CHALMERS
UNIVERSITY OF TECHNOLOGY

Distributed mixed sensitivity UAV airspeed control

Evaluating the feasibility of mixed sensitivity airspeed control for distributed UAV control system

Master's thesis in Systems, Control and Mechatronics

BJÖRN LÖNNQUIST

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Distributed mixed sensitivity UAV airspeed control

Evaluating the feasibility of mixed sensitivity airspeed control for distributed UAV control system

BJÖRN LÖNNQUIST

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2023

Distributed mixed sensitivity UAV airspeed control
Evaluating the feasibility of mixed sensitivity airspeed control for distributed UAV control system
BJÖRN LÖNNQUIST

Supervisor: Lars-Berno Fredriksson, Kvaser
Examiner: Petter Falkman, Department of Electrical Engineering

Master's Thesis 2023
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Picture of resulting UAV with implemented control system.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Distributed mixed sensitivity UAV airspeed control
Evaluating the feasibility of mixed sensitivity airspeed control for distributed UAV control system
Björn Lönnquist
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Control systems for automotive, aerospace or any application can be synthesised and implemented in many ways. In some applications it is desirable to use a distributed control system, with measurement and actuation tasks being performed on several nodes communicating on a shared bus. With MIMO controllers this distribution poses challenges, as it is not trivial where and how different parts of a synthesised controller should be implemented. In this thesis a mixed sensitivity approach to airspeed control of a fixed wing UAV is explored, as well as exploring general requirements for distributing MIMO control systems. A model of the relevant dynamics of the UAV has been created in Simulink and the model has been compared to the flight characteristics of the physical UAV. A control approach using suitable actuators, elevator and throttle, has been created and verified in the simulation environment. To implement the synthesised controller in the available hardware CAN communication and RTOS tasks have been designed and scheduled. It has been found that MIMO mixed sensitivity control can be implemented in a distributed CAN control system achieving the same functionality in the complete distributed embedded system as in a simulated controller/plant model. In practical flight tests functionality has been confirmed, achieving reliable airspeed control, both subjectively and quantitatively. Performance in certain frequency ranges due to external factors, such as turbulence and model inaccuracies, makes direct flight comparisons to simulated response impractical. Integrating the developed airspeed control approach in a wider fully autonomous flight control scheme can be a suitable continuation.

Keywords: Mixed sensitivity controller, Distributed control system, Embedded Systems.

Acknowledgements

The initialisation, implementation and completion of this master could never have been done without the support of many individuals.

Firstly, I would like to thank Kvaser for the opportunity to embark on this journey with their support. My supervisor, Lars-Berno Fredriksson, has throughout the thesis provided excellent ongoing insights to the problems arising and helped contextualize it in a broader setting. For technical implementation questions I would like to especially thank Andre Idoffsson for his insight and expertise regarding the hardware used and Lars-Göran Fredriksson for his expertise and helpfulness regarding Kvasers commercial products. A further thank you goes out to all the wonderful colleagues at Kvaser which always welcomed me as an equal.

For the academic portion of the thesis I would like to sincerely thank my examiner Petter Falkman for his cooperation in enabling and overseeing my thesis. A warm thank you also goes to Simon Hansson for his contributions to this work.

Finally, I would like to thank my family and friends for their continuous support during my studies.

Björn Lönnquist, Gothenburg, February 2023

Contents

List of Figures

List of Tables

1
Introduction

Unmanned Aerial Vehicles are becoming more and more widespread, in industry and society as a whole. Many of the applications use centralised control units [1].

As functionality expands and complexity increases the utility of a distributed control system becomes appealing. When designing control systems with interconnected behaviour across several nodes challenges not present in an idealised continuous space arises. For a control engineer this distribution requires extra attention as information arriving from different nodes at different times effects for example stability and robustness. Furthermore the challenge of where in the system to add different parts of a synthesised controller is present.

Kvaser has a distributed system deployed in a fixed wing UAV with nodes controlling each control surface as well as the motor. Currently the control surfaces are controlled remotely by an operator. Moving forward autonomous features are desired. This project aims to add functionality such that the motor no longer needs to be manually controlled by the operator. Since many inputs and outputs interact in the dynamics of an aircraft, aircraft control becomes a complex control problem, with the need of simplifications and assumptions.

It can be reasoned that two main controllable inputs effect the airspeed, throttle and elevator. Increasing the throttle will increase the forward trust, thus increasing speed until a new equilibrium between thrust and drag has been reached. Tilting the plane downward will convert some of the potential energy into kinetic energy, hence increasing speed. Correspondingly, decreasing throttle or tilting the plane upwards will decrease speed. Therefore a control strategy which uses both throttle and elevator to control airspeed might be appropriate. Throttle actuation should be used to correct for steady state errors in airspeed, while elevator actuation may only be used for high frequency corrections as a constant offset on elevator would result in undesired climbing/descending. Some mixed sensitivity approach could be taken.

Addition of an automatic airspeed control does add dangers not present in a manually controlled aircraft, especially on the ground with people close to the motor. Operation of the automatic airspeed control should not cause a hazardous environment for the operator or other persons. Hence accidental or unsafe activation of an automatic flight mode should not be allowed.

2
Background

Humanity has since its first steps always sought to improve the quality of life for one self and others. In the last centuries technology has advanced at an ever accelerating rate, and in the last decades computerisation has swept across the globe. To remove humans from harmful situations and enable higher productivity, moving vehicles without human operators have been introduced. One example are Unmanned Aerial Vehicles which are becoming more and more widespread, in industry and society as a whole. Many of the applications currently use centralised control units.

As functionality expands and complexity increases the utility of a distributed control system becomes appealing. When designing control systems with interconnected behaviour across several nodes challenges not present in an idealised continuous space arises. This distribution requires extra attention as information arriving from different nodes at different times effects for example stability and robustness. Furthermore the challenge of where in the system to add different parts of a synthesised controller is present.

While some mixed sensitivity controller have been designed for UAVs, both for general navigation [2] and airspeed control specifically [3], actual flight verification is not so well documented. Furthermore, in [3] frequency dependent behaviours for actuation, error and disturbances are largely unexplored. When it comes to the implementation of mixed sensitivity controllers, or any controllers for that matter, in distributed control systems for UAVs the area seems largely unexplored(or at least undocumented). All that was found in a intial litterature study was related to distributed control systems for UAVs was in regard to high level planning within cooperation or swarming, and not low level flight dynamics. Distributed control systems have significant prevalence in other areas such as automotive, exemplified here for a fuel cell drivetrain [4].

3
Theory

3.1 Mixed sensitivty H_∞ controller synthesis

Many control problems can be visualised with a simple single input and single output model, an example could be a pendulum with one degree of freedom. The input of torque on the axle will directly influence the position of the pendulum, and the pendulum will only be influenced by the input torque. In a case where the pendulum has two degrees of freedom and two input torques it is no longer trivial how the inputs effect each other and the motion of the pendulum. Or, in the context of airplane speed control, what (controllable) inputs affect the speed of UAV. To provide a structured approach to the control of systems with more than one input several approaches can be taken[5]. Creating a MIMO process model with a multi dimensional transfer function can consist of a matrix of transfer functions, with each transfer function describing the relationship between each input and output. Just as in the one dimensional case a sensitivity and complementary sensitivity function of the controller-plant model can be created, although in matrix form. The sensitivity function is the transfer function from a disturbance to output. If the disturbance is a low frequency signal, such as a steady state offset, the maximum singular value of the sensitivity function should be low in this low frequency range. To achieve this sensitivity function the goal is to minimise the function $\|\omega_1 S\|_\infty$ where ω_1 being a low pass filter and S being the sensitivity function. Since ω_1 is a low pass filter, with suitable cutoff frequency, the synthesis of the controller will be weighted towards lowering the sensitivity function for low frequencies. It should be noted that the weights ω_i must be stable, and as such pure integration(giving infinite weight to low frequencies), is not allowed.

After a transfer function of the dynamics of the plant has been created and suitable transfer functions for the weights have been chosen a systematic approach to minimising $\|\omega S\|_\infty$ is done. This optimization can be done using μ-synthesis[5], in a DK-iteration. As it is an iterative procedure it is not guaranteed that the solution will reach the true minimum, and it is often desirable with a controller not reaching the true optimum since the slight increase in optimisation cost often gives a more aggressive high-frequency attenuation.

3.2 Kvaser similarities

Previous to this thesis several other master theses have been carried out in co-operation with Kvaser. Although the different theses have had different topics of

discussion, shared elements include the usage of a CAN bus, the approach to designing distributed systems CANKingdom, and the usage of the RTOS RT-Kernel. Due to these similarities in shared technologies, the sections 3.2.1, 3.2.2, 3.2.2.1 and 3.2.3 will, after some modification, be borrowed from the previous work conducted by Anton Olsson and Robert Sjöberg [6].

3.2.1 Controller area network

CAN is a communication protocol first published in 1986 by Robert Bosch GmbH [7]. It describes a way for multiple nodes to access a common bus that supports distributed real-time control [8]. The CAN protocol has a maximum transfer speed of 1 Mbit/s. Communication is done through CAN messages which consist of an identifier (ID), a control field, and a data payload. The ID for a CAN message must be unique in the sense that an ID can only be used by one node but a node can use multiple IDs. The ID serves three purposes: (1) it identifies the message, (2) it allows prioritization between messages when transmitting simultaneously and (3) allows nodes to filter out messages not intended for them. A message with a lower ID value will be prioritized over a message with a higher ID value through an arbitration process that is built into the protocol.

3.2.2 CAN Kingdom

CAN Kingdom is a set of protocol primitives (sometimes called a meta protocol) that enables designing a higher-level protocol for setting up a distributed control system using CAN [9]. The protocol separates the roles of the system designer and node designer. Node configuration is built into the protocol which enables the separation. This gives more freedom at the system level and promotes more general, system-independent, nodes. At least two operation modes are required to facilitate the node configuration: a start-up mode and one or more run-time modes.

The protocol places heavy emphasis on similes to aid in system comprehension.

3.2.2.1 Protocol structure

The protocol divides the nodes in a DECS into sensor or actuator nodes, called Cities, and a master node, called King. These together with the CAN bus, called Postal System, make up a Kingdom. Every City has a Mayor that specifies what data it needs and what data it can transmit. The King is responsible for defining what information each City should send and when, as well as what information it should listen for so that every City can fulfill its function in the system.

The information on the bus is in the form of CAN messages, referred to as Letters, with a CAN identifier (Envelope) and a payload (Page). A Page consists of at most eight Lines with each Line being represented by one byte in the data payload. For a receiver to correctly interpret a Page it needs to be decoded. The tool for this is called a Form which says what the data on each Line represents.

When transmitting more than eight Lines multiple Letters are needed since each Letter can only carry one Page. These Letters have the same Envelope containing

a Page where one or more Lines (or at least some bits of a Line) are dedicated as indexation. To interpret an Envelope with multiple Pages a set of Forms is needed (one per Page) which constitutes a Document.

To receive and interpret a Letter, its Envelope needs to be matched with a Document. This matching is done through a structure called Folder. Any Document for interpreting received Pages needs to be placed in a Folder. An Envelope is then assigned to the Folder. Now when a Letter is received it gets matched with a Document based on its Envelope and can be correctly understood. Similarly, when transmitting a Letter a Document is used to write the Pages before sending it out on the Postal System with the assigned Envelope.

3.2.3 RT-Kernel

RT-Kernel is an RTOS developed by the company RT-labs [10]. The OS provides timers, task management, and basic synchronization primitives such as mutexes and semaphores. In addition to its real-time capabilities, RT-Kernel also supports several communication protocols, file systems, and I/Os.

Tasks in RT-Kernel, as in most OSes, are in one of four states: ready, running, waiting, or dead. Starting a task sets its state to ready and places it in a queue of all tasks in the ready state, according to priority. The task with the highest priority in the ready queue enters into the running state and starts executing. A task is put in the waiting state if it is waiting for a resource or if it is stopped and needs to wait for a restart. When a task finishes it enters into the dead state. Tasks in the dead state will signal a reaper task that will free the memory.

There is a system in RT-Kernel for implementing periodic tasks called time-triggered tasks. Time-triggered tasks need static schedules that are defined at compile time. The schedules can be changed during run-time but they cannot be reconfigured which is a requirement for use with the CAN Kingdom protocol. Reconfigurable periodical tasks can instead be realized by using timer callback functions to start a task at a fixed interval.

4
Methods

4.1 Scheduling of communication on CAN bus

To implement a performant controller it is important that the delays introduced by the distribution of (airspeed)measurements on the bus is kept low, and to ensure robust performance it is important that the delay is somewhat consistent. To achieve both these goals, high bandwidth and low delay variance, scheduling of the messages on the CAN bus will be used [11].

To obtain maximal performance of the physical control system it is desirable to introduce as short of a delay as possible from the distribution of the measurements, specifically the airspeed measurement. Given the limitation of the CAN bus operating at 125 kbps, with the communication already, a upper boundary for the update of the airspeed measurement both in terms of mean delay and maximal delay exists.

For the airplane the maximal bitrate of the CAN bus is 125 kbps where some of the bandwidth is already taken up by other already implemented and necessary communication:

ID	Description	Data type	Period [ms]
10	Yaw stick	uint16	14
11	Throttle stick	uint16	14
12	Pitch stick	uint16	14
13	Roll stick	uint16	14
14	Trim switches	uint16[4]	200
15	Flaps slider	uint16	200
17	Auxiliary switches	uint16[4]	200
60	Roll and pitch angle IMU estimates	float32[2]	10
70	Airspeed measurement	float32	10

Table 4.1: Table of messages on CAN bus

Because tasks in RT-Kernel can only be scheduled in 1 ms intervals, it makes sense to attempt a schedule where each time slot is 1 ms even though the actual transmission time on the CAN bus for the messages relevant is less then 1 ms resulting in less then 100% utilisation. A schedule based on the table 4.1 is proposed, where conflicts of time demanded are solved by moving the message time slot such that the message arrives at the intended time or before in table A.1. The airspeed message is scheduled first, and will always be transmitted with exactly 10 ms between messages.

4.2 Nodes in the system

The distributed control system for the UAV already contains a multitude of nodes needed for flight. For this thesis, the scheduling of all transmissions will be changed, and an airspeed node will be added, responsible for make the airspeed measurements available for the other nodes. At both end of the bus a termination of 120 Ω is used. A drawing of the nodes can be seen in the appendix A.1.

4.3 Logging of messages on CAN bus

As all internode communication is done via one CAN bus the messages will be recorded for fault detection and performance verification. In the proposed testing environment of the complete system, flying, it would be beneficial if some lightweight portable device could be installed in the aircraft to record the time and content for all messages on the bus. A Kvaser Memorator Pro[12] is used. After a test flight, the data will be transformed using Kvaser Database Editor[13] to a suitable format for subsequent analysis in Matlab.

4.4 Airspeed node

To control the airspeed of the aircraft some sense of the airspeed is needed. A simple approach is to measure the air pressure at a stagnation point and compare that pressure to the static pressure [14]. The difference is the dynamical pressure, from which the airspeed can be calculated.

$$p_{stagnation} - p_{static} = \frac{1}{2}\rho v^2 \tag{4.1}$$

Where ρ denotes the air density and v is the airspeed.
Airspeed can be obtained;

$$v = \sqrt{\frac{2p}{\rho}} \tag{4.2}$$

While the pressure will be measured the air density remains unknown. Air density varies over time, depending on weather and wind. A simple fixed value will be used, 1.293 kgm^{-3} [15].
To measure the air pressure a differential pressure sensor MS4525DO[16] will be used. Since the sensor measures the difference between two inputs the dynamic pressure can be measured directly by appropriate mounting and connection of a pitot tube. The inlet at the front of the pitot tube will be a stagnation point and the inlet at the side of the pitot tube will experience the static pressure since the air passing by is considered to have roughly the same velocity and pressure as the incoming air.
The sensor, MS4525DO, communicates with the CAN node via I2C according to the specifications from TE[17]. A measurement fetch consists of the master, in this case the CAN node microcontroller, requesting data from the sensor and the sensor

returning two data bytes. The order of most and least significant bytes between the I2C output from the airspeed sensor and the hardware/software stack of the CAN node is not the same, hence bit shifting has to be done. The 2 bytes from the pressure sensor in each update contains two most significant status bits followed by 14 least significant data bits. A pressure difference between the sensor inports of 0 Pa gives an integer value of 8192, with a non-zero pressure difference gives a positive or negative offset from 8192 with the magnitude of the offset linearly corresponding to the measured pressure. Since the measured dynamic pressure is assumed to always be bigger than the static pressure only non-negative pressure readings are expected. After removing any negative pressure measurements the equation 4.2 is applied and a float representation of the airspeed is obtained. The data type float is used to accomodate an arbitrary choice of the constant of ρ air density, keeping the data type inline with what is already used in the system such as IMU measurements [4.1], as well as that a decimal representation of airspeed is needed for control transfer functions.

Interfacing with the pressure sensor over I2C is done within the RT-Kernel framework. For the airspeed node's main function, to make an airspeed measurement available on the bus, two main tasks are scheduled in the real-time operating system, RT-Kernel. One task for reading the pressure from the sensor and calculating an estimate for the instantaneous airspeed, and one task for sending the airspeed message on the CAN bus. To follow the designed communication schedule a combination of period and offset is used in RT-Kernel.

After an estimate of the current airspeed has been calculated it will be broadcasted on the CAN bus, as a message with id:70, containing one float(four bytes), being transmitted at 100 Hz.

4.5 Node hardware and pressure sensor

The airspeed node consists of one microcontroller, STM32F302, mounted on a pcb with IO ports and needed periphery integrated circuits. The microcontroller is connected by twisted wires to the other microcontrollers on the CAN bus. Attached to one of the IO ports on the microcontroller is the pressure sensor, which communicates over the I2C protocol. The differential pressure ports on the pressure sensor are connected with silicon tubes to the pitot tube to measure the pressure of the incoming air.

The other nodes on the CAN bus have the same microcontroller, but with different IO ports.

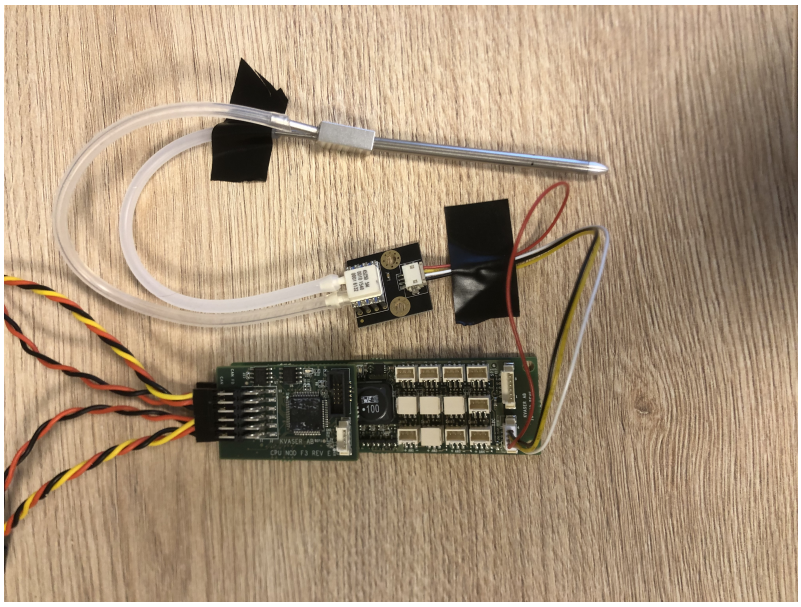

Figure 4.1: Node hardware pictured with pressure sensor and pitot tube

4.6 Controller approach

It can be reasoned that two main controllable inputs effect the airspeed, throttle and elevator. Increasing the throttle will increase the forward trust, thus increasing speed until a new equilibrium between thrust and drag has been reached. Tilting the plane downward will convert some of the potential energy into kinetic energy, hence increasing speed. Correspondingly, decreasing throttle or tilting the plane upwards will decrease speed. Therefore developing a control strategy which uses both throttle and elevator to control airspeed is desired. Throttle actuation should be used to correct for steady state errors in airspeed, while elevator actuation may only be used for high frequency corrections as a constant offset on elevator would result in undesired climbing/descending. A mixed sensitivity H_∞ approach will be taken. It should be noted that the controller will only strive to keep the airspeed as close to the demanded value as possible, and no other pilot objectives will be considered.

4.7 Modelling of pitch dynamics

A simple modelling of pitch dynamics is proposed where the force from the horizontal stabiliser is linear with elevator angle. The plane has a self righting behaviour(pitch angle will go to zero without control input) which will be initially modelled as spring force pulling the plane to level. It can also be reasoned that drag on the aircraft body and stabiliser will oppose a rotational speed, acting as a damper. In such a way a simple model is proposed.

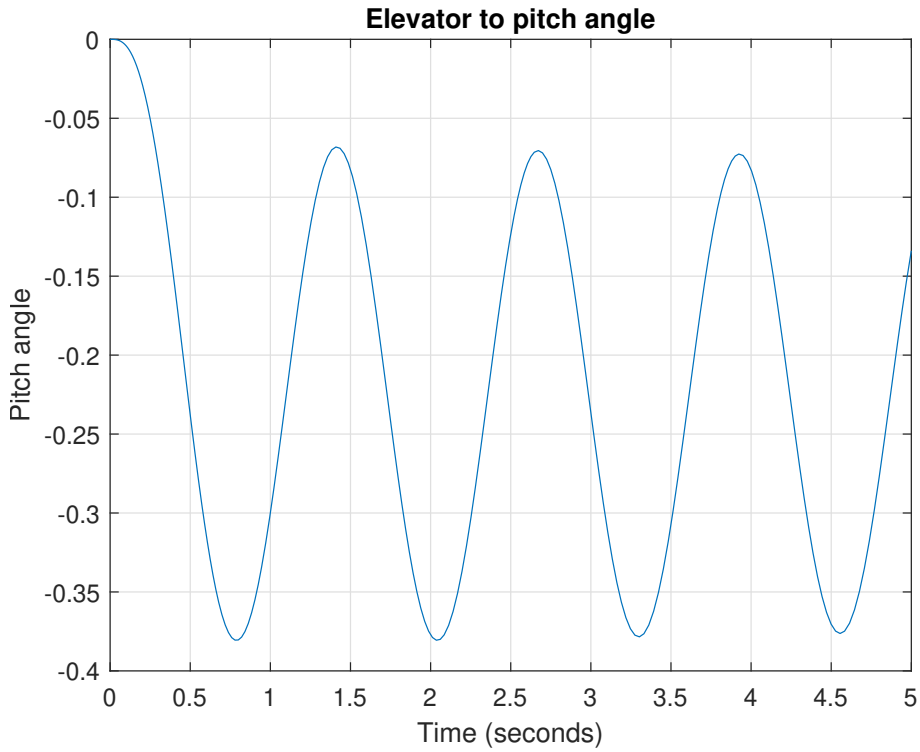

Figure 4.2: -0.1 rad elevator step without angle of attack damping

A stronger damping force then from the drag on the aircraft is also given from the fact that angle of attack on the stabiliser increases by the upwards movement part of a pitch oscillation and decreased in the downwards movement part of a pitch oscillation:

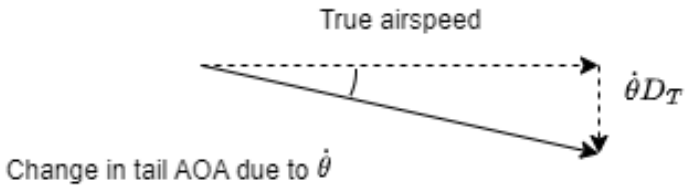

Figure 4.3: Angle of attack damping of pitch rate

The same step in elevator angle to pitch angle with added angle of attack induced damping:

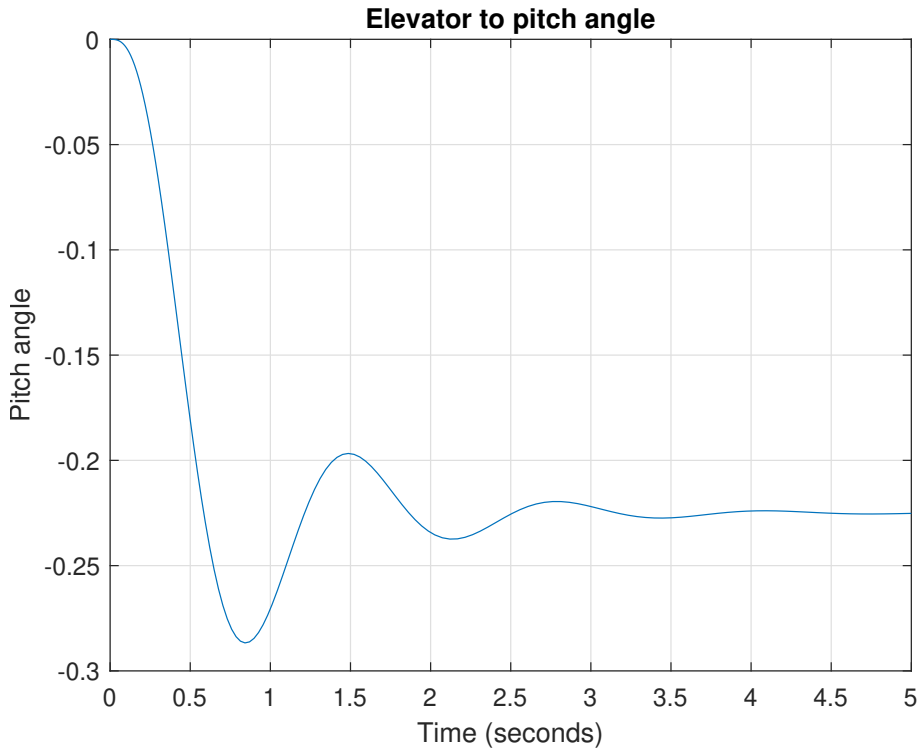

Figure 4.4: -0.1 rad elevator step with angle of attack damping

The forces proposed to be acting on the tail can be summarised in a figure:

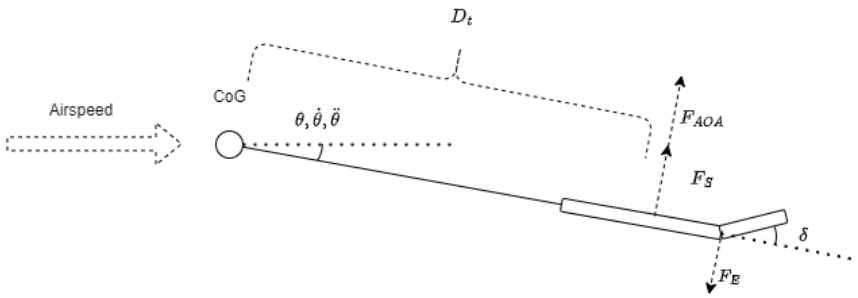

Figure 4.5: Forces acting on tail and elevator

4.8 Linearisation of pitch dynamics

As the model contains several non-linearities, such as quadratic relationship between velocity and drag, a sinusoid term for glide induced acceleration and tangens for aerodynamic pitch damping, care must be taken to linearize the model in a suitable state. As can be seen, using Simulinks builtin linearize tool, the behaviour is different for around different times as the simulation elapses. The exact amplitude of the step response might not be critical, but the sign directly impacts control synthesis. A positive step on the elevator should result in the plane pitching up and a reduction in speed. The linearisation obtained around the state at T=0 is used.

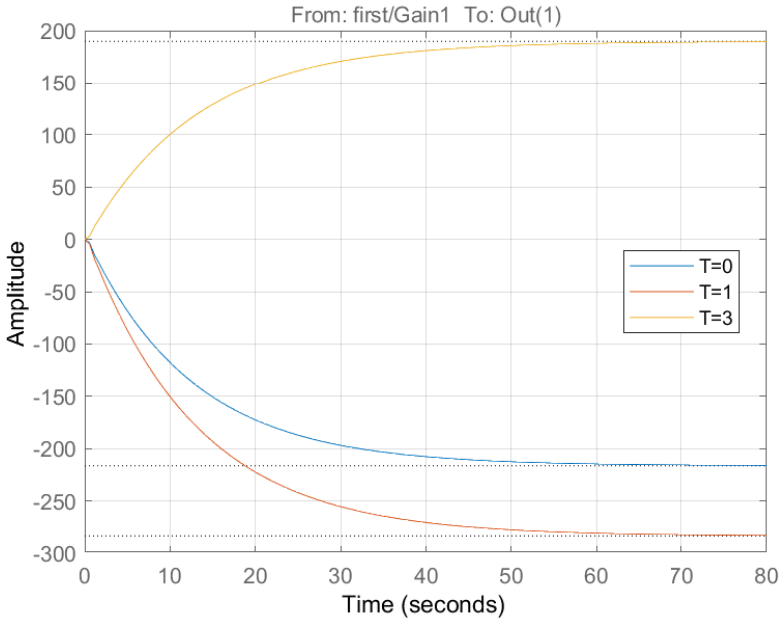

Figure 4.6: Step response on linearised transfer functions from elevator angle to speed at different times

4.9 Modelling of thrust dynamics

Two brushless direct current motors for propulsion are mounted on the airplane, one on each wing. For the purpose of controlling the speed of the airplane these two motors will be considered to combined produce one purely propulsive force, instead of treating all phenomena relevant such as lift, drag, torque and rotational velocity. Each motor is driven by an inverter which is controlled by microprocessor connected to the CAN bus. The communication between the CAN node and the inverter is a PWM signal, where the pulse duration corresponds to a desired rotational velocity ω of the motor. Given factors such as the electrical properties of the motor, the moment of inertia of the motor and the propeller, and the aerodynamic drag on the propeller this desired rotational velocity to actual rotational velocity will have some relationship. From this actual rotational velocity a thrust will be produced, varying with the aerodynamic properties of the propeller, velocity of the incoming air, air density and other factors. Combing these mentioned physical properties to a model describing the relation from input PWM to the inverter to obtained thrust requires several steps of approximation and characterisation of equipment. A simplified model consisting of a first order transfer function between desired thrust and actual thrust is instead proposed for initial controller synthesis:

$$\frac{F_{actual}}{F_{demand}} = \frac{1}{\tau_{Motor} s + 1}$$

Where τ_{Motor} denotes the time constant of the motor-inverter combination.
To capture that the produced trust has some limit, influenced by factors such as

airspeed, propeller properties, motor properties, this transfer function is be followed by a saturation block.

4.10 Modelling of speed behavior

Given a model of the relationship between elevator angle and pitch angle, and a model of the relationship between desired thrust and actual thrust a model of the velocity of the aircraft can be assembled. Starting with the relation between thrust and speed it is noted that the aircraft will accelerate at a rate of the quotient of (sum) applied thrust and mass.

$$\dot{v}_{thrust} = \frac{F_{thrust}}{m} \tag{4.3}$$

This acceleration can be integrated, knowing the initial airspeed, to obtain the airspeed.
For the relation between pitch angle and airspeed it can be noted that in a simplistic manner the airplane will accelerate along the glide slope just like a trolley would accelerate along a mountainside. Here it can be reasoned that the glide slope will directly correspond to the pitch angle, assuming the angle to attack stays constant.

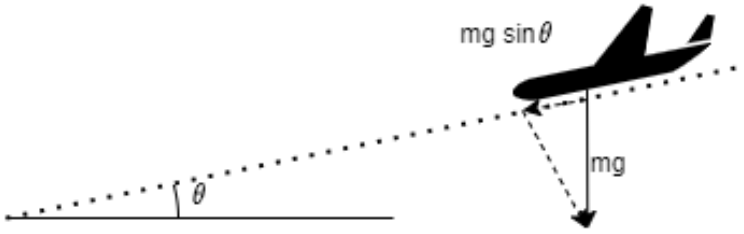

Figure 4.7: Relationship between glideslope and forward acceleration

Then the pitch angle can be considered to influence the airspeed as in 4.4

$$\dot{v}_{pitch} = gsin\theta \tag{4.4}$$

The acceleration component from the glide slope can also be integrated to obtain airspeed. As such a model with two inputs, elevator angle and demanded thrust, and one output, airspeed is obtained.

$$\dot{v} = \dot{v}_{thrust} + \dot{v}_{pitch} = \frac{F_{thrust}}{m} + gsin\theta \tag{4.5}$$

For verification of reasonable performance of the airspeed controller it will also be useful to calculate the accumulated altitude changes from glideslope/pitch angle and airspeed as large or unreasonable/impractical changes in altitude are unacceptable.

4.11 Implementation of numerical model

The differential equations proposed are to be implemented in Simulink for numerical analysis. As a linear model is needed for the controller synthesis, linearsation will also be performed for a suitable operating point. After the model has been linearised and two transfer functions has been obtained, from elevator actuation to airspeed and from motor actuation to airspeed, a controller will be synthesised in Matlab.

4.12 Controller design

To achieve the goal of controlling airspeed using two actuators, elevator and throttle, a systematic control synthesis of a mixed sensitivity control, with the objective of minimising a cost function. This cost function will have inputs of error signals, actuator signals and optionally state signals. The weights of the components in the cost function are frequency dependent, and will be represented with continuous transfer functions. In the implementation in this report no cost will be applied to the states, as airspeed error is assumed to be of superior importance. The weight on airspeed error has dimension $\boldsymbol{R}^1$ id est containing one transfer function. The weight on the actuation signals has dimension $\boldsymbol{R}^2$ id est containing 4 transfer functions. The transfer function in location (1,1) is the weight for the elevator actuation and the transfer function in location (2,2) is the weight for the motor actuation. On the off diagonal entries in the actuation weight the transfer functions are 0.

To capture the desired frequency dependent behaviour of actuation and error signals the weights will be manually tuned. Achieving a low settling time for a step on airspeed reference. Using the elevator to compensate for fast changes in airspeed, and the throttle for slow changes in airspeed. The actuation signals for likely errors in airspeed should not saturate the physical limits of the UAV or in the case of elevator cause excessive altitude loss. To achieve these control objectives the magnitude, and cut-off frequencies of the controller weights will be tuned.

The matlab command mixsyn[18] is used to synthesise the controller transfer functions given the model of the plant dynamics and the cost function.

To visualise how the feedback control loop with performance weight looks the following illustration can used:

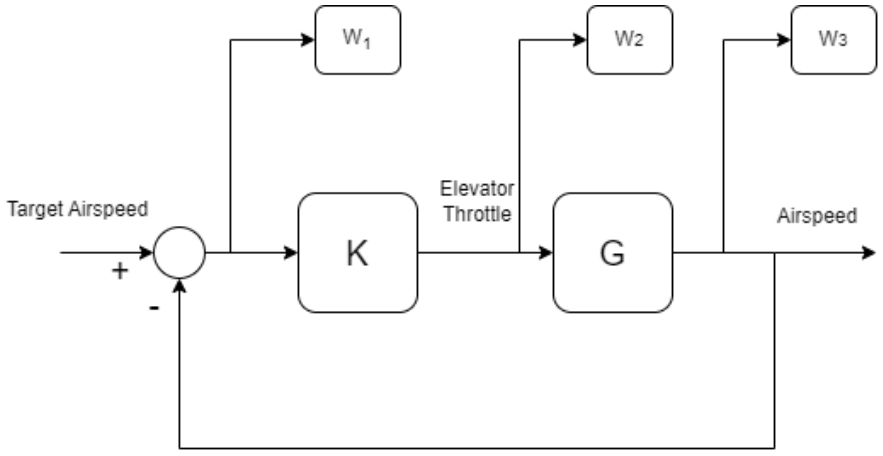

Figure 4.8: Controller-Plant overview

Zooming in on the multiple input plant and multiple output controller, it is possible to decompose both transfer functions:

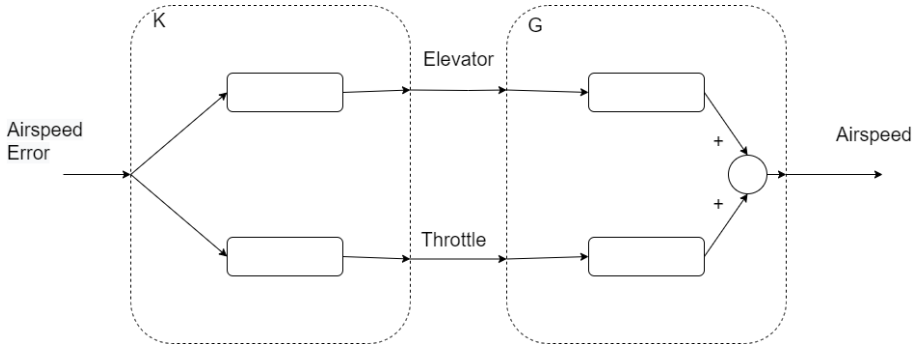

Figure 4.9: Exploded view Controller-Plant

with each small block representing a transfer function of dimension 1.

4.13 Controller discretisation

When a controller is discretised the closed-loop behaviour will no longer be the same as in the continuous case, but discretisation is necessary to implement the transfer functions in the microcontrollers. How this discretisation effects the closed loop behaviour can be assumed to be somewhat equal to introducing a delay in the continuous closed loop system of half the discretisation timestep[19].

Using a discretization timestep of 10 ms, resulting in a delay of around 5 ms. The induced delay is assumed to have a negligible effect on the dynamics of the complete feedback system since a delay of 5 ms is small in comparison to the time constants in elevator servo actuator, motor, controller weights and the resulting controller transfer functions.

To validate that the added delay of 5 ms is negligible, a delay is added to the continuous simulation model and controller performance is compared to the case without an added delay of 5 ms.

For discretising the two control transfer functions with a timestep of 10 ms the the Matlab command c2d[20] is used.

4.14 Controller implementation

From the discretisation the two transfer functions from airspeed error to elevator angle and motor thrust, respectively, are obtained. These two transfer functions need to be implemented in the nodes in the distributed system. The transfer function from airspeed error to elevator angle will be implemented in the elevator node and the transfer function from airspeed error to motor thrust should be implemented in the motor node.

A transfer function of N-th order will require the last N inputs to and the last N outputs from the transfer function to calculate the next output[21]. Hence, the last eight airspeed measurements and the last eight outputs, elevator angle or motor thrust, need to be saved. An array containing the inputs to the transfer function is maintained in the c-function receiving the airspeed messages in the elevator and motor node, respectively. An array containing the previous outputs from the transfer function, elevator angle or motor thrust, is maintained in the c-function where the propagation of the transfer function is executed. Apart from the arrays containing previous inputs and outputs the coefficients for each input and output is also needed. These coefficients define the transfer function and are obtained in the controller discretisation step. The controller coefficients are saved in two arrays, one array for input term coefficients and one array for output term coefficients. A weighted sum of the input and output terms, with the weights defined in the coefficient arrays, is calculated and the result is the output from the transfer function. The output from the transfer function is saved in the array of outputs and also used to actuate the elevator or motor.

The RT-Kernel task of receiving the airspeed CAN message and subsequently triggering the transfer function task is based on that the airspeed messages are received exactly each 10 ms, since 10 ms is the time step used in the discretisation of the transfer functions.

4.15 Activating controller

As the proposed airspeed controller will have complete authority over the inverter for the motor, and hence the movement for the propellers, a dangerous situation can arise if one of the motor-propeller units were to be powered when a person or object is close. While practising general caution such as avoiding the vicinity of the propellers when the aircraft is connected to power reduces the risk of injury, some safeguards are proposed.

In the configuration of the aircraft in the beginning of the project the signal to the inverters directly corresponds to the position of the throttle stick on the transmitter. Following a test protocol where the aircraft performs takeoff and initial flying manoeuvres in full manual control by an operator, it stands to reason that the motors

should only be allowed to be spun up by the automatic airspeed controller if the motors are already spinning in the manual flight mode. Hence, the throttle stick should be over the halfway position.

By similar reasoning, as the aircraft should be flying before the automatic speed control can be activated, the airspeed should be sensed above what is needed for the plane to maintain flight and especially above what is measured on the ground which can be in the magnitude of 1-3 ms^{-1}. A value of 10 ms^{-1} is proposed as a starting point.

Finally, a switch on the operator remote control is used to activate the automatic airspeed controller.

These three conditions, throttle stick position, sufficient airspeed and switch position all have to be fulfilled to activate the automatic airspeed controller. If any of the conditions then were to become false, the operator lowers the throttle stick too much, the airspeed falls below the threshold or the activation switch is lowered, the automation airspeed controller should immediately be disengaged.

To provide a structured description of the specification for activating the automatic airspeed controlled a EFA[22] is proposed. The events would be activate/ deactivate automatic airspeed control. The guard for activating the automatic airspeed control can be seen in figure 4.10. The action is to change from manual to automatic airspeed control.

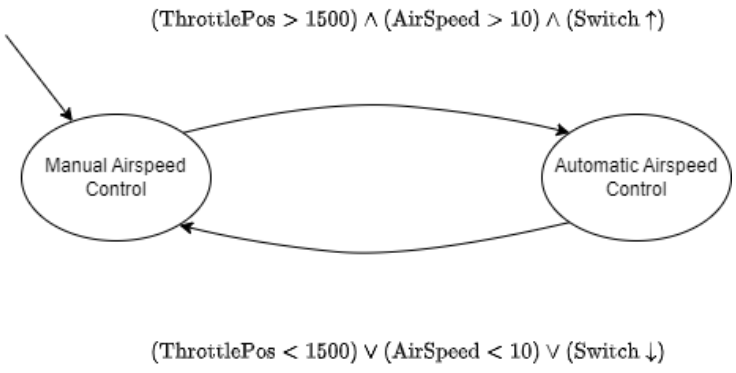

Figure 4.10: EFA for controller activation

4.16 Flight testing

To verify the function of the complete system several layers of verification needs to be done. Is the UAV possible to fly manually by an operator. Are the propositions done in the modelling of the dynamics of the UAV reasonable, and if so, are the proposed modelling constants of appropriate numerical values. Is it possible to activate the automatic airspeed controller, and if so, does the controller respond to changes in reference airspeed as in the simulations. Does the controller handle disturbances such as turbulence and change in attitude well. The result from the test flight will be presented in the results section of the report.

As a first step in flight testing the UAV is flown manually where it can be noted that the pitch and roll movement of the UAV induced by turbulence in the air make characterisation of modelling constants impractical. It is at least subjectively noted that the pitch self righting property is less significant then initial assumed, so the model is updated accordingly.

5

Results

For the readers convenience discussion will be interwoven with the results, as having the plots and diagrams close to the reflections aims to give a more intuitive understanding.

5.1 Block diagram Simulink

The model derived in simulink to describe the effect of elevator command and thrust command to airspeed is seen in figure 5.1.

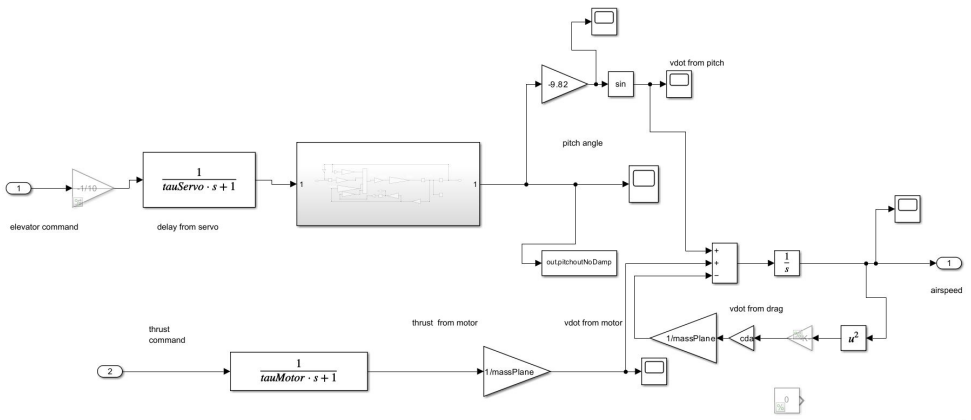

Figure 5.1: Block diagram Simulink

A detailed view of the dynamics from elevator angle to pitch angle can be seen in figure 5.2.

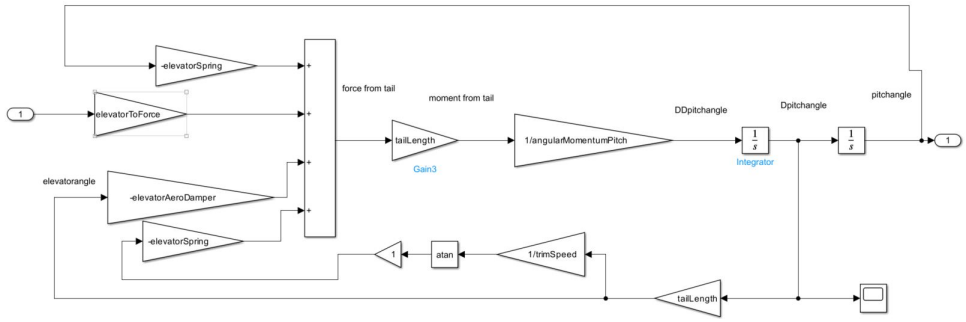

Figure 5.2: Detailed view, sub block pitch dynamics

And finally the complete feedback system with controller and the nonlinear process model in figure 5.3.

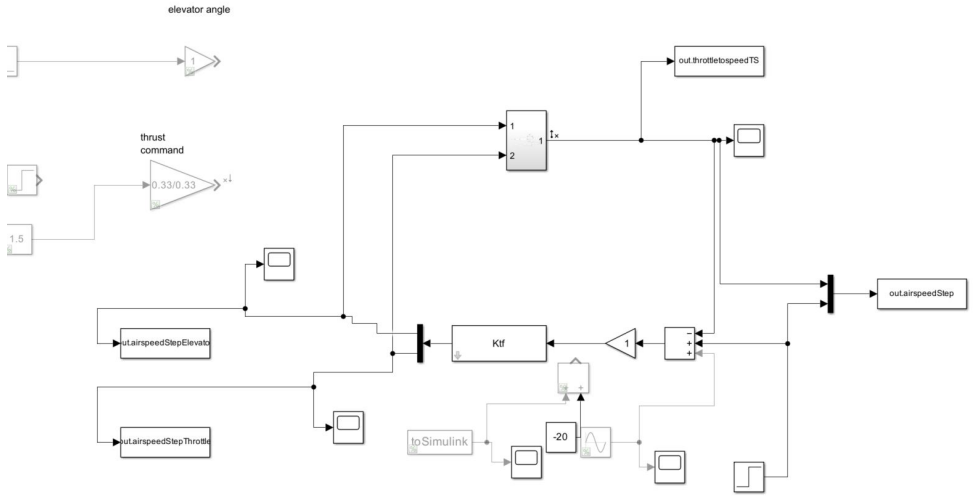

Figure 5.3: Simulink complete feedback with controller

5.2 Controller synthesis weights

A bode diagram is used to visualise the weights used for the controller synthesis. The weights are adjusted manually to achieve the control objective.

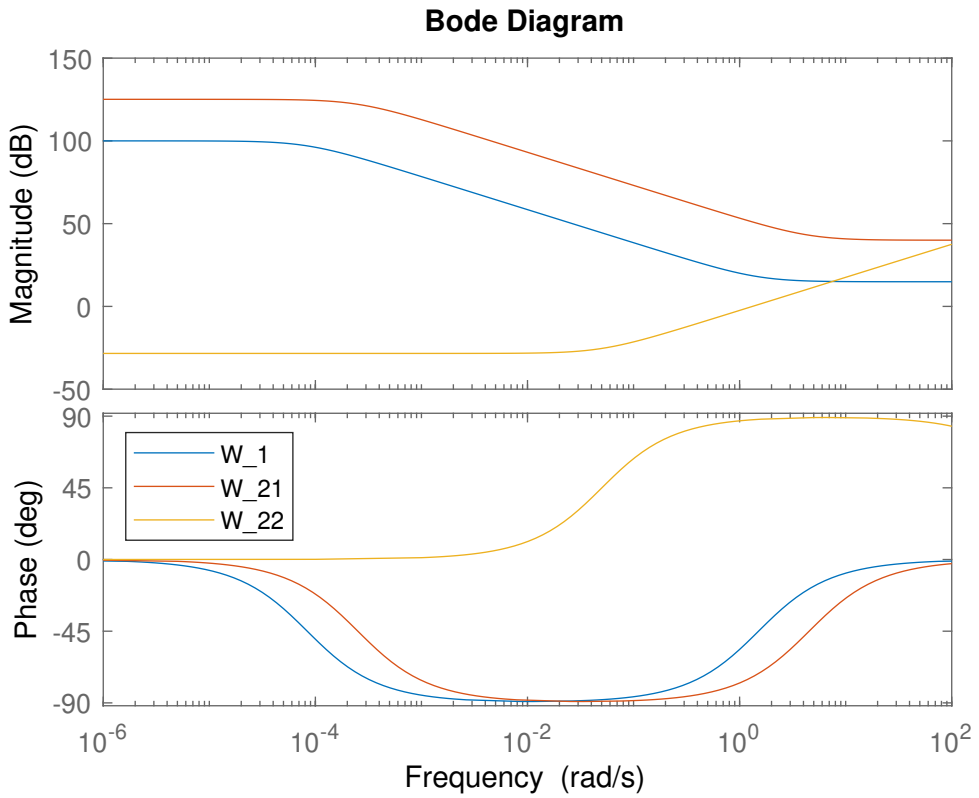

Figure 5.4: Bode plot of weights

Weight 1, the gain for the airspeed error in the controller synthesis cost function, is of low pass character, meaning that high frequency measurement noise does not noticeably effect control actuation. Weight 21, the gain for elevator actuation is of low pass character meaning that the elevator will be heavily punished for steady state actuation. Weight 22, the gain for motor actuation, is of high pass character meaning that high frequency use of the motor will be avoided and low frequency use will be allowed (to ensure there is no steady state offset in airspeed error).

5.3 Subresult from simulation model

The response of the airspeed after applying force of 5 N can be seen in figure 5.5.

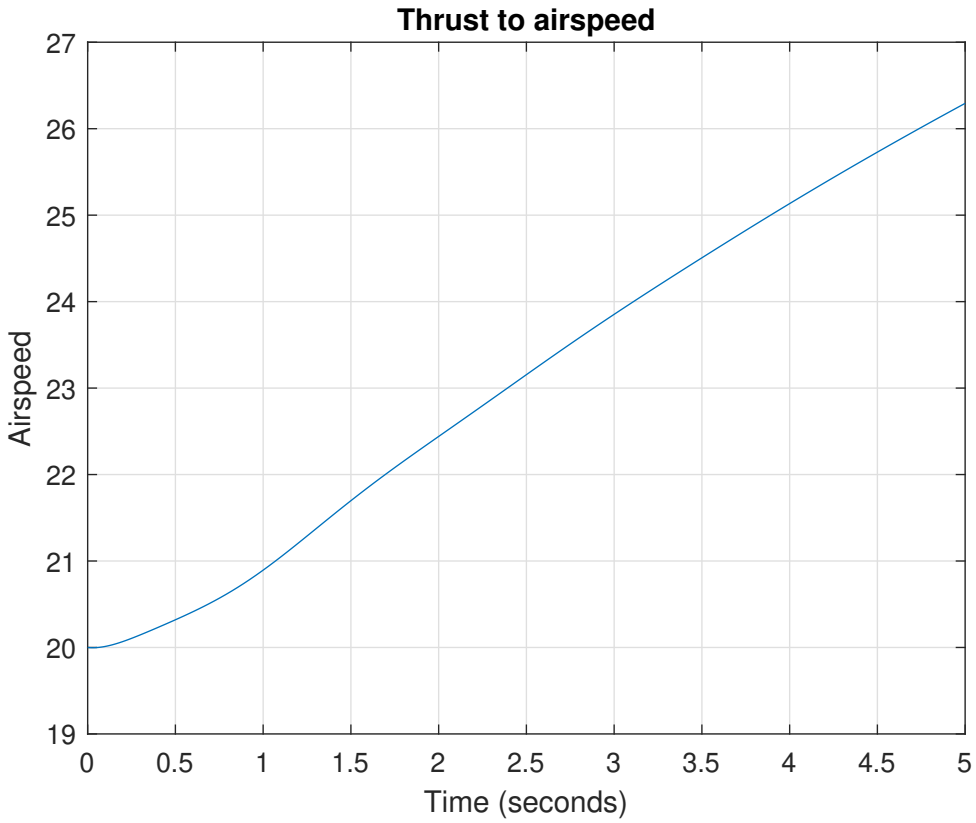

Figure 5.5: Step of 5 N, to airspeed

5.4 Controller verification

A step on airspeed reference of 5 ms^{-1}, from 20 ms^{-1} to 25 ms^{-1}, is applied to evaluate controller tracking of reference signal and the actuation signals used.

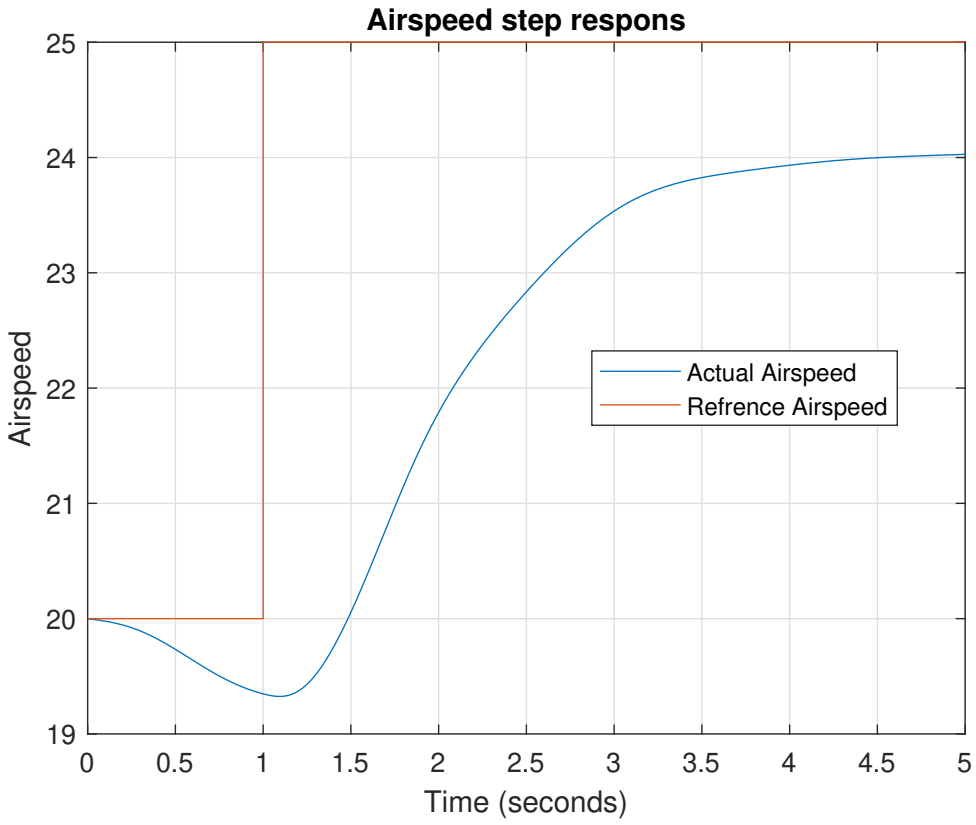

Figure 5.6: Reference step on airspeed

It can be seen that the actual airspeed follows the reference airspeed, with a settling time of around 2.5 seconds. The dip in airspeed between 0 and 1 seconds is proposed to be a result of that the 'integrating'-action of the motor transfer function is not yet equalised against the thrust needed to overcome the drag force at 20 ms^{-1}, since the simulation starts at time 0. The same lack of time to 'integrate' is proposed to cause that the airspeed does not converge to 25 ms^{-1} until several ($\sim$ 20) seconds later.

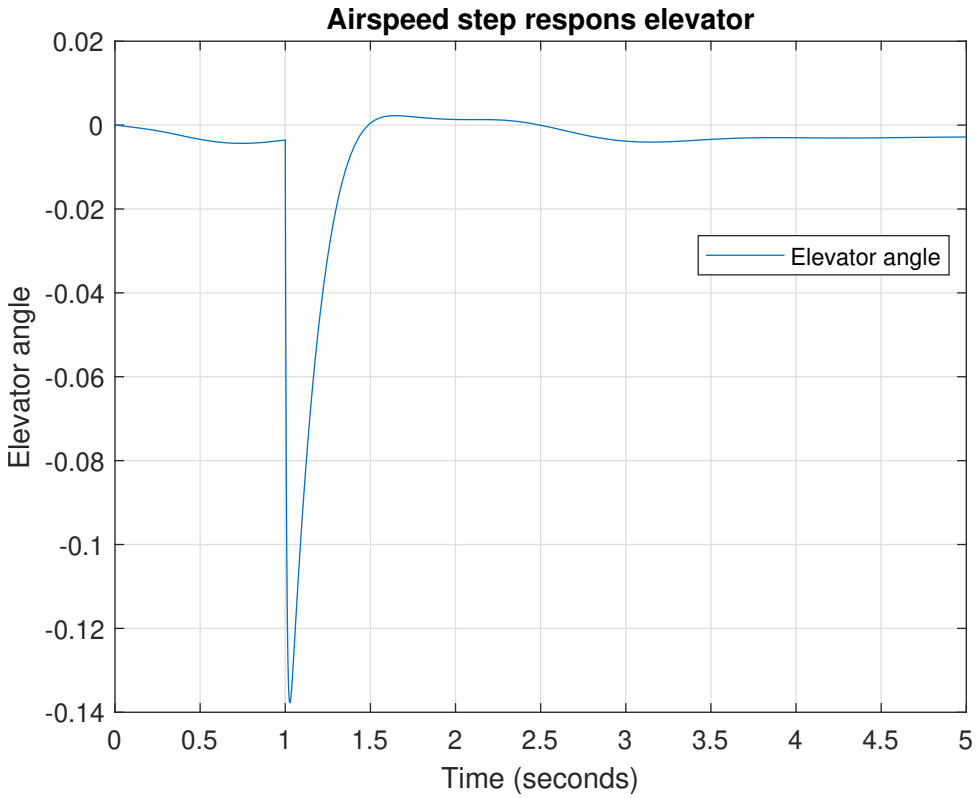

Figure 5.7: Elevator angle response to reference step on airspeed

The elevator angle response to the step in reference airspeed is reasonable and as desired, a fast decline in angle to pitch the plane downward and initiate an acceleration, followed by the return to around zero, with seeming low activity at steady state error. A peak elevator angle of -0.14 rad is reasonable.

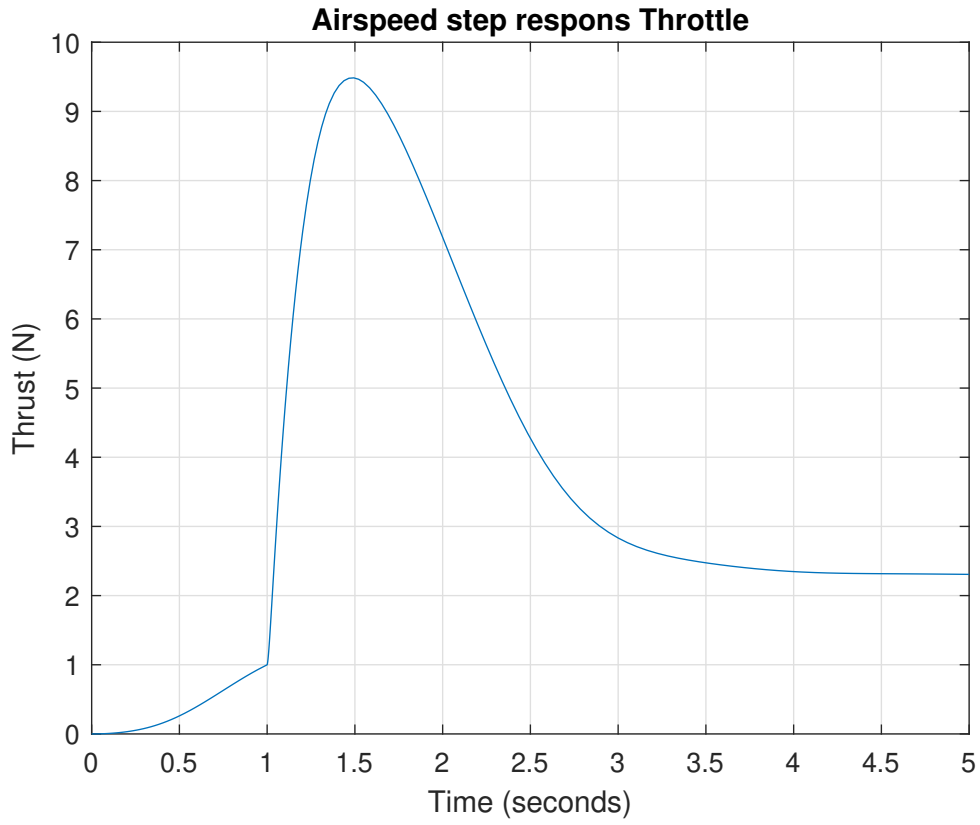

Figure 5.8: Motor thrust response to reference step on airspeed

The thrust response to the change is also as expected, a large force to initiate an acceleration followed by a steady state thrust to overcome the drag force at the new airspeed. A force of 10 N is also in the realm of reasonable performance from the UAV drive train as the max thrust at no airspeed is 40 N.

5.5 Controller implementation

To verify that the synthesised control law, expressed as two transfer functions, behaves the same way implemented in the distributed system as when developed in Matlab/ Simulink some tests are conducted. The airspeed measurement broadcasted on the CAN bus at 100 Hz will step from 0 ms^{-1} to 5 ms^{-1}. This ability to emulate a measurement sequence by injecting arbitrary messages on the CANbus and having the nodes react correspondingly is a useful tool for verification. The received airspeed measurement and calculated output from the elevator and motor nodes are transferred to a PC via serial communication and the log file is imported to matlab for the comparison.

The transfer function for airspeed error to elevator angle is of the 8th order, and once implemented in the node demonstrated unstable behaviour, giving unbounded output for a finite input. To reduce this observed instability an order reduction was performed with the matlab tool 'balred'[23]. A reduction to the 6th order was chosen as this reduction gave observably identical frequency, phase and step

response behaviour while making the transfer function coefficients several orders of magnitude closer to 10^0.

This response from the reduced transfer function from airspeed error to elevator angle is evaluated for a step from 0 ms^{-1} to 5 ms^{-1}. In the same plot the calculated step response from Matlab and the logged response from the implemented transfer function in the node is observed.

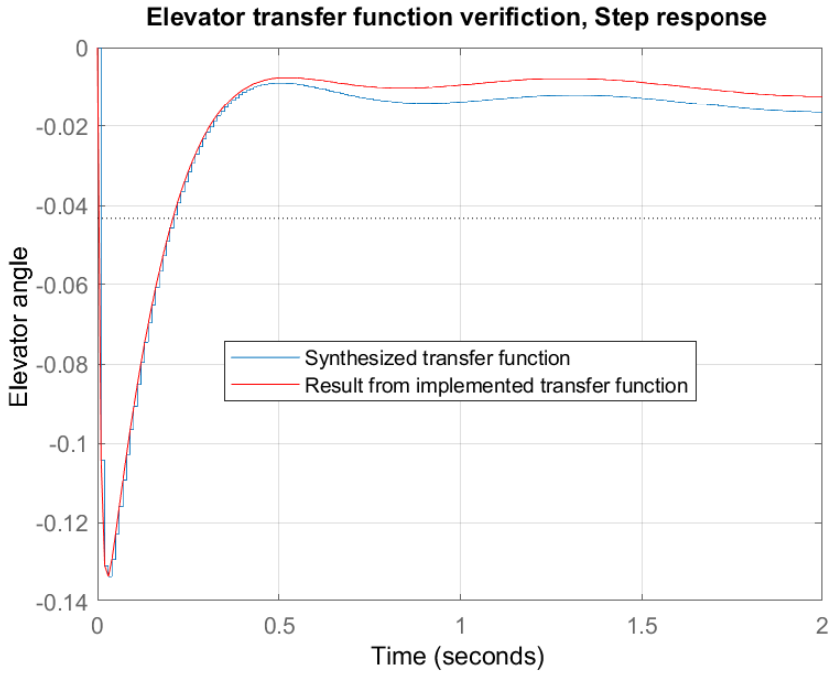

Figure 5.9: Evaluation of 6:th order reduced elevator transfer function

Overall similar behaviour can be observed, with some divergens towards the end of the sequence.

For the implementation of the transfer function from airspeed error to motor actuation a similar approach is taken by first order reducing from 8:th order to 6:th order. The frequency, phase and step response behaviour is similar.

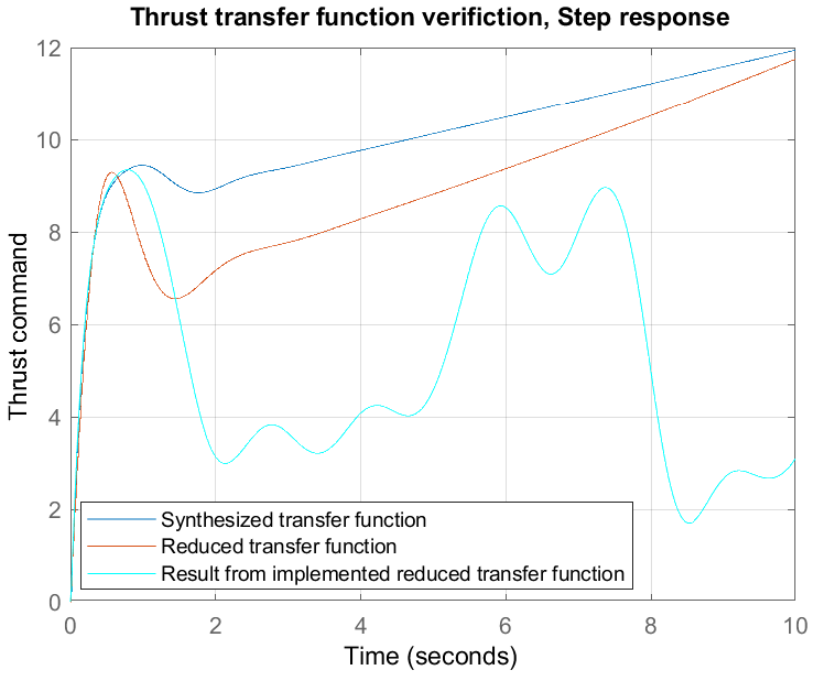

Figure 5.10: 6:th order motor transfer function divergence

As can be seen in figure 5.10, the implemented transfer function does not behave as expected with a large divergence from the calculated response at ∼1 s, corresponding to the change from increasing to decreasing output.

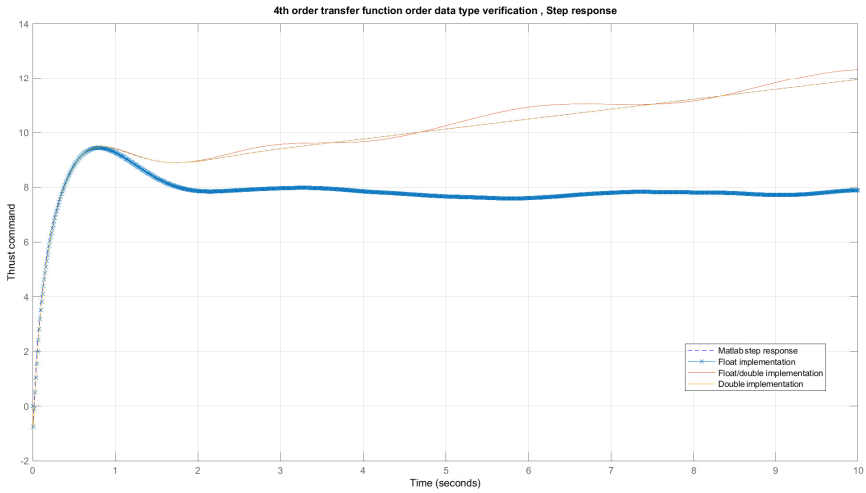

Figure 5.11: Comparison between float, float/double mixture and double implementation against simulated response for 4:th order motor control transfer function

In an attempt to understand what is causing this large divergence, the transfer function is further order reduced, to 4:th order and implemented as previously in the node, using the datatype float(4 bytes, in this context) for all coefficients and data as

previously. This implementation(figure 5.11) does not show desirable characteristics, the output has a different shape than simulated and no integrating action can be observed in the implemented response. To reduce these numerical errors the transfer filter coefficients are implemented as the data type double(8 bytes, in this context), and some improvement were observed.

Finally all variables used for the implementation of the transfer function are initialised as doubles and the resulting implemented response is inseparable from the calculated response. A further attempt is done using the data type long double, which does not increase performance as double and long double are the same in the used hardware/software environment, 8 bytes.

Moving forward all variables using in connection with the transfer functions will be initialised as doubles, and the result from the implemented full-order(8:th) transfer function for thrust response in the motor node:

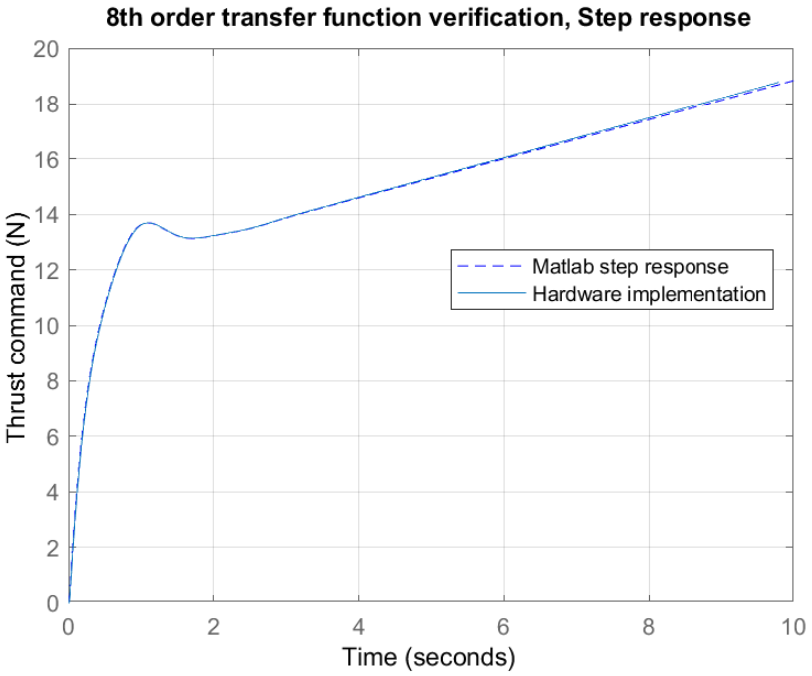

Figure 5.12: Motor transfer function 8:th order all doubles

and in the elevator node for elevator angle:

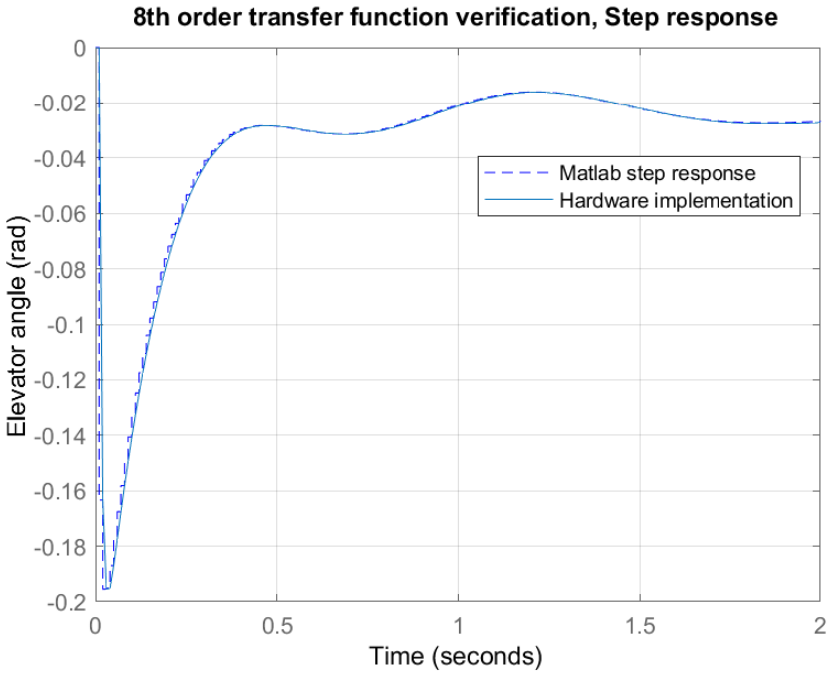

Figure 5.13: Elevator transfer function 8:th order all doubles

As can be seen in figure 5.12 and 5.13 the hardware implemented controller now behaves very similar in the distributed system as in the simulation environment.

It is reasoned that the significance of data type used, single or double, is a result of how the accuracy of the floating point number effects the transfer function iteration. Each transfer function update step consists of 16 multiplication and 16 additions of previous inputs and outputs. In each floating point multiplication and addition there will be a (small) numerical error. After iterating the transfer function for hundreds of iterations these inaccuracies will accumulate, which causes the instability of the transfer functions shown. Using double instead of single significantly reduces the numerical errors, and gives a stable transfer function iteration.

5.6 Flight performance

To verify that the controller behaves as expected a series of test flights are conducted. Several qualities are deemed to be of interest. For comparison with the simulated controller-plant response a step on airspeed reference from 20 ms^{-1} to 25 ms^{-1} should be evaluated. Endurance testing of the controller holding a set airspeed for a prolonged amount of time to ensure there is no apparent drift can be useful. Finally, a subjective evaluation of how the airspeed controller handles flying with sharp pilot manoeuvres will be conducted.

Since the controller consists of two parts, the motor and the elevator, it is possible to only activate one of the controller parts. For the first test flight only the motor node has a active controller, meaning that the elevator node is not used in the controller flight verification initially.

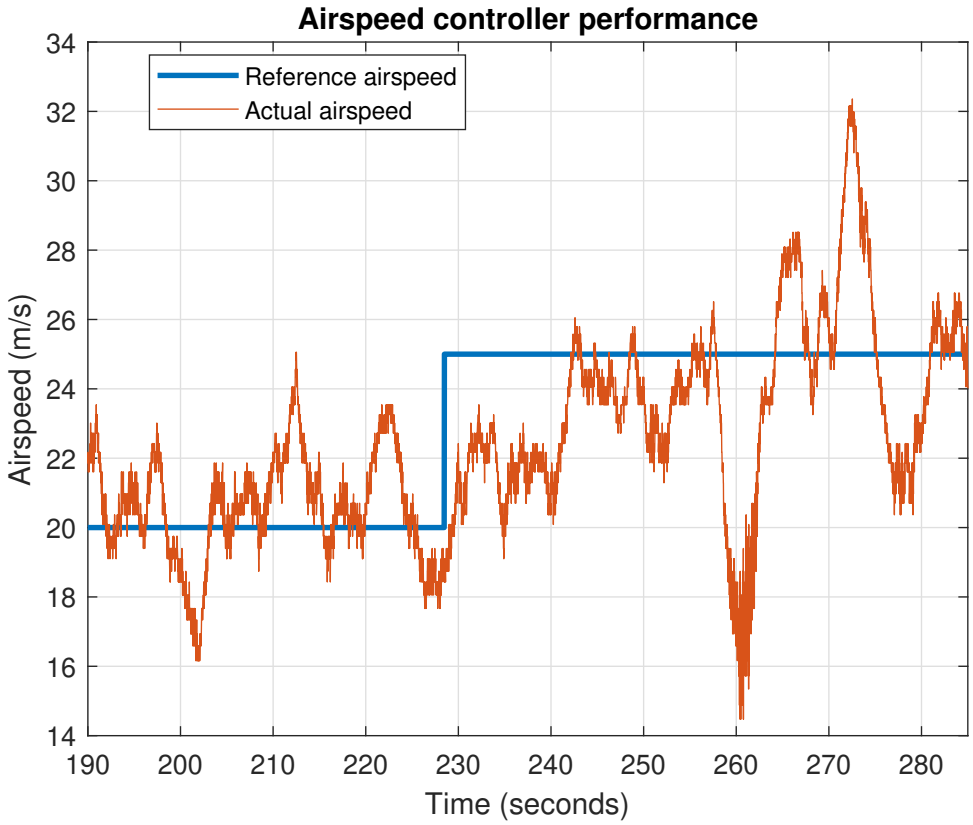

Figure 5.14: Airspeed controller performance with only motor controller activated

A few things can be observed in the airspeed plot from the flight. The controller does seem to provide reliable airspeed control around the desired speed, although the noise induced by turbulence and plane movement make any comparison of step response against the simulations impractical.

For the next test flight the controller is activate in both the motor and elevator node, meaning that the complete implemented controller is active.

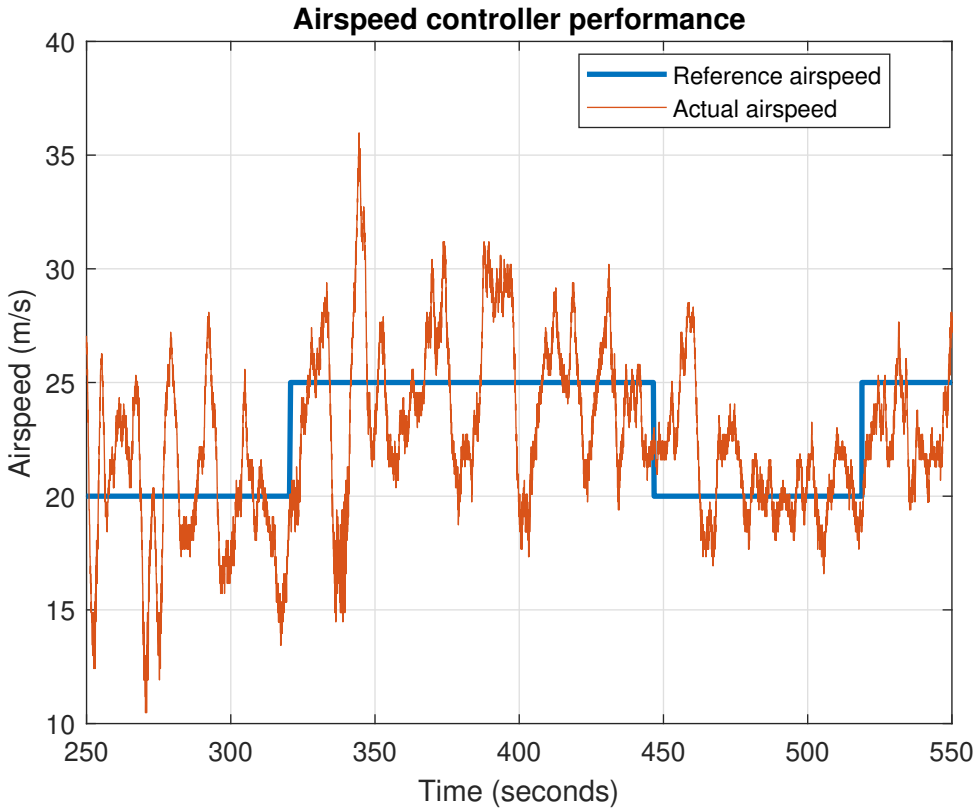

Figure 5.15: Airspeed full controller performance

Observably no apparent difference in airspeed control performance is obtained by activating the elevator controller. This lack of clear difference in performance does not mean that the elevator controller is useless, but rather that the noise dominates. Subjectively for the pilot the flight performance is worse with the elevator controller active, since it responds quite strongly in the frequency range of the turbulence induced airspeed variations, meaning that the pitch of the UAV becomes rather erratic.

Subjectively for the operator it is noted that the airspeed controller provides a quite pleasant flying experience. Sharp manoeuvres such as climbing or descending rapidly are dealt with effectively, with the thrust being increased/ decreased rapidly.

6
Conclusion

After the steps carried out as outlined in this report it can be concluded that it is possible to implement a mixed sensitivity controller in a distributed system. The challenges with this implementation includes scheduling the communication for low delay variance of important messages, modelling the dynamics of the system, prioritising control objectives, synthesising a controller, implementing the controller in the hardware in the distributed system and verifying performance, both on the bench as well as in the intended operating domain. Although performance of the controller scheme has been verified in hardware against the simulated response it is not feasible to fully compare flight performance to simulated performance due to external disturbances. It can be observed that the distributed mixed sensitivity controller does achieve a reliable airspeed control. In a testing environment with less turbulence it is likely that the elevator part of the controller will result in higher performance, like in the simulations. The developed airspeed control scheme can be integrated in a wider partial or fully autonomous flight environment.

The approach to distributing a multiple output mixed sensitivity controller taken in this thesis should be transferable to other control applications or industry domains.

7

Ethical and sustainability aspects

When discussing the implications of developing UAV flight control software one quickly realises the implications of UAVs as a whole must be considered. Previous applications include such as search and rescue, delivery of blood, medication or defilrators. Such an application demonstrates the aiding good health and well being. Of course, military applications have to be considered. While UAVs unquestionably has been used, and like will continue to be used, to hurt humans it should be noted that Kvaser does not currently develop UAV control systems for commercial sale, but a more likely application is in educational efforts, reducing potentially harmful long-term consequences beyond increasing the knowledge of modern control theory in new applications.

A more widespread use of distributed systems within UAVs could lead to efficiency improvements in some cases, either as less complex and less costly hardware/software or that more functionality can be added without changing existing hardware or software. Further benefits should also be noticeable since a distributed control system of the UAV means the development can be conducted while any functionality of other nodes can be simulated by injecting arbitrary messages on the bus. It follows that a more modular approach could enable a more responsible consumption. A continued development of distributed UAV control systems could aid in increasing capability to handle complex tasks, hence increasing societal gain from UAVs.

Bibliography

[1] Dronecode Foundation, "Px4 development," Available at https://px4.io/ (2022/11/25).

[2] M. Z. Babar, S. U. Ali, M. Z. Shah, R. Samar, A. I. Bhatti, and W. Afzal, "Robust control of uavs using h control paradigm," in *2013 IEEE 9th International Conference on Emerging Technologies (ICET)*, 2013, pp. 1–5.

[3] R. Farhadi, B. I. Kortunov, and A. Mohammadi, "Robust control design for the airspeed of uav," in *2015 IEEE International Conference Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD)*, 2015, pp. 145–148.

[4] X. Liangfei, H. Jianfeng, L. Xiangjun, L. Jianqiu, and O. Minggao, "Distributed control system based on can bus for fuel cell/battery hybrid vehicle," in *2009 IEEE International Symposium on Industrial Electronics*, 2009, pp. 183–188.

[5] S. Skogestad and I. Postlethwaite, *Multivariable feedback control, Analysis and design*, 2nd ed. Chichester: John Wiley and Sons, 2001.

[6] A. Olsson and R. Sjöberg, "Bare-metal implementation of a real-time distributed control system using can," Master's thesis, Chalmers University of Technology, 2022.

[7] R. B. GmbH, "CAN Specification Version 2.0," Stuttgart, 1991.

[8] CiA, "CAN in Automation. History of CAN technology," Available at https://www.can-cia.org/can-knowledge/can/can-history/ (2022/11/27).

[9] CanKingdom International, "Cankingdom," Available at https://www.kvaser.com/about-can/higher-layer-protocols/cankingdom/ (2023/01/13).

[10] RT-Labs, "Rt-kernel," Available at https://rt-labs.com/product/rt-kernel/ (2023/01/17).

[11] L.-B. Fredriksson, "Time and event based message transmission," US Patent 8,037,226 B2 (2011/10/11).

[12] *KVASER MEMORATOR PRO 2XHS V2*, Available at https://www.kvaser.com/product/kvaser-memorator-pro-2xhs-v2/ (2022/11/24), Kvaser, 4 2013, rev. 3.

[13] *Kvaser Database Editor User's Guide*, Available at https://www.kvaser.com/software/7330130980334/V2/UG_98033_kvaser_database_editor_userguide.pdf (2022/01/17), Kvaser, 4 2010, rev. 2.

[14] J. Anderson, *Fundamentals of Aerodynamics*, 6th ed. New York, NY: McGraw-Hill Education, 2017.

[15] NASA, "Air mass/density," Available at https://www.earthdata.nasa.gov/topics/atmosphere/atmospheric-pressure/air-mass-density (2022/10/05).

[16] TE, "Ms4525do," Available at https://eu.mouser.com/datasheet/2/418/6/ENG_DS_MS4525DO_B10-1130293.pdf (2023/01/17).

[17] ——, "Interfacing to te digital pressure modules," Available at https://www.amsys.de/downloads/notes/I2C-Interface-to-Digital-Pressure-Sensors-AMSYS-an802e.pdf (2023/01/17).
[18] MathWorks, "mixsyn," Available at https://se.mathworks.com/help/robust/ref/lti.mixsyn.htmll (2022/10/02).
[19] B. Lennartsson, *Reglerteknikens grunder*, 4th ed. Lund: Studentlitteratur AB, 2000.
[20] MathWorks, "c2d," Available at https://se.mathworks.com/help/control/ref/lti.c2d.html (2022/10/08).
[21] ——, "Control systems in practice: 4 ways to implement a transfer function in code," Available at https://se.mathworks.com/videos/control-systems-in-practice-part-7-4-ways-to-implement-a-transfer-function-in-code-157796731
html (2022/11/02).
[22] B. Lennartson, "Introduction to discrete event systems," 2009, department of Signals and Systems, Chalmers University of Technology.
[23] MathWorks, "balred," Available at https://se.mathworks.com/help/ident/ref/lti.balred.html (2022/10/08).

A

Appendix 1

Message	Time [ms]	Message	Time [ms]
Imu	9	Pitch	108
Airspeed	10	Imu	109
Roll	11	Airspeed	110
Pitch	12	Throttle	111
Throttle	13	Yaw	112
Yaw	14	Imu	119
Imu	19	Airspeed	120
Airspeed	20	Roll	123
Roll	25	Pitch	124
Pitch	26	Throttle	125
Throttle	27	Yaw	126
Yaw	28	Imu	129
Imu	29	Airspeed	130
Airspeed	30	Roll	135
Roll	37	Pitch	136
Pitch	38	Throttle	137
Imu	39	Yaw	138
Airspeed	40	Imu	139
Throttle	41	Airspeed	140
Yaw	42	Imu	149
Imu	49	Airspeed	150
Airspeed	50	Roll	151
Roll	53	Pitch	152
Pitch	54	Throttle	153
Throttle	55	Yaw	154
Yaw	56	Imu	159
Imu	59	Airspeed	160
Airspeed	60	Roll	165
Roll	65	Pitch	166
Pitch	66	Throttle	167
Throttle	67	Yaw	168
Yaw	68	Imu	169
Imu	69	Airspeed	170
Airspeed	70	Roll	177
Imu	79	Pitch	178
Airspeed	80	Imu	179
Roll	81	Airspeed	180
Pitch	82	Throttle	181
Throttle	83	Yaw	182
Yaw	84	Imu	189

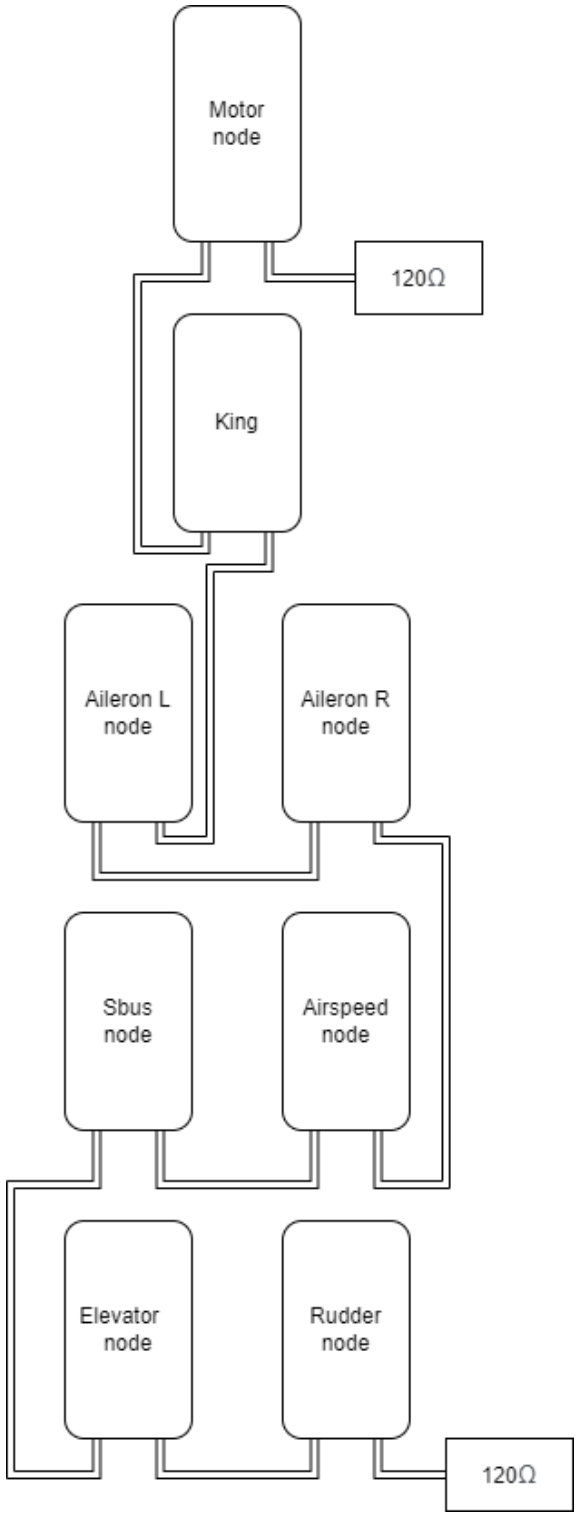

Figure A.1: Nodes on CAN bus in aircraft

Description	Specification
Model name	Skywalker Wall-E2000
Weight	3.2 kg
Angular momentum I_Y	0.27 kgm^2s^{-1}
Wingspan	2.0 m
Taillength	0.7 m
Tailarea	0.0656 m^2
Elevatorarea	0.016 m^2
Propeller diameter, pitch	0.28m, 0.2 m
Propulsion battery	22.2 v, 3.3 ah
Electronics battery	14.8 v, 3 ah
Motors	2816, 650 kv
Inverter	Hobbywing 40A
Maximum thrust, 0 apparent airspeed	40 N

Table A.2: Specification of UAV airframe

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY