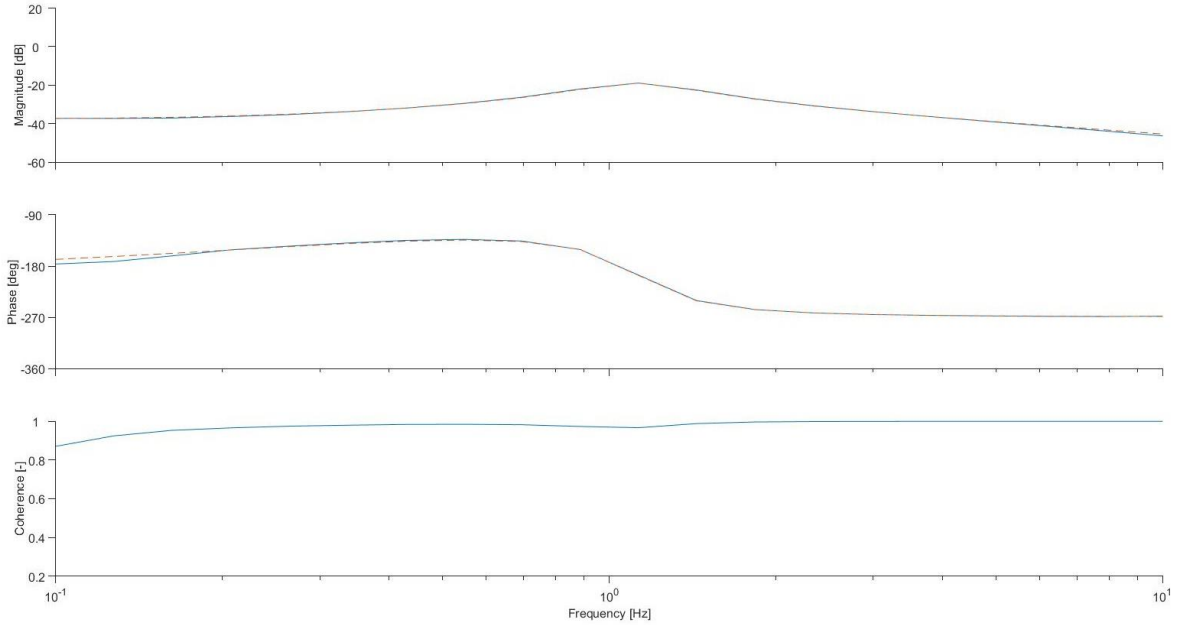
Appendix I Verification case MIMO bode plots



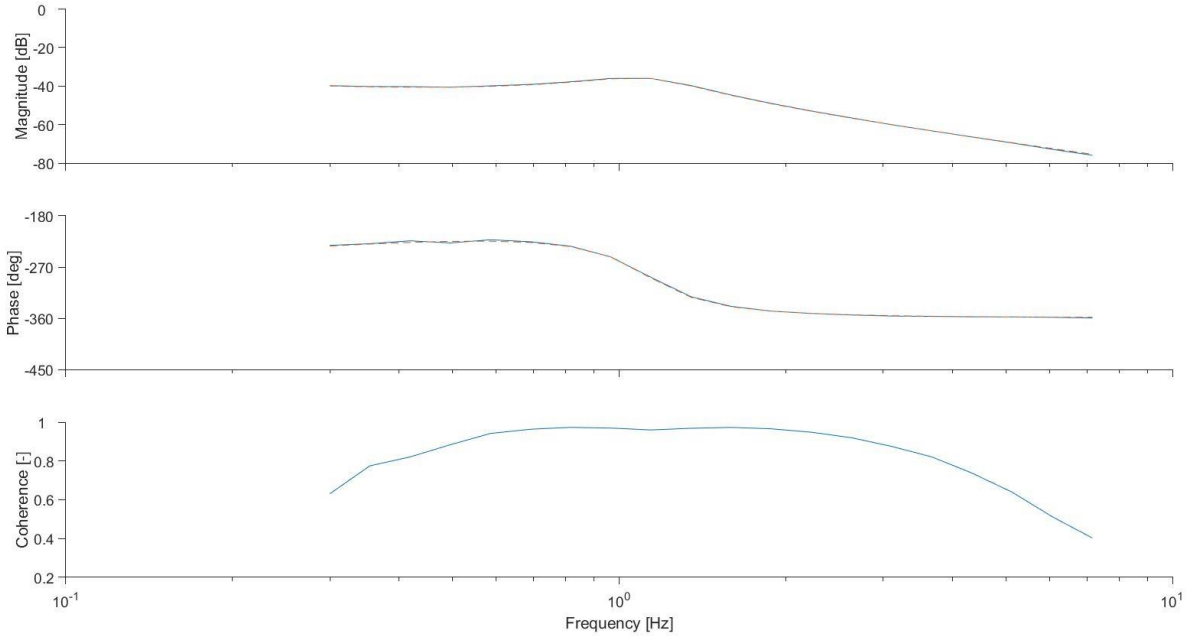
Appendix figure I-1: Forward speed response relative to elevator deflection fit



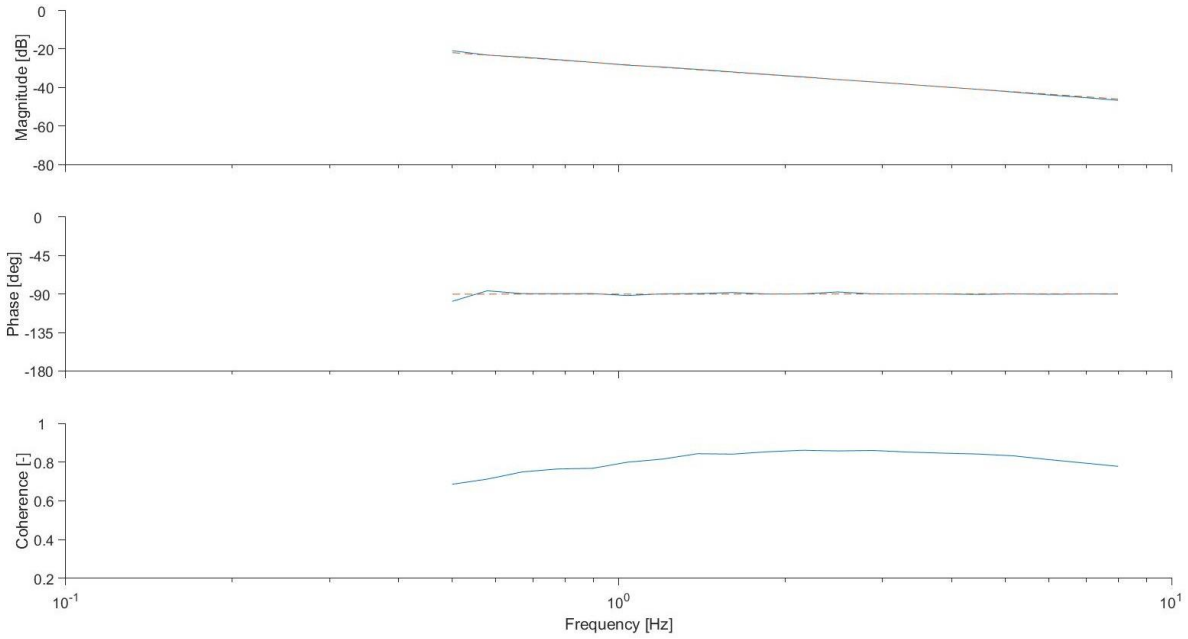
Appendix figure I-2: Downward speed response relative to elevator deflection fit



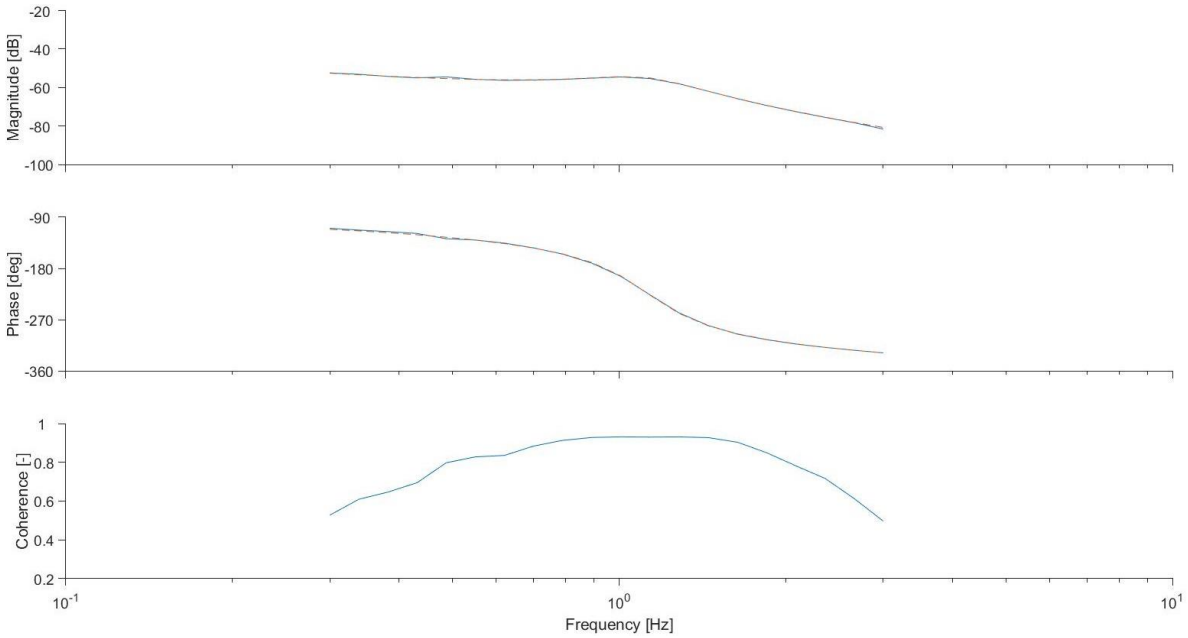
Appendix figure I-3: Pitch speed response relative to elevator deflection fit



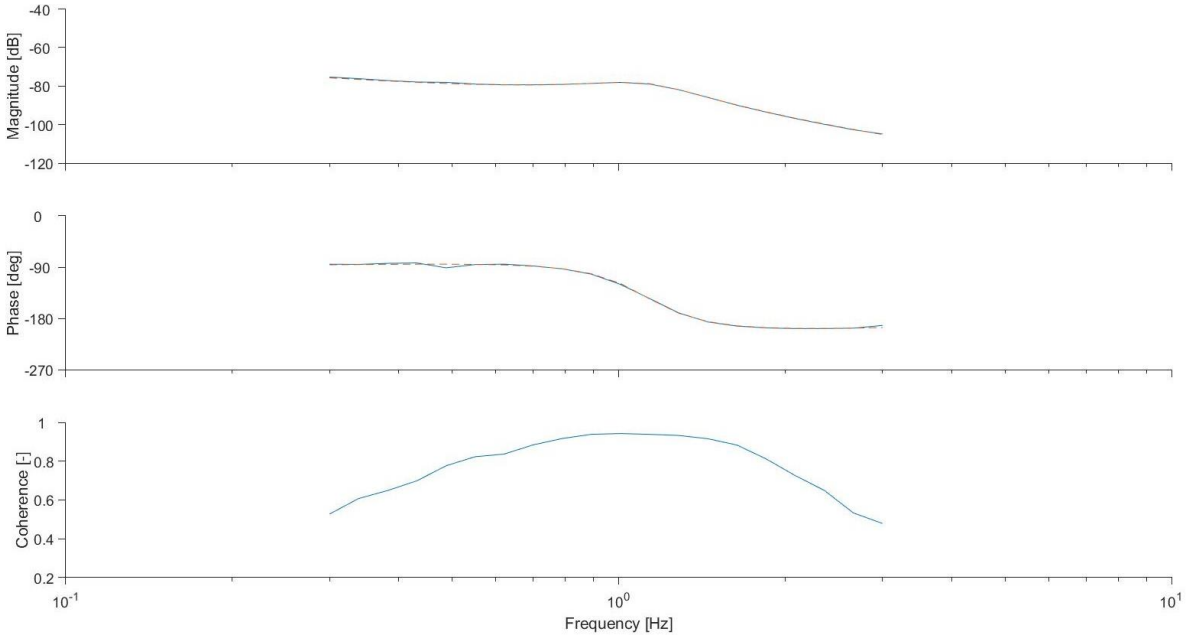
Appendix figure I-4: Pitch angle response relative to elevator deflection fit



Appendix figure I-5: Forward speed response relative to throttle change fit



Appendix figure I-6: Downward speed response relative to elevator deflection fit



Appendix figure I-7: Pitch speed response relative to elevator deflection fit