Automated Landing of a Fixed-Wing Unmanned Aircraft onto a Moving Platform

Mohamed Zahier Parker

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Electronic) in the Faculty of Engineering at Stellenbosch University.

Supervisor: Prof. J.A.A. Engelbrecht
Department of Electrical and Electronic Engineering

March 2023

Acknowledgements

I would like to send my thanks and appreciation to the following people and organisations:

- My mother, father, grandparents and siblings for supporting me during my master's while I was at home. Thank you for helping me deal with my stress levels and assisting me in more ways than I can imagine.
- My supervisor Prof Japie Engelbrecht for guiding and assisting me during my project. Thank you for going out of your way to fulfil my requests and being available for the flight tests.
- The Department of Science and Innovation (DSI) and the Council for Scientific and Industrial Research (CSIR) for providing me with a bursary for my master's and also having excellent programs to assist me during my project.
- My flight test crew Andrew Murdoch, Clayton Pheiffer and Merrick Hughes for always being available and willing to wake up early for the flight tests.
- Dr Willem Jordaan and Dr Callen Fisher for your advice on ROS, PX4 autopilot software, 3D printing and mini PCs.
- The ESL students who help me with parking lot tests and provided me with advice regarding mechanical work and software.
- Mr Wessel Kroukamp, Mr Johan Arendse and Mr PH Petzer for assisting me with my airframe, electronics soldering and mass moment of inertia experiment.
- Michael Basson for being an excellent safety pilot and being available for my practical flight tests.
- Daniel for providing me with the thrust jig to test my motor.
- Dr Willie Smit for providing me with a Raspberry Pi 4.
- Prof Herman Kamper for this master's thesis template.

DECLARATION

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2023

UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Plagiaatverklaring / *Plagiarism Declaration*

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
   Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.

2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
   I agree that plagiarism is a punishable offence because it constitutes theft.

3. Ek verstaan ook dat direkte vertalings plagiaat is.
   I also understand that direct translations are plagiarism.

4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
   Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism

5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
   I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

Abstract

This thesis presents the development, implementation and practical testing of a control system that can automatically land a fixed-wing unmanned aerial vehicle (UAV) onto a moving platform. The control system consists of the flight control system and guidance control system. A landing strategy is proposed that is inspired by a real aircraft carrier landing, but is scaled down to the size of the fixed-wing UAV used in this research project. A prediction method is suggested to predict the touchdown point between the aircraft and the moving platform. A mathematical model of the fixed-wing aircraft was established to capture the aircraft's flight dynamics. The model was used to design the flight control system. The flight control system architecture combines classical control with model predictive control to control the local states of the aircraft. The model predictive controller was added to improve the landing accuracy of the aircraft, by having improved airspeed and altitude control compared to classical controllers. The guidance control system contains a guidance algorithm, waypoint scheduler, landing position predictor, and state machine to allow the aircraft to navigate around the airfield and land on the moving platform. The control systems were then implemented in PX4 autopilot software, which together with the Gazebo simulator, was used to perform software-in-the-loop simulations to verify the control systems' performance using a representative simulation model.

A new avionics stack was developed for the physical fixed-wing UAV using commercially available hardware and open-source software. A new fixed-wing UAV was assembled by mounting the newly developed avionics stack into an existing airframe. A new moving platform was also assembled by mounting commercially available hardware onto an RC car chassis. Practical flight tests were performed using the physical UAV to validate the control system's performance in practice. The developed control system was able to accurately land the physical fixed-wing UAV within a 3 m x 3 m static bounding box on a runway, and also on a virtual moving platform with the same dimensions travelling at 3 m/s (or 10 km/h).

Uittreksel

Hierdie tesis beskryf die ontwikkeling, implementering en praktiese toetsing van ’n beheerstelsel aan wat ’n vastevlerk onbemande vliegtuig outomaties op ’n bewegende platform kan land. Die beheerstelsel bestaan uit die vlugbeheerstelsel en leidingbeheerstelsel. ’n Landingstrategie word voorgestel wat geïnspireer is deur ’n regte vliegdekskip landing, maar is afgeskaal tot die grootte van die vastevlerk-UAV wat in hierdie navorsingsprojek gebruik word. ’n Voorspellingsmetode word voorgestel om die raakvalpunt tussen die vliegtuig en die bewegende platform te voorspel. ’n Wiskundige model van die vastevlerkvliegtuig is ontwikkel om die vliegtuig se vlugdinamika te beskryf. Die model is gebruik om die vlugbeheerstelsel te ontwerp. Die argitektuur van die vlugbeheerstelsel kombineer klassieke beheer met modelvoorspellingsbeheer om die vliegtuig se eie toestande te beheer. Die model voorspellende beheerder is bygevoeg om die landingsakkuraatheid van die vliegtuig te verbeter, deur verbeterde lugspoed- en hoogtebeheer in vergelyking met klassieke beheerders te hê. Die leidingbeheerstelsel bevat ’n leidingalgoritme, wegpuntskeduleerder, landingsposisievoorspeller en toestandmasjien om die vliegtuig in staat te stel om om die vliegveld te navigeer en op die bewegende platform te land. Die beheerstelsels is toe in PX4 Autopilot sagteware geïmplementeer, wat saam met die Gazebo-simulator gebruik is om sagteware-in-die-lus-simulasies uit te voer om die beheerstelsels se werkverrigting met ’n meer verteenwoordigende simulasiemodel te verifieer.

’n Nuwe avionika stelsel is ontwikkel vir die fisiese vastevlerkvliegtuie met behulp van kommersieel beskikbare hardeware en oopbronsagteware. ’n Nuwe vastevlerkvliegtuig is opgebou deur die nuutontwikkelde avionika stelsel in ’n bestaande vliegtuigraam te monteer. ’n Nuwe bewegende platform is ook opgebou deur kommersieel beskikbare hardeware op ’n RC-motoronderstel te monteer. Praktiese vlugtoetse is uitgevoer met behulp van die fisiese vliegtuig om die beheerstelsel se werkverrigting in die praktyk te valideer. Die ontwikkelde beheerstelsel kon die fisiese vastevlerkvliegtuig land binne ’n 3 m x 3 m teiken area op ’n aanloopbaan, en ook op ’n virtuele bewegende platform met dieselfde afmetings wat teen 3 m/s (of 10 km/h) beweeg.

Publications

The following paper has been published from the work done for this thesis:

- M. Z. Parker and J.A.A. Engelbrecht, "Precise automated landing of a fixed-wing aircraft onto a moving platform," MATEC Web Conf., vol. 370, p. 05007, 2022. [Online]. Available: https://doi.org/10.1051/matecconf/202237005007

Contents

List of Figures

List of Tables

Nomenclature

Greek letters

α	Angle of attack
β	Angle of sideslip
$\delta_a, \delta_e, \delta_f, \delta_r$	Deflection angles from trim for the aileron, elevator, flaps and rudder respectively
$\delta_A, \delta_E, \delta_F, \delta_R$	Deflection angles for the aileron, elevator, flaps and rudder respectively
$\Delta\dot{h}_{ref}$	Rate of climb rate reference
ΔT	Thrust magnitude from trim
ΔT_c	Thrust command from trim
$\Delta\Delta T_c$	Rate of thrust command from trim
γ	Glide slope angle
ϕ, θ, ψ	Roll, pitch and yaw angles from trim respectively
Φ, Θ, Ψ	Roll, pitch and yaw angles respectively
ψ_{crab}	Crab angle
Ψ_r	Runway heading from true north
ρ	Air density
τ_c	Filter time constant
τ_{crab}	Crab angle controller's rise time
τ_{ct}	First cross-track controller's rise time
τ_e	Motor time delay
ω_c	Filter centre frequency
ω_n	Natural frequency
ζ	Damping ratio

Small letters

b	Wing span
$\bar{c}$	Mean chord
e	Oswald efficiency factor
g	Gravitational acceleration
h	Altitude
$\dot{h}$	Climb rate
m	Mass
n_u	Control horizon
n_y	Prediction horizon
p, q, r	Roll, pitch and yaw rates from trim
u, v, w	Axial, lateral and normal velocity deviations from trim
$\bar{v}$	Airspeed from trim
x	In-track distance
y	Cross-track error

Capital letters

A	Aspect ratio
C_L, C_D	Lift and drag coefficients respectively
C_l, C_m, C_n	Roll, pitch and yaw moment coefficients respectively
C_x, C_y, C_z	Axial, lateral and normal force coefficients respectively
I	Moment of inertia
L, M, N	Roll, pitch and yaw moments respectively
P, Q, R	Roll, pitch and yaw rates respectively
S	Wing area
T	Transpose
T	Thrust magnitude
T_c	Thrust command
T_s	MPC sample time
U, V, W	Axial, lateral and normal velocities respectively
X, Y, Z	Axial, lateral and normal forces respectively

Acronyms and abbreviations

3D	Three dimensional
ABC	Acceleration-Based Control
ATOL	Automatic Take-off and Landing
AVL	Athena Vortex Lattice
CG	Centre of Gravity
CV	Computer Vision
DC	Direct Current
DCM	Direction Cosine Matrix
DGPS	Differential Global Positioning System
DLC	Direct Lift Control
EKF	Extended Kalman Filter
ESC	Electronic Speed Controller
ESL	Electronic Systems Laboratory
FCS	Flight Control System
GCS	Guidance Control System
GCSN	Ground Control Station
GPS	Global Positioning System
HITL	Hardware-in-the-Loop
IMU	Inertial Measurement Unit
LiPo	Lithium-Polymer
LSA	Lateral Specific Acceleration
LQR	Linear Quadratic Regulator
MIMO	Multiple-Input-Multiple-Output
MPC	Model Predictive Control
NED	North-East-Down
NMP	Non-Minimum Phase
NSA	Normal Specific Acceleration
NSADLC	Normal Specific Acceleration Direct Lift Control
PC	Personal Computer
PD	Proportional Derivative
PI	Proportional Integral

PID	Proportional Integral Derivative
PM	Power Management
PWM	Pulse-width modulation
QGC	QGroundControl
RC	Radio-control
RTK	Real-Time Kinematic
SITL	Software-in-the-Loop
TBS	Team BlackSheep
TECS	Total Energy Control System
UART	Universal asynchronous receiver-transmitter
UAV	Unmanned Aerial Vehicle
UI	User Interface
USB	Universal Serial Bus

Chapter 1

Introduction

1.1. Background

Unmanned Aerial Vehicles (UAVs) have been utilised for many years in both the military and civilian sectors. The first recorded use case of a UAV was in 1849 when Austrians tried to bomb Venice with balloons containing explosives [1]. The first pilotless aircraft were created during World War I by the British Army Royal Flying Corps and the Royal Navy. These aircraft were remote-controlled and were used to assist the war efforts [2]. As radio, sensing, and electronic technology developed, more advanced UAVs were created with increased capabilities. Modern military drones, such as the General Atomics MQ-9 UAV, have improved flight performance compared to earlier drones, as they have longer flight times and increased payload capacity, that allow them to be used in a variety of complex applications. These applications include both military and civil use cases. Examples of military use cases include surveillance and anti-warfare, while examples of civil use cases include law enforcement and search and rescue [3].

The increased number of applications for UAVs have generated a demand for safe and robust automatic systems to control them. According to Airbus, the most dangerous phase of flight, when the most accidents occur, is the landing phase [4]. Some of the factors leading to these accidents are severe weather conditions, aircraft component failure, and pilot fatigue. The use of an automated landing system can reduce these incidents as they can be designed with these factors in mind. Currently, automatic landing systems are commonly used in commercial airliners for landing in low visibility weather conditions [5]. Automatic landing systems are also used to land military jets onto aircraft carriers. The development of automatic landing systems enable even more applications for UAVs, such as fixed-wing UAV delivery systems using a moving home station, or the delivery of medical supplies to rural areas [6].

The problem to be solved for this research project is the creation of a system that can automatically land a fixed-wing UAV onto a moving platform. This scenario imitates the real-life landing of a jet aircraft onto an aircraft carrier in the ocean. The aircraft is normally manually piloted for the landing with the pilot using aids to align the aircraft with the aircraft carrier. This places excessive stress on the pilot which makes them prone

to errors. The US navy statistics [7] show that there have been a number of mishaps every year, however it has been on a downwards trend. These mishaps are quite costly, therefore it is desired to reduce the chances of them occurring. Landing on a platform in the ocean also provides its own challenges such as unpredictable movement of the platform and decreased visibility due to weather conditions. The use of an automated landing system can take these challenges into account and also provide consistent performance, no matter the pilot's fatigue level.

1.2. Project Goals and Objectives

The primary goal of this research project is to design, implement, and practically test a control system that can automatically land a fixed-wing UAV onto a moving platform. A secondary goal is to develop new avionics hardware and software for the fixed-wing UAV projects at the Electronic Systems Laboratory, by procuring, modifying, and integrating off-the-shelf hardware and open-source software, to replace the in-house developed avionics that have become obsolete. The project goals are broken down into the following objectives:

- Create a mathematical model that captures the flight dynamics of the fixed-wing aircraft so that it can be used to design and simulate the control systems.
- Design the low-level flight control system (FCS) to control the local states of the aircraft.
- Design the guidance control system (GCS) to provide commands to the FCS to allow the aircraft to follow a trajectory and land onto a moving platform.
- Implement the flight and guidance control systems in the autopilot software and test them in software-in-the-loop (SITL) simulations.
- Develop new avionics hardware and software for a fixed-wing UAV, as well as the associated ground station and communications systems.
- Assemble the fixed-wing aircraft and integrate the new avionics hardware.
- Design and build a moving platform capable of moving at a speed of 3 m/s to be used for the practical moving platform landing flight tests.
- Perform flight tests to practically demonstrate automatic landing of the physical UAV onto a moving platform.

By completing these objectives the goals of the project will be fulfilled. This will be evaluated in the conclusion to determine the project's success.

1.3. Project History

The Electronic Systems Laboratory (ESL), where this masters project is conducted, is a research group at Stellenbosch University that has a history of developing flight control systems for unmanned aerial vehicles. UAV projects at the ESL have involved both hardware and software development, starting as early as 2001. Flight control systems have been developed for various types of UAVs, including fixed-wing, rotary-wing, and multi-rotor vehicles. The major UAV research activity at the ESL was catalysed in 2005 when Peddle developed, implemented, and practically flight-tested a flight control system that enabled a fixed-wing UAV to automatically navigate a set of waypoints [8]. Peddle [9] subsequently improved his control system design by creating an acceleration-based control (ABC) architecture for the control of fixed-wing aircraft. This ABC architecture served as the basis for many subsequent projects.

One of the main research focus areas at the ESL has been the automatic landing of UAVs. The research was kicked off by Roos who developed a flight control system for autonomous take-off and landing of a fixed-wing UAV using normal GPS [10]. This was followed by several other projects on automatic landing of fixed-wing aircraft, including De Hart [11], Alberts [12], Smit [13], Le Roux [14], Hugo [15], and De Bruin [16].

Automatic landing systems have also been developed for rotary-wing UAVs. The research was kicked off by De Jager who developed a flight control system for automatic vision-based landing of a rotary-wing UAV [17]. This was followed by several other projects on automatic landing of rotary-wing and multi-rotor UAVs, including Swart [18], Möller [19], Fourie [20], Ioppo [21], Mfiri [22], and Grobler [23].

Three previous ESL projects investigated the automatic landing of UAVs onto moving platforms. However, only one of these projects was for a fixed-wing UAV. Möller practically demonstrated the automatic landing of a quadcopter UAV onto a moving platform [19]. Fourie practically demonstrated the automatic landing of a helicopter UAV onto a moving platform [20]. Le Roux attempted the automatic landing of a fixed-wing UAV onto a moving platform [14], but he was only able to demonstrate the moving platform landing in simulation. A hardware malfunction during one of his flight tests resulted in the loss of the physical UAV, and he was unfortunately not able to demonstrate the practical landing of the fixed-wing UAV onto the moving platform.

The hardware used in the previous ESL fixed-wing automation projects was developed in-house and has undergone various iterations. Some software components such as the ground station were also designed in-house. Even though this was a remarkable feat, it had its fair share of issues including no community support and reliability concerns. There were a few instances of aircraft crashing due to bugs or hardware issues that were not related to the control system design. Le Roux is a prime example of this as his UAV crashed due to an unknown bug. To increase the reliability of the system used, it is decided

to move to hardware and software that are commercially available with a proven track record and community support. This will provide increased confidence in the system's success in real-life.

1.4. Research Approach

The approach taken for some of the aspects of the research project are as follows:

- The new avionics hardware was developed using commercially available hardware that has been successfully used in many applications in the industry. The new avionics software is built on open-source software that has been successfully implemented on many different UAVs and has community support.
- DGPS is used to obtain the relative position and velocity of the aircraft and the moving platform instead of a vision-based approach. DGPS has recently become more affordable and can be used in this project. DGPS also does not have any major drawbacks in the environment where the practical flights will be performed.
- The flight control system uses a hybrid architecture combining classical control with model predictive control (MPC). The classical controllers were selected as they have been successfully used by many previous ESL students for fixed-wing UAV automated landing scenarios. The MPC was added to obtain improved landing accuracies compared to the classical controllers.
- For the practical flight tests, the new avionics hardware, individual flight controllers and aircraft waypoint navigation were first incrementally tested. Thereafter, the runway landing test with the fixed-wing UAV was performed. Finally, the moving platform landing test was executed, concluding the practical flight tests.
- For the practical moving platform landing, the fixed-wing UAV touches down on a virtual platform a few metres above the physical moving platform. This is because the physical moving platform is too small for the fixed-wing UAV to physically land on.
- No arrestor system was developed for the moving platform. This means that for the moving platform landing, the fixed-wing UAV is not captured on touchdown. The fixed-wing UAV will instead perform a go-around manoeuvre to land back on the runway.

1.5. Scope and Limitations

The scope of the research project includes:

- A DGPS-based moving platform landing. A vision-based moving platform landing is outside the scope of the project.
- The moving platform landing only considers the translational motion of the platform and is not designed for heaving, rolling or pitching motion.
- The flight tests only demonstrate an accurate "touchdown" on a moving platform. The fixed-wing UAV will not come to a complete stop on the platform, as the development of an arrestor system is outside the scope of the project.
- The moving platform will be manually controlled by a human driver, who will be alerted when to move the platform. An automated system for the platform is outside the scope of the project.

The limitations of the research project include:

- The DGPS system is dependent on GPS signal availability. Therefore, it cannot be used in environments with GPS jamming.
- The model predictive controller used in the flight control system is model-dependent. This means that the model used to capture the aircraft's flight dynamics needs to be as accurate as possible.
- No arrestor system is used. Therefore, the aircraft cannot be captured on the moving platform.
- The fixed-wing UAV will not be able to land in high crosswind weather conditions (more than 6 knots) as the UAV will be too crabbed to land. The UAV will also not be able to land in high wind conditions in general, as the control systems would not be able to maintain the glide slope for an accurate landing.

1.6. Thesis Outline

This thesis is split into nine chapters that presents different aspects of the research project. The layout of the thesis is as follows:

1. Chapter 1 provides a background of UAV development, the project's goals and objectives, the history of this project, the research approach, the project's scope and limitations, and the thesis outline.

2. Chapter 2 investigates aircraft carrier landing techniques and reviews the literature available for fixed-wing UAV control.

3. Chapter 3 presents the physical system used to practically test the moving platform landing, the simulator used for control system testing and the landing strategies considered in this project.

4. Chapter 4 presents the fixed-wing UAV non-linear and linear models which are used for simulation and control design respectively.

5. Chapter 5 describes the design of the flight control system which is used to control the local states of the aircraft. The linear models developed in chapter 4 are used for designing the controllers.

6. Chapter 6 presents the guidance control system which is used to navigate the aircraft around the airfield and land it onto the moving platform.

7. Chapter 7 discusses the implementation of the non-linear simulation and tests the designed flight and guidance control systems using this simulation.

8. Chapter 8 discusses the practical flight test logistics and flight test campaign used to test the developed control systems on the physical system introduced in chapter 3. The results from the practical flight tests are then analysed.

9. Chapter 9 provides a conclusion to the thesis and determines if the goals of the project have been accomplished. The research contributions that have been made by this research project are then highlighted. Finally, the recommendations for future research work are presented.

Chapter 2

Literature Review

This chapter first investigates the techniques used by manned aircraft to land on aircraft carriers so that the important characteristics of landing on a moving platform can be identified. Thereafter, a review is performed of previous literature both internal and external to the ESL. The review will first focus on systems used to land the fixed-wing UAV on a runway and then move to systems used to perform the moving platform landing. The key findings that were observed during the review will then be highlighted. This will be followed by a discussion on the design decisions taken for this research project.

2.1. Aircraft Carrier Landing

The systems used to perform the real-life scenario of landing a fixed-wing jet fighter aircraft onto an aircraft carrier are analysed in this section. This scenario is the main inspiration for creating the automated moving platform landing system. Therefore, the real-life systems should be investigated to understand what aspects of the landing should be considered.

For the real-life scenario, the pilot manually lands the jet onto the aircraft carrier runway with the aid of the Fresnel-lens optical landing system [24]. The pilot aims to keep the aircraft on a glide slope to intersect the carrier. The optical system will visually indicate to the pilot their location with respect to the glide slope using a series of lights as shown in Figure 2.1.

Figure 2.1: Image showing the Fresnel-lens optical landing system onboard an aircraft carrier [24].

The optical system is ineffective in low visual weather conditions such as dense fog. Pilots are often diverted to a land-based airfield if there were no improvements in weather conditions [25]. A GPS-based system would not suffer from this issue as it operates in all weather conditions. The jet fighter does have additional systems that can function in low visibility conditions. However, for safety purposes, the jet can only land on the carrier if visual contact is maintained.

The aircraft carrier runway is only around 150 m long, which is shorter than the length required to land the jet aircraft. This problem is solved by using an arrestor system with four arresting wires to slow the aircraft down [26]. The pilot aims to touch down the jet on one of these wires so that the jet's tailhook attaches and slows the aircraft down, as shown in Figure 2.2. The aim can only be achieved if the aircraft lands with high accuracy. The automated landing system designed for this research project will also need a high landing accuracy to land the UAV onto a moving platform.

Figure 2.2: Image showing the arrestor system on an aircraft carrier slowing the fighter jet down [24].

For the aircraft carrier landing, the carrier generally moves at around 30 knots in the direction of the wind to allow the jet to have a lower ground speed for the landing [27]. This makes it easier for the pilot to aim for the touchdown point, as the relative speed between the jet and aircraft carrier is lower. For this research project, the moving platform will only be able to move at a certain heading due to environmental factors. This means the headwind's effect cannot be exploited to help the UAV land. Instead, the designed control systems would have to account for the wind's effect.

The automated landing system developed in this research project could be applied to land fixed-wing UAVs onto aircraft carriers. This requires high in-track landing accuracy, but is more relaxed in terms of cross-track landing accuracy. An example commercial application would be to have a moving, ground-based delivery vehicle with multiple fixed-

wing UAVs that take off and land to perform local deliveries. This application would require both high in-track landing accuracy and high cross-track landing accuracy. For this reason, the system will be developed so that it is able to land on a small moving platform with a landing zone of 3m x 3m.

Literature regarding the landing of military drones onto aircraft carriers is scarce due to the high-level secrecy of the research. The focus will therefore be shifted to the literature available in the public domain.

2.2. Previous Research

Published literature regarding fixed-wing UAV control systems was consulted to determine the state of the art and to identify any research gaps that exist for systems that can automatically land fixed-wing UAVs onto moving platforms. This will provide insight into the different solutions that exist and if they are successful. As previously mentioned, the ESL has a long history of developing control systems for fixed-wing UAVs. The automated landing of a fixed-wing UAV onto a moving platform is the next step in the ESL's research on the automated landing of UAVs. Therefore, an overview is first provided of previous ESL research projects on flight control for fixed-wing UAVs and automated landing of UAVs. This is then followed by a general literature review of external research on the automated landing of UAVs.

2.2.1 Internal ESL Research Projects

Peddle was one of the first pioneers in the ESL to develop a control system for a fixed-wing UAV as represented by his master's and PhD research projects. For Peddle's master's [8], he developed a classical control system to navigate around an airfield using waypoints. His control system performed well as he was able to practically verify its performance. For Peddle's PhD [9], he developed an acceleration-based control (ABC) architecture that further improved the fixed-wing UAV's control. In 2007, Roos [10] built on Peddle's master's work to design a system that could perform automatic take-off and landing (ATOL) for a fixed-wing UAV. Roos's control system structure used sequences for take-off and landing. These sequences would be an early interpretation of a state machine. This design method would be used by many research projects from then onward. An accurate landing was not a primary focus for Roos as his system could not achieve the accurate altitude control required for such a landing. Nonetheless, Roos's system was able to land his UAV on the runway both in simulation and in practice.

In 2008, Visser [28] used Peddle's ABC architecture combined with a vision-based controller to create an accurate landing system. While his system provided adequate lateral tracking performance, the altitude tracking was insufficient to perform a precise

landing. Unfortunately, during one of the flight tests his UAV became unresponsive and crashed, creating irreparable damage which halted his practical testing.

In 2012, Alberts [12] used Peddle's ABC architecture, but augmented a direct lift control (DLC) component to the normal specific acceleration (NSA) controller. DLC uses the flaps to generate lift without significantly changing the aircraft's incidence angle [12]. The newly formed NSA controller commanded both the elevator and flaps so that longitudinal control could be achieved from both pitch moment based and direct lift based actuation. The NSA controller splits the NSA reference and state into high-frequency and low-frequency components using high pass and low pass filters. The high-frequency components are sent to the direct-lift portion of the NSA controller, while the low-frequency components are sent to the moment-based portion of the NSA controller. Alberts's design significantly improved the wind disturbance rejection capabilities of the UAV, which is necessary when landing in adverse wind conditions. A deficiency in Alberts's design was the inability of the altitude controller to track a ramp reference, and hence the glide slope, with zero steady-state error. The altitude response had an offset from the glide slope, which caused a large longitudinal error on touchdown. Alberts addressed this issue by offsetting the glide slope reference by the steady-state error so that the aircraft would land at the desired touchdown point. With the solution applied, Alberts's aircraft managed to land within 3 m of the intended touchdown point. While this accuracy would be sufficient for a runway, it would not be acceptable for landing on a moving platform. It is imperative that the control system landing accuracy is precise (less than 1 m) for a moving platform landing, as the platform is a small target for the UAV.

In 2013 Smit [13] designed a control system to perform an accurate landing on a stationary platform using a Differential Global Positioning System (DGPS). Smit's control system utilised Peddle's ABC architecture with the addition of passive flap selection, where the flaps were set to a predetermined deflection angle allowing the aircraft to land at a slower speed. Smit also used a new airframe which he modelled using the Athena Vortex Lattice (AVL) method for his control system development. For the practical flight tests, Smit's UAV landed within 7.5 m of the intended touchdown point. This accuracy is unsatisfactory for a moving platform landing.

In 2017 De Bruin [16] developed a control system that could accurately land a fixed-wing UAV on a runway in crosswind conditions. De Bruin's control system also utilised Peddle's ABC architecture with Alberts's DLC component to the NSA controller. De Bruin modified Alberts's NSA controller architecture by removing the low-pass filter to simplify the controller design. De Bruin solved Alberts's issue of the altitude controller not tracking the glide slope by feed-forwarding the desired climb rate as a reference to the climb rate controller. The altitude controller would then reject disturbances as the aircraft descended on the glide slope. De Bruin also added a limited integrator to the altitude controller to account for climb rate biases. For the lateral controllers, De Bruin

used Peddle's design as a base and then added his own components to allow the UAV to execute the different crosswind landing techniques he was testing. The landing techniques he considered were crabbed, de-crab, and low-wing. De Bruin found that the de-crab manoeuvre produced the best result for a crosswind landing. De Bruin's UAV landed on the runway within 0.5 m of the intended touchdown point both in simulation and in practice. This accuracy is very high, especially considering that the UAV is around 2 m long. This landing accuracy would be sufficient for a moving platform landing as the platform would, at minimum, be the size of the UAV.

Peddle's ABC architecure was also used in ESL projects that focused on the control of the fixed-wing UAVs in different configurations, for example Hugo [15] and Goosen [29]. Hugo's system was designed to land a UAV with partial stabiliser and wing losses. Goosen constructed a system to recover a fixed-wing UAV from upset conditions.

Only three previous research projects in the ESL considered landing a UAV onto a moving platform, with only one using a fixed-wing UAV. The first project performed by Möller [19] considered landing a quad-copter onto the platform. Even though the control system used by Möller is not directly applicable to this research project, the decisions he made for his practical implementation should be considered. Möller utilised a Novatel DGPS to obtain a centimetre level relative position accuracy between the UAV and moving platform. He altered his DGPS operation to place it in ALIGN mode so that he could use it for his practical tests. Möller used a 2 m by 2 m trailer as the moving platform for the practical flight tests. The trailer was towed by a tow vehicle during the tests. His control system landed the quad-copter onto the moving platform both in simulation and in practice.

The second research project, performed by Fourie [20], consisted of autonomously landing an unmanned helicopter (rotary-wing UAV) onto a moving platform. Fourie's system was split into two sub-systems, namely the navigation and control sub-system and the safe-landing sub-system. Similar to Möller, the design of Fourie's system is not directly applicable to this research project, due to the helicopter's dynamics being different to those of a fixed-wing UAV. However, his implementation decisions are worth considering. Fourie used the same Novatel DGPS as Möller in ALIGN mode to obtain the relative position between the UAV and the moving platform. Fourie also used a trailer as the moving platform for his practical flight tests. However, unlike Möller, Fourie's platform had a size of 3 m by 3 m. Fourie's helicopter landed successfully on the moving platform in hardware-in-the-loop simulation and in practical tests. The physical helicopter could land on a platform that moved up to 3 m/s, with a land accuracy within 27 cm, which is remarkable.

The third project, performed by Le Roux [14], focused on landing a fixed-wing UAV onto a moving platform. Le Roux's longitudinal control system contained a combination of a total energy control system (TECS) for the outer loop and an NSA controller for

the inner loop. His lateral control system used a standard architecture implemented by many previous students. However, he, added a hybrid cross-track controller design that functioned based on the aircraft's distance to its trajectory. This new design solved a major problem of the autopilot being unable to be activated far from its reference circuit. To land on the moving platform, Le Roux devised an algorithm that would predict the touchdown point between the UAV and the moving platform. The algorithm worked for his scenario due to the simplistic motion of the platform. In the simulation, Leroux's altitude controller had a fast response with minimal oscillation, which is desired for an accurate landing. As a result, his system could accurately track the glide slope and, on average, land within 13 cm of the touchdown point, which is impressive. Unfortunately, he could not verify his system's landing performance with the physical UAV due to a malfunction in the hardware. The few controllers that were practically tested contained excessive oscillation. Therefore he would have had to retune them to practically test his landing scenarios. Le Roux planned to use a 3 m by 3 m trailer for his moving platform, similar to Fourie. However, Le Roux could not perform his practical moving platform landing test as his UAV crashed.

The hardware and software used in previous ESL fixed-wing projects were developed during the early stages of ESL research in fixed-wing UAVs by a group of ESL students. The hardware and software went through many iterations to bring improvements. This system, however, was no longer maintainable due to the in-house developed hardware and software becoming obsolete with limited availability and support. The in-house developed hardware was also not as reliable as commercial-off-the-shelf hardware, and hardware malfunctions during flight tests caused a number of UAVs to crash over the years. The crashes abruptly ended the students' practical flight testing, even though the fault was in no manner caused by their control systems. ESL students who experienced crashes were Visser [28], Le Roux [14], and Hugo [15], who all lost their UAVs due to malfunctions during flight tests, which ended their respective flight test campaigns. De Bruin [16] also lost his UAV due to a malfunction during a flight test, but then built a second aircraft and completed his practical flight testing. The most common cause of the crashes was a servo board failure resulting in the UAV becoming unresponsive. The servo board was used to switch between the autopilot and safety pilot (manual control) commands. Conventional manned aircraft have redundant systems when switching between manual and autopilot control; however, such a system is too complex and expensive for a low-cost fixed-wing UAV. Alternatively, switching control via software would be more reliable than the servo board and still be inexpensive enough to be within the realm of possibility for the UAV.

An ESL multi-rotor research program was developed concurrently with the ESL fixed-wing research program. During the multi-rotor program, it was decided to standardise the hardware and software so that the students would not waste unnecessary time developing components that were not vital for their projects. Ultimately, they chose to go with PX4

autopilot software and Pixhawk 4 autopilot hardware, as they have been proven to work on a commercial level and are widely adopted by multi-rotor UAV developers. PX4 has built-in redundancies with failure modes and a proven safety record.

For this research project, it was decided to follow in their footsteps by standardising the hardware and software for the fixed-wing UAV. This will allow future fixed-wing UAV students to only use development time on their applications. The exact software and hardware that will be used for the fixed-wing UAV will be discussed in chapter 3; however, PX4 supports fixed-wing UAVs which makes it a prime candidate for consideration.

2.2.2 External Research

There is very limited research on performing a moving platform landing with a fixed-wing aircraft. This literature review will therefore first cover previous research on automated landing of fixed-wing UAVs on normal runways. This will then be followed by previous research on automated landing of fixed-wing UAVs onto moving platforms.

Jantawong et al. [30] designed a system to land a fixed-wing UAV on a runway using GPS and relative barometric pressure. The exact control system used was not specified; however, the landing scheme included a flare manoeuvre to help reduce shock forces on the aircraft, and a final leg offset from the runway to compensate for crosswind. Their landing system achieved a landing accuracy of 4.3 m, which is adequate for landing on a runway, but not accurate enough to land on a moving platform.

Dharmawan et al. [31] developed an auto-landing system for a fixed-wing UAV to land on a runway. The control system utilised the linear quadratic regulator (LQR) method. The UAV used both a barometer and an ultrasonic range finder as altitude sensors for the UAV. The LQR regulated the UAV's roll, pitch, and yaw angles by commanding the roll, pitch, and yaw moments respectively. These moments are converted to the appropriate control surface deflection angles to produce the change in attitude. The LQR is commanded by an external landing controller which controls the UAV's altitude. The LQR was able to maintain the UAV's commanded attitude during the landing. The landing controller initially struggled to maintain the glide slope but was able to catch it towards the end of the response.

Brukarczyk et al. [32] presented a vision-based system that identifies ground signs to determine a glide path used for landing. The vision system provides inputs to the control system, which imitates a human pilot's control for landing the aircraft. The control algorithms used a fuzzy logic expert system approach. The vision system kept the ground target centred within its line of sight by adjusting the aircraft's altitude, hence maintaining a constant glide path. The entire system was tested both in software-in-the-loop (SITL) and hardware-in-the-loop (HITL) with fairly promising results. A drawback of using this system is that it requires clear weather with no obstructions between the UAV and the target. This cannot be guaranteed in a practical scenario due to mist and obstacles such

as trees. If this system were to be used for a moving platform scenario, the aircraft would struggle to keep the moving target in the centre of its line of sight, due to it not being agile enough. The aircraft would therefore not be sufficiently stabilised for landing on a moving platform.

Laiacker et al. [33] designed a vision-aided automatic landing system for a fixed-wing UAV to land on a runway. The vision system was used to detect and track runways. The landing system used a combination of vision data, DGPS data, and other sensor measurements to obtain the states of the UAV. The control system used cascaded PID methods with gain scheduling and non-linear elements. Three separate controllers were designed to control the altitude, course, and airspeed of the UAV independently of each other. In the flight practical tests, the system was able to land the UAV at the specified location on the runway. Laiacker et al. [33] planned to implement this system to automatically land the UAV on a mobile ground vehicle in the future. The advantage of using both the DGPS and vision-based system is that it increases the functionality of the overall system; because if one of the position acquisition methods were to become unreliable, then the other method could take over. The disadvantage of using both the DGPS and vision-based system is that it is more costly, complex, and time-consuming to implement. Using both position acquisition methods would also add additional weight to the UAV, which is not ideal.

A paper submitted by Santos et al. [34] highlighted a system to land a fixed-wing UAV onto a moving patrol boat. The system consisted of using computer vision (CV) to obtain the position of the UAV relative to the landing zone on the patrol boat. The CV method was chosen due to the possibility of jamming being present in the environment, rendering conventional GPS modules unusable. Two different approaches for CV were considered, namely, airborne and ground-based. The airborne approach consisted of obtaining the relative position of the UAV itself using an onboard camera and external markers. The UAV's computer would then calculate the landing trajectory. The ground-based approach used a monocular vision system at the landing area to obtain the relative position. The landing trajectory was calculated on the ground station and was then transmitted to the UAV. The airborne approach did not function correctly due to the simple image processing algorithms used, which were required because of the UAV's low processing power. The ground-based approach provided more accurate results and was suitable for the landing. The ground-based CV system is a good alternative for a DGPS system in jamming environments. However, it is more complex than the DGPS and requires good visibility between the landing area and the UAV.

Wang et al. [35] investigated the use of differential games to land a fixed-wing UAV onto a moving platform. The method essentially treats the UAV and platform as a pursuit-evasion problem. The UAV aims to reach the platform while the platform tries to evade the UAV. This is regarded as the worst-case scenario for the UAV landing on

the platform. In the linear simulation, the UAV was able to land on the platform with fairly good accuracy; however, non-linear simulation and practical flight tests were not performed to verify the results.

Feng et al. [36] created a model predictive control (MPC) based system to land a multi-rotor UAV onto a moving platform. As the UAV is a multi-rotor type, the design of the MPC is not directly applicable to fixed-wing UAVs. However, the general architecture of the MPC could be adapted for a fixed-wing UAV. Feng et al. used a linear MPC architecture to control the position of the quad-copter. The MPC performs extraordinarily well as it has a fast and minimal oscillation in its response. The MPC's good performance consequently results in the UAV having a landing accuracy of 37cm when landing on a moving platform with wind disturbances, which is impressive.

Persson [37] developed an MPC-based control system to land a fixed-wing UAV onto a moving ground vehicle. Two different types of MPC control schemes were tested. The first scheme is a linear MPC that controls the relative position between the UAV and the ground vehicle. The MPC achieves its control by commanding the UAV's and ground vehicle's acceleration and heading angle as well as the UAV's flight path angle. The second MPC scheme consists of two smaller MPCs that control the UAV's horizontal and vertical dynamics separately from each other. The horizontal MPC drives the relative horizontal position and heading between the UAV and ground vehicle to zero. The vertical MPC controls the UAV's altitude. When simulating both MPCs with a non-linear model, the second MPC with the separated control was found to be superior. This MPC's performance was then verified using the FlightGear simulator, and on touchdown, its landing accuracy was at the centimetre level, which is remarkable. The MPC was then compared to a PD controller for the same landing scenario and it was found that the MPC was superior with a faster response. A drawback of using the MPC is that it requires a lot of computational power. This was noted by Persson as she used a powerful desktop computer to execute her controller.

2.2.3 Findings and Design Decisions

After consulting the literature, the following key findings were made:

- There has been substantial research done on designing different control systems for the automated landing of a fixed-wing UAV; however, only a limited few considered landing the fixed-wing UAV onto a moving platform. An even smaller group made a physically viable system to test the moving platform landing in the real world.
- A diverse set of control system architectures were used to control the fixed-wing UAV, such as the ABC architecture, LQR, and MPC. The ABC architecture was the most popular in the ESL, seeing that it was used by many students for different applications.

- DGPS and vision-based systems were the two most common approaches to accurately sense the UAV's position relative to the runway or moving platform.
- The hardware and software used for the previous ESL fixed-wing UAVs are no longer maintainable and need to be replaced for this research project.
- Previous ESL moving platform landing projects used a trailer towed by a vehicle as the moving platform. For this project, the same full-size trailer cannot be used, as it is too large to fit on the runway where the practical flight tests are performed. Therefore, an alternative solution for the moving platform is required.

Based on the literature review and the key findings, the following decisions were made for this project:

1. The flight control system will consist of a hybrid architecture combining classical control with model predictive control (MPC).
   (a) The classical controllers are based on those used by De Bruin [16] and Le Roux [14]. De Bruin's design was chosen as his system could land the UAV on the runway with high accuracy in non-optimal windy weather. Peddle's ABC architecture, used by De Bruin, was proven to be robust as it was successfully implemented for many ESL projects testing different applications. The de-crab manoeuvre implemented by De Bruin will also be included into the control design so that the UAV can land on the platform in crosswind conditions. The de-crab manoeuvre ensures that the UAV is aligned with the moving platform on touchdown so that it does not experience excessive lateral forces, which can cause the UAV to tip over and experience a wing strike. Le Roux's hybrid guidance controllers are implemented to allow the aircraft to join the circuit when far from the ground track.
   (b) The MPC was added to the system to obtain an improved landing accuracy, as Persson [37] showed that their MPC achieved superior landing performance compared to a standard PD controller. The MPC can replace the slower classical outer-loop controllers, improving the UAV's altitude tracking performance and hence the landing accuracy. The MPC architecture is derived from Amadi [38] as his design was lightweight enough to run on a Pixhawk, which is a low computational power device. Amadi's MPC design has to be modified to work for a fixed-wing UAV, as his design was for a multi-copter UAV. For this project, it is planned to run the MPC algorithm on a companion computer mounted on the fixed-wing UAV. For it to successfully perform this task, the MPC needs to be efficient, hence the decision to use Amadi's architecture.

2. The landing algorithm will be similar to the approach used by Le Roux, but will need to be improved to make it capable of operating on a physical vehicle. Le Roux's guidance system worked well in simulation, but is untested on a physical vehicle.

3. A real-time kinematic (RTK) positioning DGPS will be used as the sensor to obtain the position of the UAV. This sensor was chosen as both Möller [19] and Fourie [20] performed their moving platform landings with a DGPS. Le Roux was also planning to use a DGPS for his system. These successful implementations indicate that the DGPS would also be suitable for landing a fixed-wing UAV onto a moving platform. The DGPS operates in all weather conditions compared to a vision-based system. DGPS systems have also become less expensive over the years and are now affordable to use for this research project.

4. The new fixed-wing UAV hardware and software will be developed using commercially available hardware and open-source software to increase the system reliability and maintainability from the previous ESL fixed-UAV hardware and software. The new hardware and software will be standardised with available support, so that it would be maintainable for future ESL fixed-wing UAV projects.

5. An RC car will be used as the moving platform, as it can easily fit on the runway at the airfield where the practical flights are performed. The RC car is too small for the physical fixed-wing UAV to land on therefore a virtual platform a few meters above the RC car is used as the target for the UAV during the landing. The RC car is limited to travel at a speed of 3 m/s to ensure that it does not overshoot the runway during the practical moving platform landing test.

Chapter 3

System Overview

This chapter provides an overview of the physical system that was developed to demonstrate the automated landing of a fixed-wing UAV on a moving platform. First, the fixed-wing UAV, the moving platform, and the stationary ground station are presented. Next, the simulation environment that was used for development and testing by performing software-in-the-loop simulation, is described. Finally, the proposed landing strategy that was followed, is introduced.

3.1. Physical System for Moving Platform Landing

A block diagram of the physical system that was developed to demonstrate the landing of a fixed-wing aircraft onto a moving platform is shown in Figure 3.1. The system consists of three main components: the Fixed-Wing UAV, the Moving Platform, and the Stationary Ground Station.

The aircraft is represented by a fixed-wing UAV and it contains a control system to automatically manoeuvre the UAV. The moving platform is represented by an RC car which contains all the components to move as desired and provide the UAV with its states. The ground station is used to view the states of the UAV as it flies and to remotely change the operating mode of the UAV. Communication links are used to transmit and receive data wirelessly between the elements in Figure 3.1. The functionalities of the communication links are as follows:

- The mobile ground station communicates with the UAV using a set of telemetry modules.
- The safety pilot controls the UAV via radio control using the UAV RC transmitter that is linked to the RC receiver on the UAV.
- The moving platform driver controls the motion of the moving platform via radio control using the moving platform RC transmitter that is linked to the RC receiver on the moving platform.
- The stationary ground station laptop connects to the mobile ground station computer

via wifi. The ground station operator uses remote desktop to access the ground control software running on the mobile ground station.

- The DGPS modules on the mobile ground station and the fixed-wing UAV receive the GPS signals from the GPS satellites. When operating in RTK GPS mode, the RTK packets from the DGPS base module are transmitted to the DGPS rover module on the fixed-wing UAV via the telemetry modules.

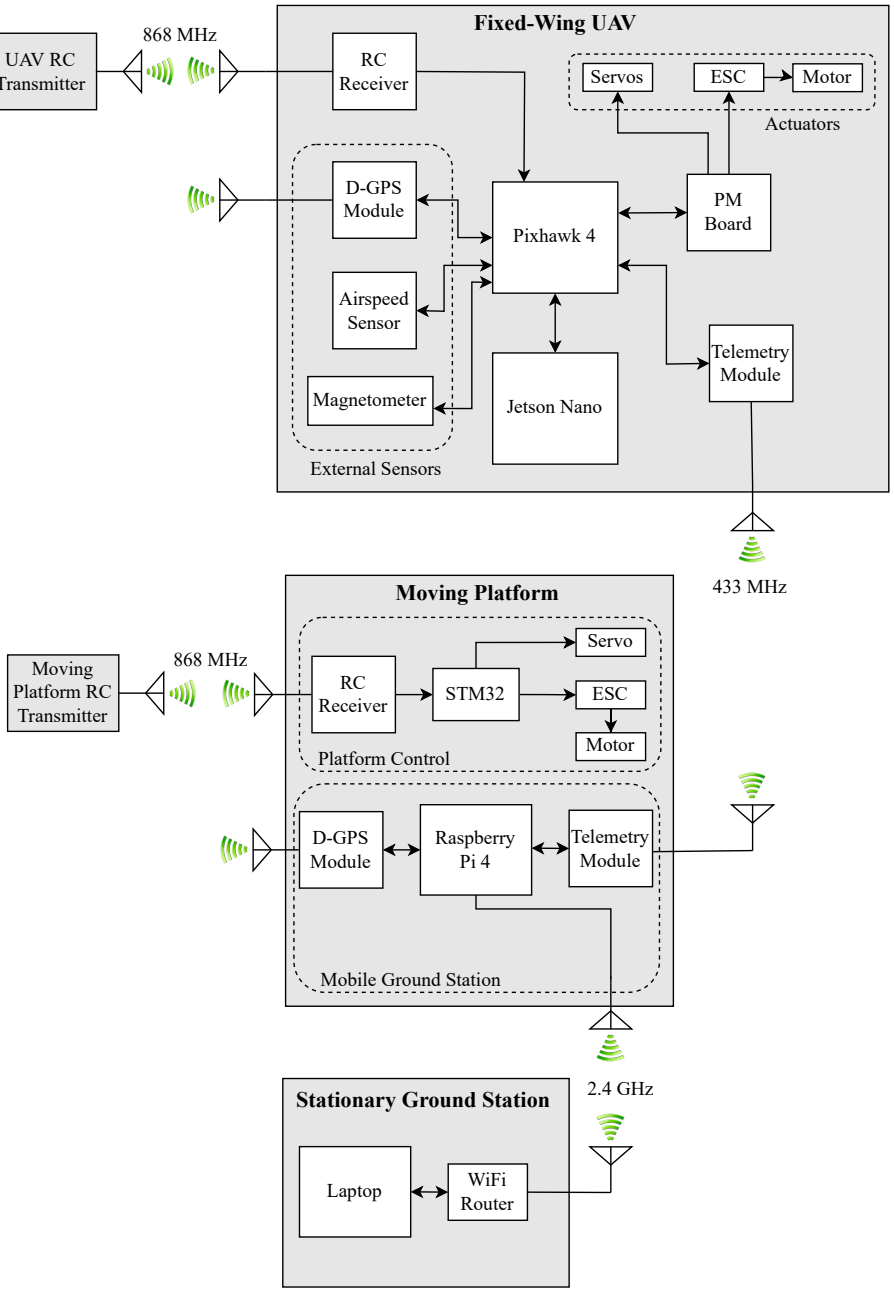


Figure 3.1: Overview of the physical system used to demonstrate the moving platform landing.

The individual components of the physical system are discussed in the following sections.

3.1.1 Fixed-Wing UAV Setup

The fully-assembled fixed-wing UAV is shown in Figure 3.2.

Figure 3.2: Fully assembled UAV used for practical flight tests.

The fixed-wing UAV consists of an airframe and an avionics system. The airframe is a Phoenix 0.60 size trainer that was constructed by De Bruin, in a previous Master's project [16]. The airframe required some maintenance and repairs. The motor and servos were kept, but the electronic speed controller (ESC) for the motor was replaced as there were too many uncertainties regarding the previous ESC. The new ESC is a Hobbywing FLYFUN-120A-6S-V5. The avionics system developed by De Bruin was no longer maintainable, and was completely replaced by the new avionics system developed in this project. The new avionics system consists of both hardware and software elements, which will be discussed separately on the following subsections.

3.1.1.1 Avionics Hardware

A block diagram of the avionics hardware components and their associated communication protocols is shown in Figure 3.3.

The new avionics hardware was designed to use the following off-the-shelf hardware components:

- a Pixhawk 4 autopilot to run the normal flight control software,
- an NVIDIA Jetson Nano single board computer to run the more computationally expensive Model Predictive Control (MPC) algorithm,
- a Power Management (PM) board to provide power to the Pixhawk and to transmit the PWM signals from the Pixhawk to the electronic speed controller that controls the motor, and to the servos that actuate the aircraft's control surfaces,

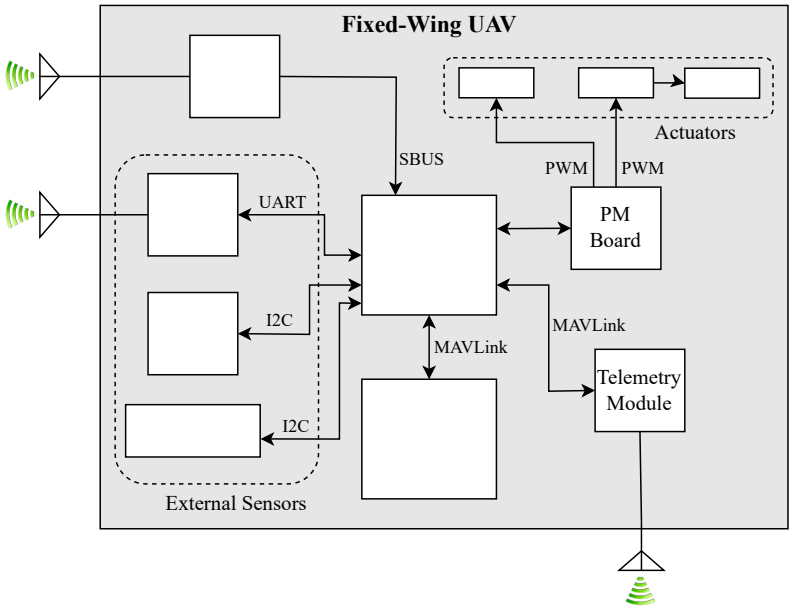
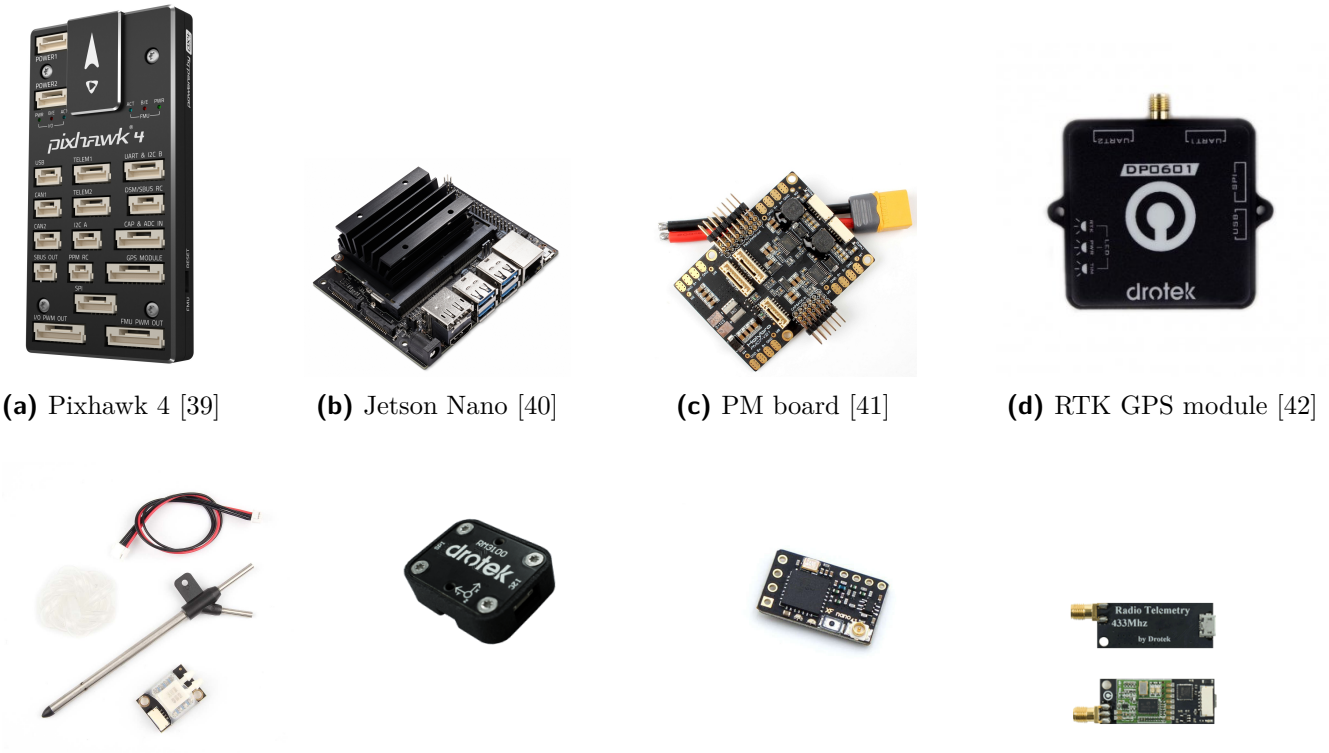


Figure 3.3: Block diagram of the avionics hardware and their associated communication protocols.

- various sensors to obtain the state measurements of the UAV including:
  - a Drotek DP0601 RTK differential GPS sensor,
  - a Holybro digital airspeed sensor,
  - a Drotek RM3100 magnetometer,
- a TBS Crossfire Nano receiver to receive the remote control commands from the RC transmitter operated by the human safety pilot,
- a Drotek 433 MHz telemetry module and antenna for communicating with the ground control station.

Images of the individual avionics hardware components are shown in Figure 3.4.

The Pixhawk 4, shown in Figure 3.4a, was chosen as it was decided to develop new flight control software using the PX4 Autopilot software stack, as will be mentioned later in this section. The Pixhawk 4 hardware supports the PX4 Autopilot software, and has also been successfully used in recent ESL projects and in industry. Since it is commercially produced and widely used, the Pixhawk 4 hardware is very reliable, which is very important for the aircraft's safety due to the high risk nature of flight testing. If the avionics were to fail, the aircraft would almost certainly crash, resulting in the loss of both the airframe and the avionics hardware. The Pixhawk 4 also supports many commercially available sensors which would be required to perform the precise automated landing attempted in this research project. An unused Pixhawk 4 was already available in the laboratory, which saved on costs and further encouraged its usage. The Pixhawk 4 contains an internal inertial measurement unit (IMU), barometer, and magnetometer which provide sensor



(a) Pixhawk 4 [39] **(b)** Jetson Nano [40] **(c)** PM board [41] **(d)** RTK GPS module [42]

(e) Airspeed sensor [43] **(f)** Drotek magnetometer [44] **(g)** Crossfire receiver [45] **(h)** Telemetry modules [46]

Figure 3.4: Individual hardware components forming the avionics.

measurements that can be used together with those provided by the external sensors to estimate the states of the aircraft.

The MPC is too computationally expensive to run on the Pixhawk, which was proven by Amadi [38], as he had to disable a significant amount of features to get it to work. A companion computer can instead be used to run the MPC algorithm and the Nvidia Jetson Nano, shown in Figure 3.4b, is chosen to fulfil this role. The Jetson Nano is connected to the Pixhawk via MAVLink to send and receive data.

The fixed-wing UAV is powered by one 6S 5000 mAh LiPo battery whose voltage is too high to power the avionics directly. Instead, the PM board, shown in Figure 3.4c, contains two voltage regulators to convert the voltage to a usable level for the Pixhawk. The second regulator is added as a redundancy to increase system reliability. The PM board also measures the battery voltage and sends it to ground station software to be monitored. The software alerts the operator if the battery is too low and in response, the operator will command the pilot to land the UAV immediately. A 5V 5A voltage regulator is also used to power the Jetson Nano from the LiPo. The ESC contains a Battery Elimination Circuit (BEC) to power the servos.

It was decided to only use one Lipo battery for the UAV for weight savings and hardware simplicity. Debruin [16] used three batteries on his UAV one of which served as a backup. Adding an additional battery would be more detrimental to the UAV due to the added weight and increased system complexity than any benefits it would provide. If any

battery were to fail in flight due to combustion, the UAV would be irrecoverable in any case. The 6S battery in this research project was operated between 50% - 100 % battery life range, so that thrust produced by the motor was approximately constant and the UAV would not be in a position to run out of battery life. For the specified battery range the UAV flew for approximately 7 minutes. Three 6S batteries were available for a flight test day and could be swapped out between flight sessions. This extended the flight test time to 21 minutes which was sufficient to perform all the tasks delegated to a flight test day.

The Drotek RTK DGPS, shown in Figure 3.4d, is used to obtain the position of the UAV at the centimeter level which is important for a high precision landing. A DGPS was chosen as it has been successfully used by previous ESL research projects. This specific Drotek GPS was chosen as it is supported by the Pixhawk 4 and autopilot software. The RTK GPS works by using a rover module connected to the UAV and a base module connected to the ground station. The RTK GPS operation will be discussed in more detail in section B.2.1. The GPS module is connected to the Pixhawk 4 via UART.

The Holybro airspeed sensor, shown in Figure 3.4e, measures the differential pressure in a pitot tube to determine the airspeed of the aircraft. This airspeed sensor was chosen as it is supported ”out of the box” by the Pixhawk. The airspeed sensor is connected to the Pixhawk via an I2C interface.

The Pixhawk’s magnetometer was found to be too susceptible to interference, as it produced incorrect measurements due to noise and a large bias error. An external Drotek magnetometer, shown in Figure 3.4f, was therefore added to the avionics to be used instead of the Pixhawk magnetometer, as the Drotek magnetometer had less noisy measurements. The Drotek magnetometer is also connected to the Pixhawk via an I2C interface.

The TBS RC receiver, shown in Figure 3.4g, receives remote control commands from the RC transmitter operated by the human safety pilot and transmits them to the Pixhawk autopilot. The TBS crossfire transmitter and receiver package were chosen as it has a very long range which is more than what is required for the flight tests with the physical UAV. It also has many safety protocols to protect against signal loss. The RC receiver is connected to the Pixhawk via an SBUS interface. The RC crossfire package operates at 868 MHz which is a different frequency from the telemetry module, preventing interference between the two communication channels.

The Drotek 433 MHz telemetry module, shown in Figure 3.4h, is used for communication between the UAV and the ground control station. . The UAV receives commands from and transmits telemetry to the ground control station via this communications link. The telemetry is monitored by the ground control station operator. The 433MHz band is a legal frequency band to use in South Africa, which is where the flight tests are performed. Large antennas were placed on both the UAV and the ground control station / moving platform telemetry modules to increase the signal coverage. The telemetry module is connected to the Pixhawk 4 via a MAVLink connection.

A custom shelf was designed and 3D printed to mount the avionics in the airframe. The shelf was designed to compact, strong, stable and easily removable. The fully assembled avionics shelf is shown in Figure 3.5.

Figure 3.5: 3D printed shelf containing avionics.

A custom airspeed sensor mount was 3D printed to attach the pitot tube onto the UAV wing. A platform was also 3D printed to mount the RTK GPS module and magnetometer in the airframe. An anti-vibration pad was created using rubber, silicone and polystyrene which was used to mount the Pixhawk to the avionics shelf. This pad would absorb the vibrations from the motor and propeller to prevent them from contaminating the Pixhawk sensor measurements.

3.1.1.2 Avionics Software

Open source autopilot software is ideal for developing custom autopilot systems as they have community support and are often tested in various scenarios which increases their reliability. New features are also continuously added which improves their functionality. There are many open source autopilot software packages to consider and these include: PX4, ArduPilot, LibrePilot and Betaflight [47].

LibrePilot is mainly used for multi-rotor drones while Betaflight is more of a hobbyist software and does not facilitate more complex autopilot system development. PX4 and ArduPilot are the two main options that are considered as they both have the proper tools required for the implementation of the flight and guidance control systems. The Pixhawk 4 runs the NuttX Real Time Operating System (RTOS) and both the PX4 and ArduPilot software stacks can be compiled and built to run on NuttX, which further cements their usage.

(a) Logo of PX4 Autopilot

(b) Logo of ROS

Figure 3.6: Logos of the avionics software

PX4 Autopilot was chosen for the development of the new avionics software as it has two key advantages over ArduPilot. These advantages are that it supports hardware-in-the-loop (HITL) simulation and that it has an asynchronous software architecture which automatically deals with time scheduling [23]. The PX4 software was also successfully used in several previous ESL research projects (including: [23], [48], [49]) and this further provides confidence in using PX4. None of the previous projects used the PX4 software to control a fixed-wing UAV, since they were all focused on multi-rotor UAVs. This research project will therefore be the first in the ESL to develop the flight control software for a fixed-wing UAV using the PX4 open source software stack.

Because the MPC algorithm will be executed on the Jetson Nano, a software library is required to facilitate communication between the Jetson Nano and the Pixhawk. Robot Operating System (ROS) is such a library and is supported by PX4. PX4 supports both ROS1 and ROS2, with the latter being recommended. However, at the time that the MPC algorithm was implemented, only the ROS1 method was well documented and it was therefore used. The ROS1 method consists of using MAVROS with MAVLink to transmit data between the Jetson Nano and the Pixhawk. MAVLink is a messaging protocol for connecting isolated components [50] while MAVROS is a communication driver for connecting MAVLink to ROS [51].

3.1.2 Moving Platform

Previous research projects in the ESL that developed systems for landing on a moving platform used a full-size trailer with a large platform towed by full size motor vehicle. However, these previous projects focussed on landing quadrotor UAVs or helicopter UAVs that are able to travel at slow speeds or hover, and do not require an arrestor system after touchdown. Since this research project focusses on landing a fixed-wing UAV, it was decided to use a radio-control (RC) car as the moving platform and to perform a touch-and-go landing on a virtual platform located a few meters above the physical car. This solution avoids the use of the arrestor system to slow the UAV down which mitigates the high possibility of the UAV crashing on touchdown. The UAV's propeller would most likely get caught by the arresting wires causing damage to it. For this research project, it was decided to focus on the performance of the developed control system in performing the moving platform landing rather than the development of a platform to capture the

fixed-wing UAV. Another reason the RC car was chosen was its small size, as the airfield where the practical test would be performed only had a small runway. The trailer was too large to be used on the airfield runway. The RC car is limited to travel at a speed of 3 m/s to ensure that it does not overshoot the runway during the practical moving platform landing tests, as will be explained in Section 3.3.1.

The moving platform for this project therefore consists of a radio-controlled car carrying a mobile ground station, as shown in 3.7.

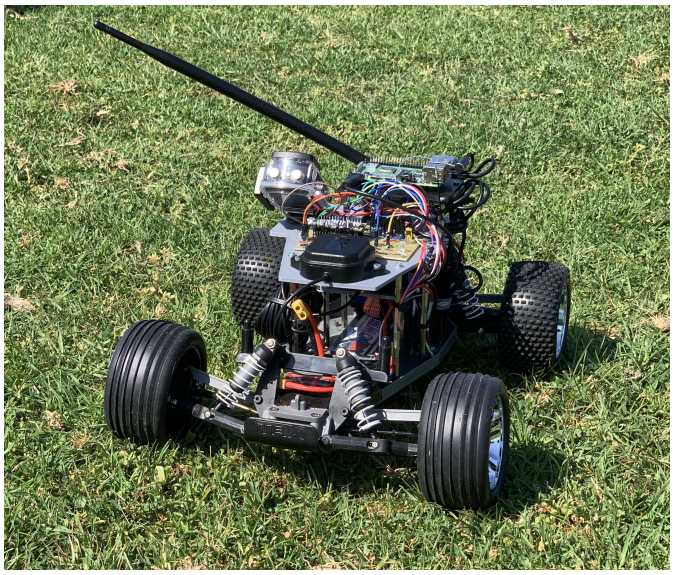


Figure 3.7: The fully assembled moving platform used for the practical flight tests.

The RC car consists of the car chassis, the platform electronics, a servo that actuates the steering angle, and a motor with an electronic speed controller to control the forward speed. The car chassis is a Traxxas Rustler XL-5. The servo, ESC and brushed motor that came with the car chassis are kept the same for the platform. The plan for the flight tests was to have the platform's motion be moving at a constant velocity in a straight line. A control system for the RC car was outside the scope of the project and was not required for the simple motion desired. As the RC car motor is brushed, applying a constant voltage on it would cause the car to move approximately at a constant speed, especially since the car moves on a flat surface. To obtain this constant voltage the ESC has to be supplied with a constant PWM signal. The value of this PWM signal was experimentally derived from trial and error. The speed that the platform is aimed to travel at is 3 m/s. The platform should approximately meet this speed and a high speed control accuracy is not required.

The moving platform electronics consists of two groups of hardware that operate independently of each other, namely the Platform Control electronics and the Mobile Ground Station electronics. An overview of the moving platform electronic groups with their associated communication protocols is shown in Figure 3.8.

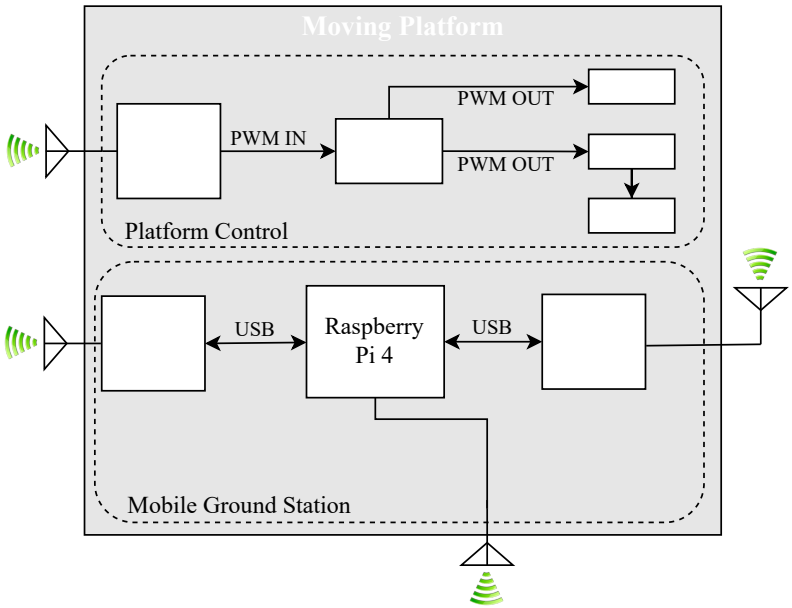


Figure 3.8: Block diagram of the moving platform electronics and their associated communication protocols.

The function of the Platform Control electronics, as the name suggests, is to control the motion of the platform by using the following hardware:

- a TBS Crossfire Nano receiver to receive the radio-control commands from the RC transmitter operated by the human driving the moving platform,
- an STM32G431KB Nucleo-32 microcontroller development board to transmit the commands from the RC receiver to the servo and the ESC. The STM32 also controls the mode of the platform,
- an MCP602 operational amplifier to act as a buffer between the STM32 and the Traxxas servo and ESC.

The STM32 microcontroller, shown in Figure 3.9, was chosen as it has a small footprint, allowing it to easily fit on the RC car. The STM32 was also available to be used in the ESL. The Traxxas ESC and servo operate at 100Hz which is not the norm as standard RC ESCs and servos operate at 50Hz. The RC receiver outputs its PMW signals at 50Hz, therefore the microcontroller must convert them to 100 Hz PWM signals to be compliant with the Traxxas components.

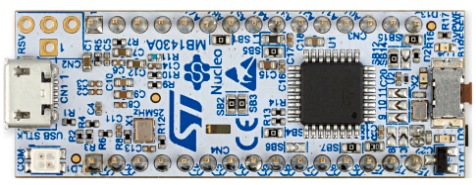


Figure 3.9: Image of the STM32G431KB Nucleo-32 microcontroller [52].

The TBS Crossfire receiver is excessive for the RC car as the car will be fairly close to the driver. A spare Crossfire receiver and transmitter were available in the ESL and therefore to save on cost and time they were adapted to be used on the RC car. The UAV and RC car Crossfire receivers are linked to their respective transmitters and therefore there is no interference between them. The datasheets and schematics of the Traxxas ESC and servo were not available, and therefore an operational amplifier was added as a buffer between the STM32 and the Traxxas components, to protect the STM32 from possible negative voltages.

The Mobile Ground Station electronics contain all the hardware required to run the ground station on the moving platform. The ground station hardware is placed on the moving platform because the accurate relative position between the UAV and the moving platform is required for the landing. The GPS receivers must be operated in RTK GPS mode, where the base GPS receiver on the moving platform transmits information about the GPS carrier signal it receives to the rover unit GPS receiver on the UAV. The mobile ground control station must therefore be located on the moving platform, so that it can receive the RTK packets from the base station DGPS module and transmit them to the rover unit DGPS module on the UAV via the telemetry module. The RTK GPS configuration is discussed in more detail in Appendix B.2.1. The hardware for the Mobile Ground Station electronics consists of:

- a Raspberry Pi 4 single board computer to run the Ground Control Station (GCSN) software,
- a Drotek 433 MHz telemetry module and antenna for communicating with the UAV.
- a Drotek DP0601 RTK DGPS module and antenna.

Figure 3.10: Image of the Raspberry Pi 4 [53].

The Raspberry Pi 4, shown in Figure 3.10, was chosen as it supports the GCSN software while still being small enough to fit on the RC car. The Raspberry Pi 4 contains a WiFi module that allows it to connect to the router of the stationary ground station. The telemetry module receives the data from the UAV to display in the GCSN software. The telemetry also sends the GCSN operator's commands to the UAV. The RTK GPS module

on the RC car is the base module and its data is sent to the UAV via the telemetry so that the rover module can obtain the UAV's position to the centimeter level. Both the telemetry and RTK GPS module are connected to the Raspberry Pi 4 via USB.

The RC car also contained some additional components that were required for it to function. These components will now be mentioned. A 3D-printed platform was created to mount all the electronics on the RC car. The moving platform's electronics, ESC and servo were powered by one 2S 5300 mAh LiPo battery. This battery's capacity was sufficient to last a whole flight test day without being replaced. A 5V 5A voltage regulator was used to convert the LiPo voltage to a safe level to power the Raspberry Pi 4, STM32 and other minor electronic components. An Olfi One.Five action camera was mounted on the RC car to capture the UAV chasing the car and landing on the virtual platform.

3.1.3 Stationary Ground Station

The GCSN software needs to be accessed by the GCSN operator to view the UAV data and send commands. Since the RC car is small and mobile, the ground station hardware cannot be physically accessed by the GCSN operator therefore the GCSN software needs to be accessed remotely. The remote connection was achieved by using WiFi to remote desktop from a laptop into the Raspberry Pi 4. A block diagram of the stationary ground control station hardware and the associated communication protocols is shown in Figure 3.11.

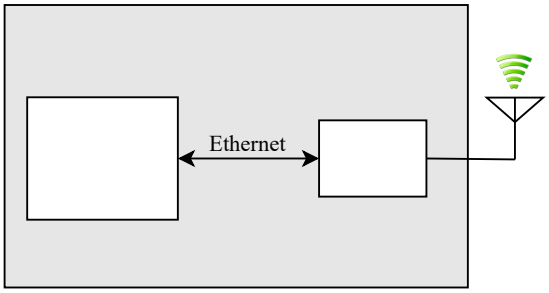


Figure 3.11: Block diagram of the stationary ground station hardware and their associated communication protocols.

The hardware used for the stationary ground station consists of:

- a laptop to allow the GCSN operator to interface with the GCSN software,
- a WiFi router to create a wireless local area network for the laptop and Raspberry Pi 4.

The laptop connects to the router with an Ethernet cable and uses remote desktop software to wirelessly access the Raspberry Pi's desktop and hence the GCSN software. The GCSN software used is QGroundControl (QGC) as it is the recommended software to use by the PX4 developers. QGC is mainly used to calibrate the UAV sensors as well as monitor the

aircraft states and sensors during a flight session. Figure 3.12 shows the user interface (UI) of QGC as the aircraft is performing a flight session.

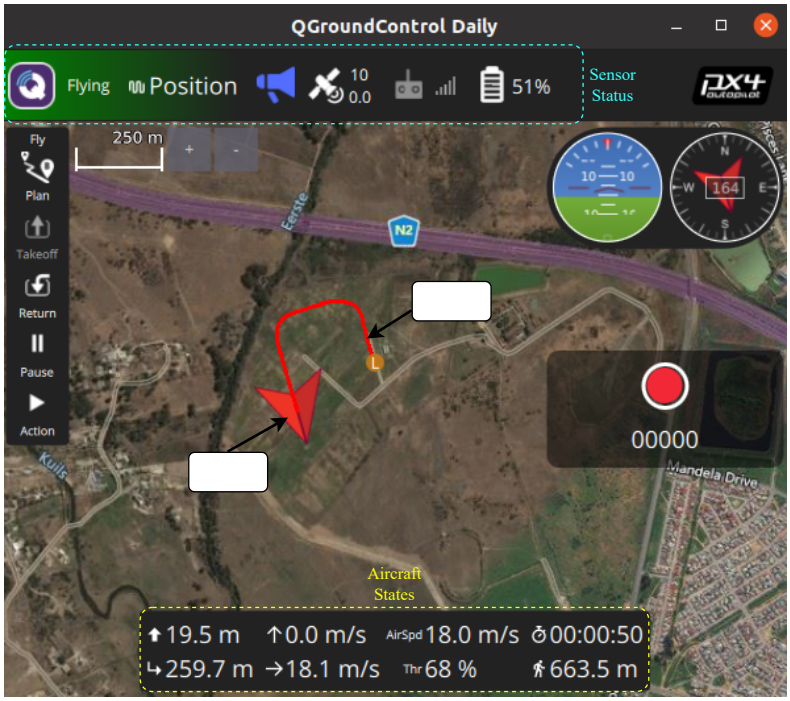


Figure 3.12: User interface of QGroundControl

The QGC UI shows the current aircraft location as well as the flight path the aircraft has flown. QGC displays warnings from issues that might occur during the flight test and this is used as an indicator to determine if the pilot should perform an emergency landing. QGC also contains a MAVLink console that is connected to the Pixhawk. This console is used to change the operating sub-mode of the control system for testing the different controllers or automated landings.

Before the moving platform landing is performed, the UAV's individual controllers are tested and a runway landing with the UAV is executed. The physical setup used to practically test the individual controllers and perform the runway landing is similar to the moving platform landing setup, with the exception that it excludes the moving platform. The setup for flight controller testing and runway landings runs the ground control station software (QGC) directly on the laptop with the telemetry and DGPS base modules connected to it via USB. The laptop will directly connect to the UAV via the telemetry module and the GCSN operator will have physical access to the ground control station. The RTK DGPS base module antenna is mounted on a tripod to give it an open view of the sky so that it can capture multiple satellites.

3.1.4 UAV Safety Pilot and Moving Platform Driver

The UAV pilot has an RC transmitter which links to the RC receiver on the UAV. The RC transmitter is a RadioMaster TX16S equipped with a TBS Crossfire Micro TX V2 transmitter module. The UAV pilot uses the transmitter to switch the modes of the UAV and control the UAV when in Manual mode. The UAV has two modes of operation which are Manual mode and Autopilot mode. In Manual mode, the pilot has full control of the UAV and it operates as a standard RC model aircraft. In Autopilot mode, the control system controls the UAV and the pilot does not have control. At any time the pilot can retake control of the aircraft by switching back to Manual mode. There are sub-modes in the control system depending on what controller is being tested or what landing scenario is being executed. The sub-modes become active when the UAV is in Autopilot mode. The pilot can also arm or disarm the UAV on the ground to make it safe to handle.

The Moving Platform driver also has an RC transmitter which links with the RC receiver on the moving platform. This RC transmitter is a TBS Tango 2 which has a built-in transmitter module to connect to the receiver. The Moving Platform has two modes of operation which are Manual mode and Cruise Control Mode. In Manual mode the driver has full control of the platform and it behaves as a standard RC car. In Cruise Control mode the driver can only steer the platform as the throttle is set to a fixed PWM value to move the platform at a constant speed. The driver can arm or disarm the platform. When the platform is disarmed, the ESC applies a breaking effect on the motor allowing the platform to be quickly brought to a halt.

3.2. Simulation Software

The PX4 Autopilot software supports software-in-the-loop (SITL) simulations and this will be exploited to test the developed control systems in a more realistic environment. To perform SITL simulations, a physics simulator is required to produce the sensor measurements that would be received by the PX4 flight control software, and to simulate the response of the aircraft to the commanded throttle settings and control surface deflections. Although PX4 supports numerous simulators, only three of them can be used with a fixed-wing UAV, namely Gazebo, FlightGear, and JSBSim.

Figure 3.13: Logo of Gazebo Simulator.

Gazebo is a light-weight simulator that is used for various models such as robots,

rovers, multi-rotor UAVs, and fixed-wing UAVs. Gazebo therefore has a large community which can be consulted when troubleshooting is required. Gazebo also uses an architecture similar to ROS which simplifies development. FlightGear and JSBSim are more graphically intensive and have a steeper learning curve then Gazebo. They also do not have the same amount of support as Gazebo. Gazebo was therefore chosen as the simulator for the SITL implementation.

3.3. Landing Strategy

This section first discusses the moving platform specifications, then the landing scenarios and finally, the landing procedure.

3.3.1 Moving Platform Specifications

The moving platform needs its size and speed to be determined to allow it to imitate an aircraft carrier in terms of the model aircraft scale. Le Roux [14] compared a real aircraft and aircraft carrier to the model aircraft and platform to derive these values. Le Roux's method consists of determining scale factors based on the ratio of the aircraft carrier runway dimensions and the wingspan of the landing real aircraft. These scale factors would then be multiplied by the wingspan of Le Roux's fixed-wing UAV to derive the ideal platform's dimensions. Le Roux found that the ideal platform should have a length of 3.24m and a width of 0.77m. Unfortunately, Le Roux only had a 3 m by 3 m platform to do the test and therefore that became his platform size.

Le Roux's method is deemed acceptable as scaling the aircraft carrier's runway down to the size of the fixed-wing UAV is the best approach usable with the resources available. This project uses the same size airframe as Le Roux, therefore the same ideal platform dimensions can be used. It is planned for a future ESL project to physically land the fixed-wing UAV onto the 3 m by 3 m trailer. Therefore similar to Le Roux, 3 m by 3 m is chosen to be the size of the moving platform. The moving platform, however is represented by the RC car, which has a fixed dimension. Therefore the 3 m by 3 m size is set to the virtual platform located directly above the moving platform. The virtual platform size results in both the maximum allowable longitudinal and lateral errors for aircraft on touchdown being ± 1.5 m.

Le Roux chose a platform speed of 10 m/s due to his GPS providing noisy measurements at low speeds. This speed however is too high for the RC car to maintain. This speed is also too high for the length of the runway where the practical flight tests are performed, as the platform would overshoot the runway before the aircraft would land on it.

A US Navy Nimitz-Class aircraft carrier can cruise at 56 km/h [54] while an aircraft can land on the carrier at 241 km/h [26]. The fixed-wing UAV is set to land on the virtual

platform at 18 m/s. The moving platform speed can therefore be calculated by multiplying the UAV speed by a scale factor formed by the ratio of the carrier speed over the aircraft speed. The moving platform speed is therefore,

$$\bar{V}_{vp} = 18\,\frac{56}{241} = 4.18 \text{ m/s} \tag{3.1}$$

The length of the runway at the airfield is 150 m long. However, to have a margin of safety and a runoff area for the UAV and the moving platform, the runway is instead assumed to be 120 m long. When the moving platform moves at 4.18 m/s, it uses a distance of 128.55 m before the UAV touches down on the virtual platform. This distance is larger than the assumed runway length, and therefore the speed of the platform has to be reduced. The platform's speed is therefore set to 3 m/s, which is mathematically represented as,

$$\bar{V}_{vp} = 3 \text{ m/s} \tag{3.2}$$

At this speed, the platform covers a distance of 85 m before touchdown, which is well within the assumed runway length. 3 m/s is also an easily attainable speed for the RC car.

3.3.2 Landing Scenarios

There are two main landing scenarios that the aircraft should be able to execute to successfully achieve the aim of this project. The fixed-wing UAV should be able to accurately land on both a stationary runway and a moving platform. For the moving platform landing, it is assumed that the platform is performing a horizontal translational motion at a constant velocity (i.e. not changing its direction), and is not performing any rotational motion.

3.3.2.1 Stationary Runway Landing

This scenario represents a landing where the aircraft is aiming to land accurately within a designated bounding box demarcated on the runway. The runway landing essentially represents a stationary platform landing, therefore, it is vital that this landing be accurate. It is important for the aircraft to able to successfully execute this scenario as the moving platform landing will be even more challenging. This scenario will allow the flight and guidance control systems to be verified to ensure that they work correctly. This landing scenario will be performed first on the practical vehicle before the moving platform test is performed.

3.3.2.2 Moving Platform Landing

This landing scenario represents a landing on a moving platform which is translating horizontally at a constant velocity. The translational motion consists of linear horizontal movements which are the most dominant motion of the moving platform. The moving platform does not change its altitude and attitude during this motion as it only changes its position in the horizontal plane. The moving platform's translational motion will be constrained so that it generally moves at a constant speed in a straight line with slight deviations due to environmental factors. This is to simulate an aircraft carrier which generally moves in one direction for landing procedures. For the moving platform flight tests, the platform will move along the runway in the same direction as the aircraft would land for a normal runway landing. The aircraft lands at a much higher speed than what the platform moves and this allows the aircraft to catch up to the platform and land on it. Since the practical moving platform vehicle is an RC car that is quite agile, the aircraft must be able to compensate for any longitudinal or lateral offsets that may occur due to the RC car deviating from its intended trajectory. The practical moving platform landing test is the final test that will be performed after the flight and guidance controllers have been tested and verified to behave correctly.

There are various other motions that the moving platform can exhibit such as heaving and rotational motion. However, these motions are considered to be outside the scope of this project.

3.3.3 Landing Procedure

The landing procedure for the two landing scenarios are generally the same with only minor differences existing between them which will be highlighted. First, the landing procedure for the stationary runway landing will be introduced, and thereafter the moving platform landing procedure will be discussed.

3.3.3.1 Stationary Runway Landing Procedure

The aircraft needs to first align itself with the runway from its current position before it can land. The aircraft can do this by following a standard circuit around the airfield. This circuit is inspired by the actual circuit flown by real, full-sized aircraft used to perform a runway landing. Figure 3.14 shows the circuit flown for a runway landing.

The circuit is flown in an anti-clockwise direction due to physical constraints of the airfield that was used for the flight testing. The entry point to the circuit can be at any position and not just the location shown in the figure. The crosswind and downwind legs are used to get the aircraft on the circuit. The base leg is the last perpendicular leg before getting in line with the runway. It is required that the aircraft be at the appropriate altitude and airspeed on this leg so that the aircraft can descend at a constant rate for

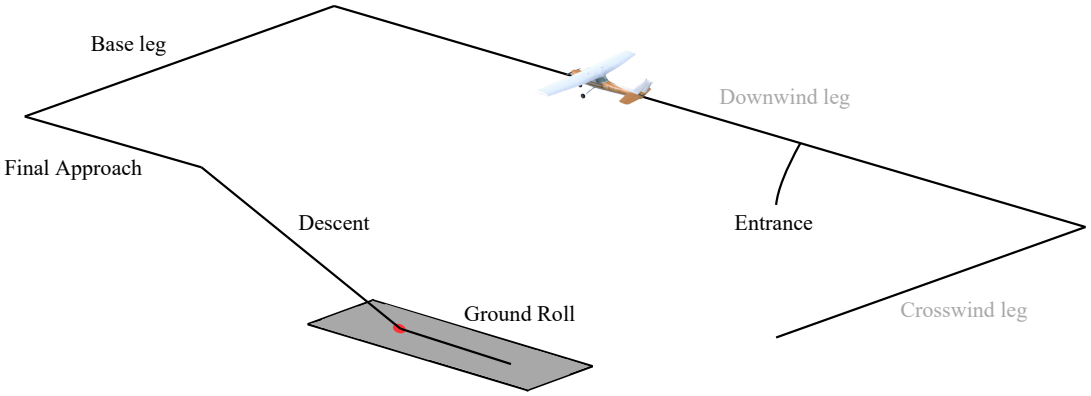


Figure 3.14: Diagram showing the circuit followed for a stationary runway landing. Adapted from Le Roux [14].

landing. On the final approach, the aircraft should be aligned with the runway and start slowing down so that it can safely land. The landing airspeed should be near the stall speed but not low enough to stall the aircraft. The aircraft then descends and touches down at the start of the runway. After touchdown, the aircraft enters the ground roll state where it slows down on the runway while maintaining the centre line until it comes to a complete stop. If the aircraft is not stabilised for landing, then it can abort the landing attempt by returning to the circuit and go around to try again. For the runway landing scenario, it is expected that the aircraft should be able to touch down accurately on a predetermined point on the runway. To do this a glide slope needs to be used which would allow the aircraft to regulate its airspeed and pitch angle when descending. The glide slope is the altitude ramp reference shown in Figure 3.15 which guides the aircraft to the touchdown point. The glide slope is defined by the glide slope angle γ and this angle has to be selected to ensure that the aircraft can track the glide slope while maintaining its airspeed. The glide slope angle is selected to be 4° as this value was chosen by De Bruin [16] to minimise landing inaccuracies from glide slope tracking offsets.

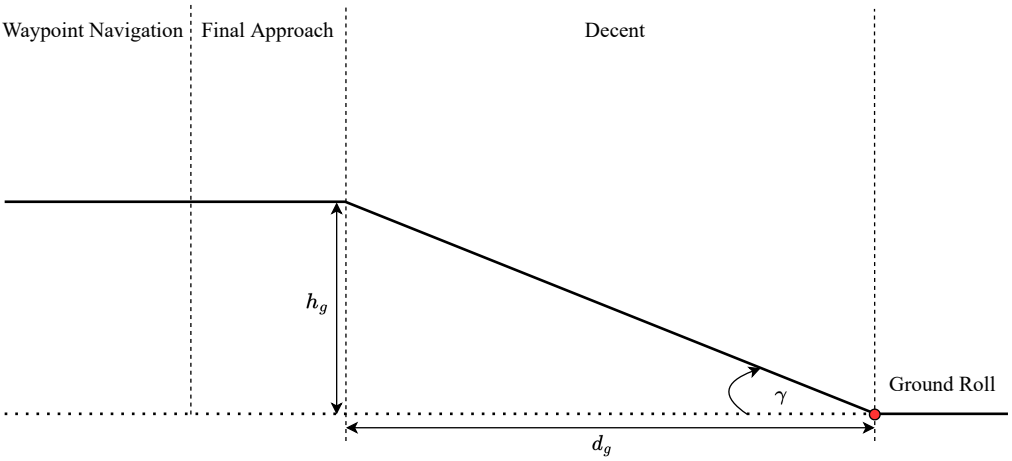


Figure 3.15: Landing profile for a stationary runway landing.

The glide slope ground distance d_g is selected as 250 m and was chosen based on the

distance available at the airfield. The required altitude at the start of the descent h_g is then calculated as,

$$h_g = d_g \, tan(\gamma) = 17.4817 \, \text{m} \tag{3.3}$$

The aircraft should be at this altitude by the latest in the base leg, so that it can focus on preparing for landing during the final approach. The aircraft is flown around the circuit at an airspeed of 18 m/s however, this airspeed is too high to land with. De Bruin experienced landing gear flexing on touchdown at 16 m/s, therefore it is safe to assume that at 18 m/s the aircraft will be damaged. The aircraft is therefore slowed down during the final approach to 16 m/s which is significantly above the stall speed. During the descent phase the aircraft executes the de-crab manoeuvre to align itself with the runway before touchdown. This manoeuvre needs to be executed to ensure that the lateral forces acting on the aircraft are minimal. A large lateral force can damage the landing gear and also cause the aircraft to tip over causing a wing strike. The de-crab manoeuvre will cause a slight error in lateral position however this is acceptable to ensure the safety of the aircraft.

3.3.3.2 Moving Platform Landing Procedure

The circuit that the aircraft flies around the airfield for the moving platform landing is similar to the one used for the stationary runway landing. The circuit differs slightly for the final approach and descent phases as shown in Figure 3.16.

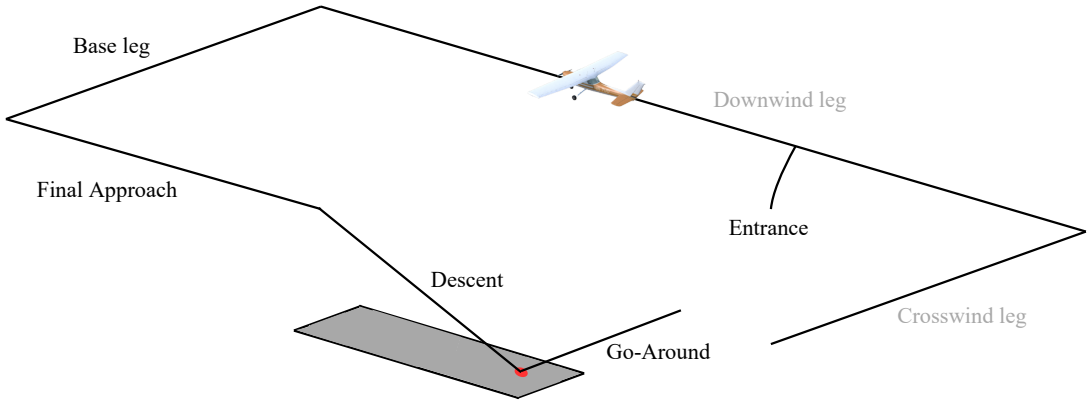


Figure 3.16: Diagram showing the circuit followed for a moving platform landing.

The touchdown point for the moving platform landing is further along the runway compared to the stationary runway landing as the aircraft is not required to come to a complete stop on the runway. Instead the aircraft will simply perform a go-around manoeuvre when it detects that it has landed on the virtual platform and will rejoin the circuit. Figure 3.17 shows the landing profile followed for the moving platform landing. This profile is similar to the runway landing in that a glide slope is used to land on the touchdown point. The glide slope angle for this profile is also set to 4°.

The touchdown point between the aircraft and moving platform is predicted by the

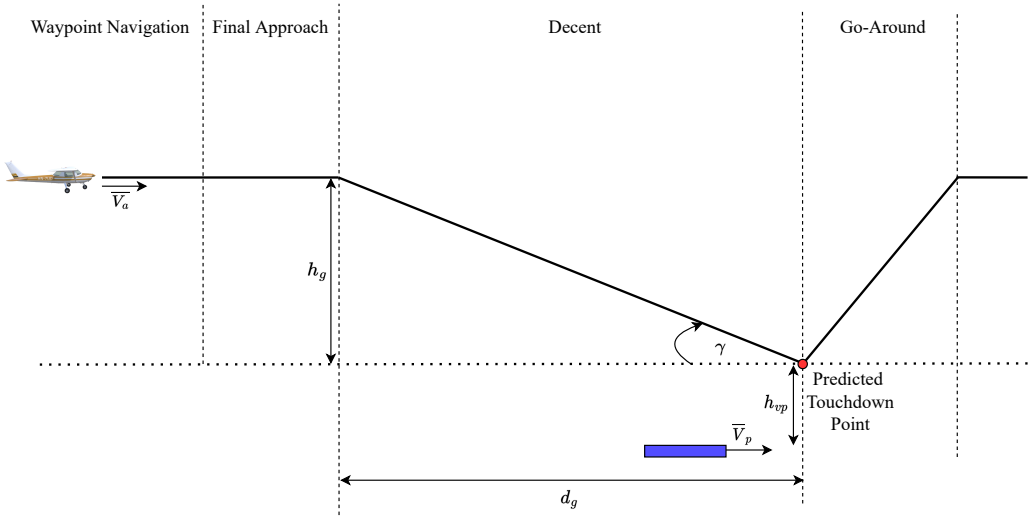


Figure 3.17: Landing profile for a moving platform landing.

guidance control system, and this point is then used to create the glide slope reference for the aircraft to follow. The predicted touchdown is continuously updated by the guidance control system based on the instantaneous speed of the moving platform and aircraft. In response, the glide slope reference for the aircraft to follow is also continuously updated. The glide slope ground distance d_g and altitude h_g for the moving platform landing profile is the same values as the stationary runway landing. The RC car that is used as the moving platform is too small for the UAV to physically land on, therefore it is decided to instead use a virtual platform that the aircraft can target for a landing. The virtual platform is a few meters above the RC car and a "touchdown" on this platform occurs when the aircraft's altitude is less than the virtual platform altitude. The virtual platform's altitude above the RC car is given by h_{vp} which is set to 3 m for this project. This value was chosen as it allows the aircraft to get quite close to the RC car while still being able to perform the go-around manoeuvre. The moving platform landing circuit's altitude is the sum of of the glide slope (h_g), virtual platform (h_{vp}) and RC car altitudes. As the moving platform moves in the same direction as the aircraft, the relative speed between them is less than the aircraft's ground speed. The aircraft's airspeed for landing is therefore kept at 18 m/s as it does not actually touch the platform on landing, which mitigates the risk of damaging the aircraft.

3.4. Summary

This chapter first presented the physical system used to perform the moving platform landing. The simulation software used to test the control system in software-in-the-loop simulations was then selected. Finally, the landing strategies that were followed, both in simulation and practical tests, were discussed. Now that a better understanding of the systems used and strategies followed has been obtained, the fixed-wing UAV model will now be presented in the next chapter.

Chapter 4

Aircraft Dynamic Model

This chapter presents the mathematical model that was used to describes the flight mechanics of the fixed-wing UAV. First, the nonlinear model of the aircraft dynamics is presented. Next, a linear, decoupled model of the aircraft dynamics is obtained by linearising the nonlinear aircraft dynamics about an equilibrium state. The linear aircraft model is used for the flight control system design, and the nonlinear aircraft model is used for simulations. Finally, the natural modes of motion of both the longitudinal and lateral dynamics are calculated and analysed for the specific UAV used in this project.

The background theory in this chapter was sourced from the unpublished Advanced Automation 833 course notes that were compiled by Peddle and Engelbrecht. (These course notes were compiled from several sources, including Cook [55], Etkin and Reid [56], and Blakelock [57].) The mathematical model uses standard reference frames, equations of motion, and force and moment models that are commonly used for modelling and control of a fixed-wing aircraft.

4.1. Reference Frames

This section presents all the reference frames that are used in this research project.

4.1.1 Inertial Frame

Newton's equations of motion can be used to model a UAV's behaviour, however, this requires an inertial frame or inertial axis system to be defined. For UAVs that have a small range, the North-East-Down (NED) axis system is often used to represent the inertial frame. This system assumes that the Earth is flat and non-rotating which is sufficient for this project as the UAV will only be operated within a few hundred meters of the runway. The origin of the inertial axis can be placed anywhere in the environment however, for this project it was chosen to be start of the runway. As shown in Figure 4.1, the X-axis points in the true North direction, the Y-axis points in the East direction, and the Z-axis points downwards, normal to the local horizontal plane. The direction of the Z-axis is downwards to ensure that the right-hand orthogonal rule is satisfied. The runway's heading is offset by an angle Ψ_r from true North. Using this axis system allows the UAV's states, such as

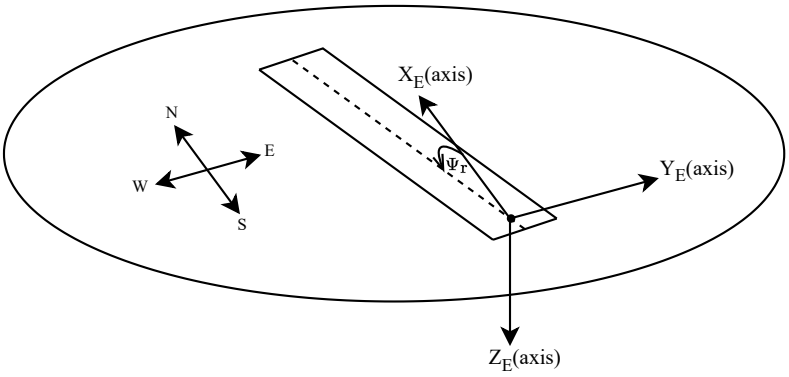


Figure 4.1: Illustration of the inertial axis system.

position and velocity, to be referenced with respect to the ground. This is required when wanting to place the UAV at a certain location, which itself is referenced with respect to the ground, as will be necessary for the moving platform landing.

4.1.2 Body Frame and Aircraft Notation

The body frame, also known as the body axis system, is an axis system that moves and rotates with the UAV. The origin of this frame is chosen to be positioned at the UAV's centre of mass. A fixed-wing UAV is bilaterally symmetrical, therefore, it has a plane of symmetry that divides it evenly. The body frame's X-axis (X_B) lies in this plane and points towards the front (nose) of the plane as shown in Figure 4.2. The Y-axis (Y_B) is perpendicular to the X-axis and points outward through the right wing of the aircraft. The Z-axis (Z_B) is also in the plane of symmetry and points towards the landing the gear to satisfy the right-hand orthogonal rule. The variables shown in figure 4.2 use the standard notation for aircraft modeling. Table 4.1 explains the meanings of the variables.

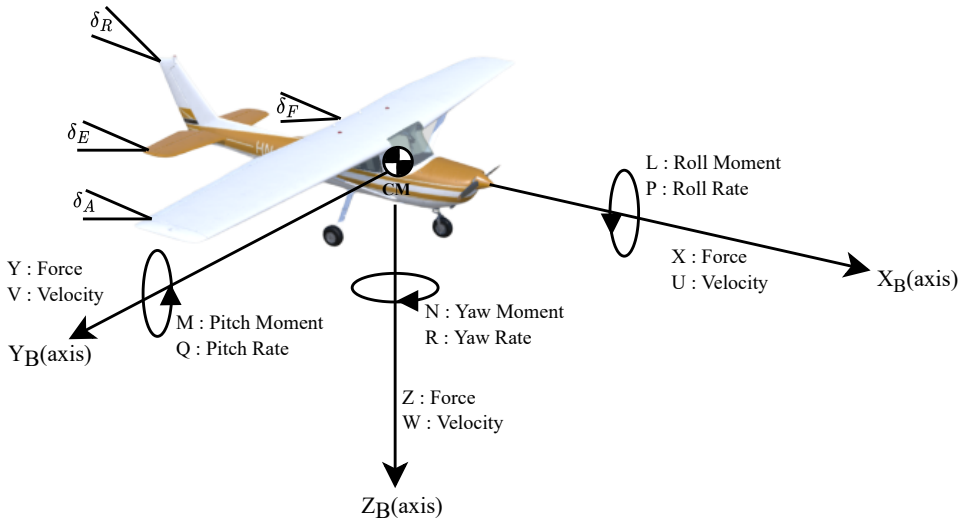


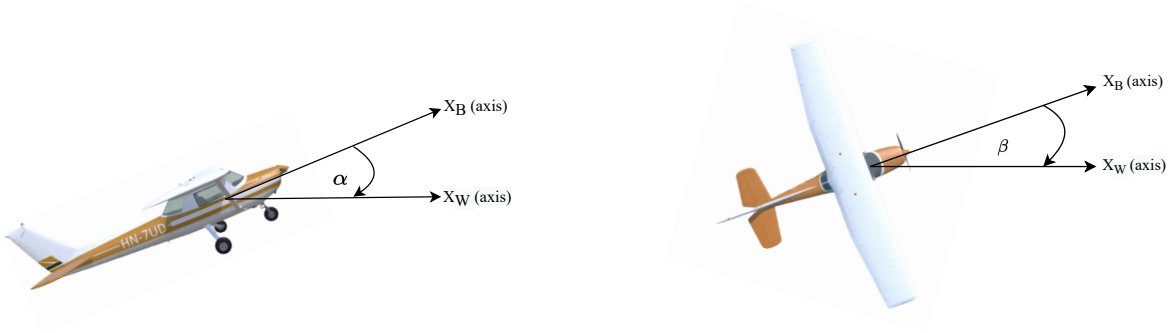
Figure 4.2: Body frame illustration with all the variables representing the forces, moments, velocities and angular rates. [adapted from AA833 course notes], [Aircraft model adapted from [58]]

Table 4.1: Notation used for aircraft model

Aircraft Notation	
Variable	**Definition**
X, Y, Z	Force vector coordinates defined in the body frame. The forces are the axial, lateral, and normal forces respectively.
L, M, N	Moment vector coordinates defined in the body frame. The moments are roll, pitch, and yaw respectively.
U, V, W	Linear velocity vector coordinates defined in the body frame. The velocities are the axial, lateral, and normal velocities respectively.
P, Q, R	Angular velocity vector coordinates defined in the body frame. The angular velocities are roll, pitch, and yaw rates respectively.
$\delta_A, \delta_E, \delta_F, \delta_R$	Aileron, elevator, flap and rudder deflection angles. The positive direction for the deflection angles is defined to produce a negative moment on the aircraft in the body axis. The positive deflection angle direction also moves counter clockwise(curl right hand rule) to the axis in which it rotates.

4.1.3 Wind Frame

The wind frame, also known as the stability frame or wind axis system, is similar to the body frame as it also moves with the aircraft and its origin also coincides with the aircraft's centre of mass. The wind frame's X-axis points in the direction of the aircraft's velocity vector. Its Y-axis points out through the right (starboard) wing and the Z-axis, which completes the right hand orthogonal rule, points downwards towards the landing gear. The body and wind frames are related to each other by two variables which are the angle of attack (α) and the angle of sideslip (β), as shown in Figure 4.3.



(a) Wind axis relation by α **(b)** Wind axis relation by β

Figure 4.3: Diagrams showing the body frame to wind frame rotation. Adapted from De Bruin [16].

Equations 4.1 and 4.2 show the conversion from the body axes to the wind axes using the rotation matrices comprised of the angle of attack (R_α) and angle of sideslip (R_β). The vector X_b is in the body frame and the vector X_w is in the wind frame.

$$X_w = \begin{bmatrix} \cos(\beta) & \sin(\beta) & 0 \\ -\sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} X_b \tag{4.1}$$

$$X_w = R_\beta R_\alpha X_b \tag{4.2}$$

It is convenient to have the velocity in spherical/polar form because it allows for easier modeling of the aircraft, as will be shown later in this chapter. The velocity spherical form is shown in Figure 4.4.

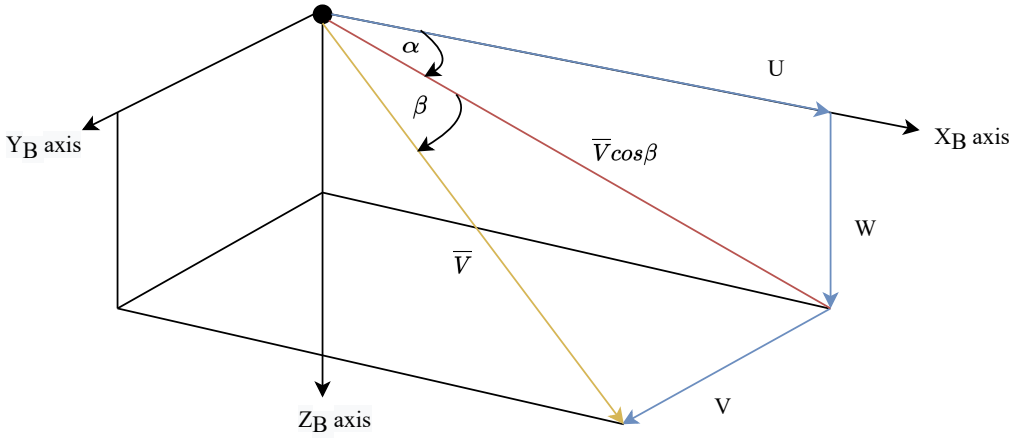


Figure 4.4: Spherical form of the aircraft velocity coordinates [adapted from AA833 course notes].

In this form, the velocity is expressed in terms of its magnitude $\bar{V}$ and two angles which are the angle of attack α and the angle of sideslip β. The equations below relate the velocity vector in Cartesian form to spherical form.

$$\bar{V} = \sqrt{U^2 + V^2 + W^2} \tag{4.3}$$

$$\alpha = \tan^{-1}\left(\frac{W}{U}\right) \tag{4.4}$$

$$\beta = \sin^{-1}\left(\frac{V}{\bar{V}}\right) \tag{4.5}$$

The inverse of relation is given by

$$U = \bar{V}\cos(\alpha)\cos(\beta) \tag{4.6}$$

$$V = \bar{V}\sin(\beta) \tag{4.7}$$

$$W = \bar{V}\sin(\alpha)\cos(\beta) \tag{4.8}$$

4.1.4 Guidance Frame and Runway Frame

Two additional reference frames, namely the guidance frame and the runway frame, are also used in this project. The guidance frame will be defined in Section 6.1 when the guidance algorithm is introduced. The runway frame will be defined when the moving platform model is introduced in Section 4.5.

4.2. Equation of Motion Development

A dynamic model that encapsulates all the significant characteristics of the aircraft must first be derived before the flight control system can be designed. A six-degrees-of-freedom model is used to represent the aircraft, as it is assumed that the aircraft is a rigid body. The six-degrees-of-freedom include three translational and three rotational degrees of freedom in the axial, lateral, and normal axes. This model is commonly used for small UAVs as their mass and structure do not change during flight, as is the case for the UAV used in this project. The equations of motion can be split into two categories, namely the kinetics and kinematics.

4.2.1 Kinetics

The kinetic equations of motion relate the forces and moments that act on the aircraft to the translational and rotational motion of the aircraft.

The kinetic equations of motion are shown below, with all vectors coordinated in the body frame. The complete derivation can be found in [57].

$$X = m(\dot{U} - VR + WQ) \tag{4.9}$$
$$Y = m(\dot{V} + UR - WP) \tag{4.10}$$
$$Z = m(\dot{W} - UQ + VP) \tag{4.11}$$
$$L = \dot{P}I_{xx} + QR(I_{zz} - I_{yy}) \tag{4.12}$$
$$M = \dot{Q}I_{yy} + PR(I_{xx} - I_{zz}) \tag{4.13}$$
$$N = \dot{R}I_{zz} + PQ(I_{yy} - I_{xx}) \tag{4.14}$$

The first three equations relate the body forces (X, Y, Z) of the aircraft to its translational velocity (U, V and W). The last three equations relate the body moments (L, M, N) to the aircraft's angular rates (P, Q and R). The m variable in the first three equations is the mass of the aircraft, while I_{xx}, I_{yy}, and I_{zz} variables are the moment of inertia in the

body frame.
Two important assumptions are made in these equations to simplify the derivation:

- It is assumed that the aircraft is symmetrical about the XZ-plane, and hence, the cross product of I_{xy} and I_{yz} is zero. This is valid for this aircraft as was discussed in section 4.1.2.
- The cross product of the moment of inertia I_{xz} is small, which is true for conventional aircraft.

With the kinetic equations now defined, the kinematic equations are presented next.

4.2.2 Kinematics

The kinematic equations relate the motion variables to each other over time. These variables include linear velocity, angular velocity, linear position and attitude (angular position). The kinematics do not consider the forces and moments that cause the motion, and is also called the "geometry of motion". The linear and angular velocities were introduced in the kinetic equations, and now the position and attitude are added for the kinematic equations. The aircraft position and attitude are parameterised as follows,

- N, E, D are the position coordinates of the aircraft in the inertial frame. N is the north coordinate along the inertial X axis, E is the east coordinate along the inertial Y axis, and D is the down coordinate along the inertial Z axis.
- Φ, Θ, Ψ are the Euler 3-2-1 attitude coordinates roll, pitch, and yaw, of the body frame relative to the inertial frame.

The reason for using Euler 3-2-1 to represent the angular position is because it is fairly easy to visualise and work with. The disadvantage of using them is that they contain a singularity in pitch (Θ) at $\pm$ 90°. A solution to this problem is to use quaternions as they do not suffer from this issue, however, they are not as easy for humans to interpret. For this project, the fixed-wing UAV will not operate at or close to the 90° pitch angle. Therefore, it was decided to use the Euler angles to express attitude. The meaning of the Euler angles is explained below with a visual representation shown in Figure 4.5.

- Φ: Roll angle represents the rotation of the wings with respect to the horizon.
- Θ: Pitch angle represents the rotation of the aircraft's nose with respect to the horizon.
- Ψ: Heading angle represents the rotation of the aircraft with respect to true North.

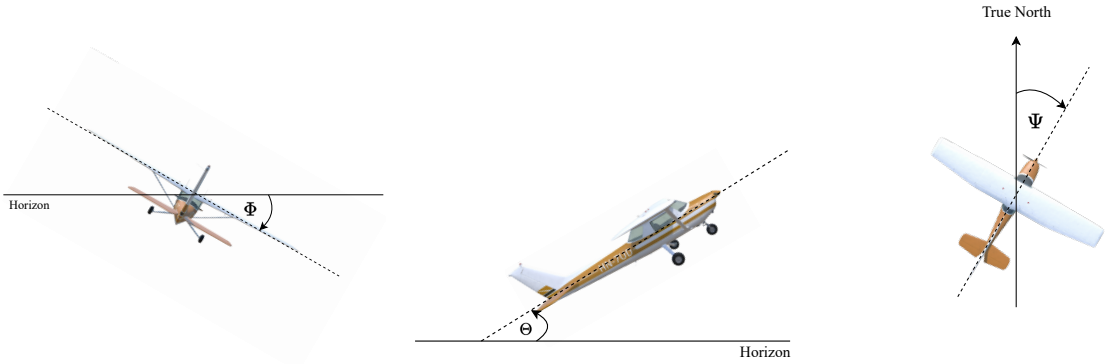


Figure 4.5: Euler angles visual representation [adapted from AA833 course notes]

Since the Euler angles (attitude) are now defined, the relationship between the time rate of change of the Euler angles and other kinematic states must now be considered. The relationship between the Euler angle rates and the body angular rates (P, Q, R) is a function of the current attitude. This relationship was derived by Ektin and Reid [56], and is given as,

$$\begin{bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\Phi)\tan(\Theta) & \cos(\Phi)\tan(\Theta) \\ 0 & \cos(\Phi) & -\sin(\Phi) \\ 0 & \sin(\Phi)\sec(\Theta) & \cos(\Phi)\sec(\Theta) \end{bmatrix} \begin{bmatrix} P \\ Q \\ R \end{bmatrix} \qquad |\Theta| \neq \frac{\pi}{2} \tag{4.15}$$

The formula is not valid when the pitch angle is $\pm 90°$ due to the singularity created at this point.

Now that we have the relationship for the rotational variables, the next step is to relate the translational variables. In the inertial axes, the following relationship exists between translational position and velocity,

$$\begin{bmatrix} \dot{N} \\ \dot{E} \\ \dot{D} \end{bmatrix} = \begin{bmatrix} V_N \\ V_E \\ V_D \end{bmatrix} \tag{4.16}$$

where V_N, V_E and V_D are the velocities in the north, east, and down axes respectively. The kinetic equations express the aircraft's translational velocity in the body frame, which means that they must be transformed to the inertial frame before they are integrated to obtain the position in the inertial frame. A rotation matrix formed from the current attitude of the aircraft is used to perform this transformation.

The Euler angles represent an ordered sequence of rotations through which the inertial frame must be rotated to align it with the body frame. For an Euler 3-2-1 sequence, the inertial frame is rotated by the yaw angle Ψ about its z-axis, then the first immediate frame is rotated by the pitch angle Θ about its y-axis, and finally the second intermediate

frame is rotated by the roll angle Φ about its x-axis. The transformation matrix that transforms a coordinate vector $\mathbf{V}_I$ in the inertial frame to the corresponding coordinate vector $\mathbf{V}_B$ in the body frame, is given by,

$$\mathbf{V}_B = \begin{bmatrix}\mathbf{R}_\Phi & \mathbf{R}_\Theta & \mathbf{R}_\Psi\end{bmatrix}\mathbf{V}_I \tag{4.17}$$

where $\mathbf{R}_\Psi$ is the rotation through yaw angle, $\mathbf{R}_\Theta$ is the rotation through pitch angle, $\mathbf{R}_\Phi$ is the rotation through roll angle, $\mathbf{V}_I$ is the vector in the inertial frame and $\mathbf{V}_B$ is the vector in the body frame. This formula can be expanded to,

$$\begin{bmatrix}x_B\\y_B\\z_B\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\0 & \cos(\Phi) & \sin(\Phi)\\0 & -\sin(\Phi) & \cos(\Phi)\end{bmatrix}\begin{bmatrix}\cos(\Theta) & 0 & -\sin(\Theta)\\0 & 1 & 0\\\sin(\Theta) & 0 & \cos(\Theta)\end{bmatrix}\begin{bmatrix}\cos(\Psi) & \sin(\Psi) & 0\\-\sin(\Psi) & \cos(\Psi) & 0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_I\\y_I\\z_I\end{bmatrix} \tag{4.18}$$

where x_B, y_B and z_B are the coordinates in the body frame and x_I, y_I and z_I are the corresponding coordinates in the inertial frame. This formula can then be simplified to,

$$\begin{bmatrix}x_B\\y_B\\z_B\end{bmatrix} = \begin{bmatrix}\cos\Psi\cos\Theta & \sin\Psi\cos\Theta & -\sin\Theta\\\cos\Psi\sin\Theta\sin\Phi - \sin\Psi\cos\Phi & \sin\Psi\sin\Theta\sin\Phi + \cos\Psi\cos\Phi & \cos\Theta\sin\Phi\\\cos\Psi\sin\Theta\cos\Phi + \sin\Psi\sin\Phi & \sin\Psi\sin\Theta\cos\Phi - \cos\Psi\sin\Phi & \cos\Theta\cos\Phi\end{bmatrix}\begin{bmatrix}x_I\\y_I\\z_I\end{bmatrix} \tag{4.19}$$

and then written in compact form as,

$$\mathbf{V}_B = (\mathbf{DCM}_{I\to B})\mathbf{V}_I \tag{4.20}$$

The DCM is the direction cosine matrix which transforms a vector from one axis system to another. In this case, it transforms a vector from the inertial frame to the body frame. To transform a vector from the body frame to the inertial frame the inverse of this transform can be taken and since the DCM is orthogonal its inverse is its transpose. This is shown by,

$$\begin{aligned}\mathbf{DCM}_{B\to I} &= (\mathbf{DCM}_{I\to B})^{-1}\\ &= (\mathbf{DCM}_{I\to B})^{\intercal}\end{aligned} \tag{4.21}$$

For the translational dynamics, it is required to convert the body velocity coordinates to the inertial velocity coordinates, and therefore the $\mathbf{DCM}_{B\to I}$ matrix is used. This results in the following equations that relate the rate of change of the aircraft's position to its velocity in body coordinates.

$$\begin{bmatrix}\dot{N}\\\dot{E}\\\dot{D}\end{bmatrix} = \mathbf{DCM}_{B\to I}\begin{bmatrix}U\\V\\W\end{bmatrix} \tag{4.22}$$

$$\begin{bmatrix} \dot{N} \\ \dot{E} \\ \dot{D} \end{bmatrix} = \begin{bmatrix} \cos\Psi\cos\Theta & \cos\Psi\sin\Theta\sin\Phi - \sin\Psi\cos\Phi & \cos\Psi\sin\Theta\cos\Phi + \sin\Psi\sin\Phi \\ \sin\Psi\cos\Theta & \sin\Psi\sin\Theta\sin\Phi + \cos\Psi\cos\Phi & \sin\Psi\sin\Theta\cos\Phi - \cos\Psi\sin\Phi \\ -\sin\Theta & \cos\Theta\sin\Phi & \cos\Theta\cos\Phi \end{bmatrix} \begin{bmatrix} U \\ V \\ W \end{bmatrix} \tag{4.23}$$

The complete six-degrees-of-freedom model is shown in Figure 4.6.

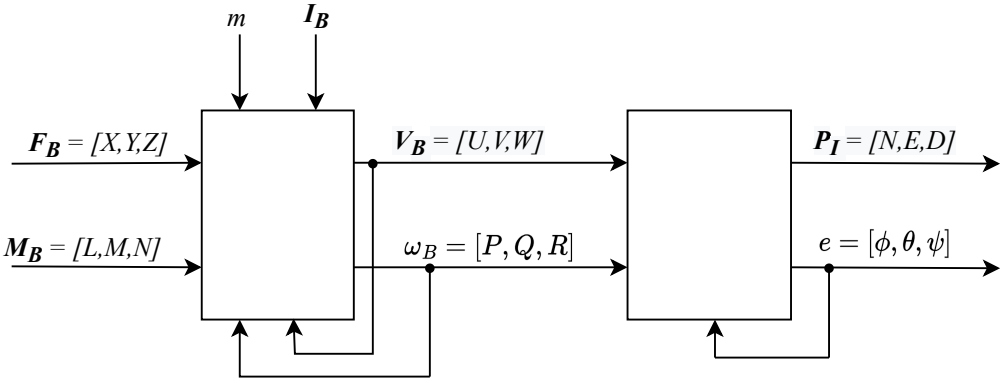


Figure 4.6: Block diagram showing the complete six degrees of freedom equations of motion.

The forces and moments in the body axis, as well as the mass and moment of inertia of the aircraft, are inputs to the model. Equations 4.9 to 4.14, which are the kinetic equations, relate these inputs to the rate of change in translational and rotational velocities in the body frame. The rate of change in velocity coordinates can then be integrated to get the velocities themselves. The velocities are then used by the kinematic Equations 4.15 and 4.23 to obtain the rates of change in attitude and position in the inertial frame.

4.3. Force and Moment Models

The inputs to the equations of motion model are the forces and moments experienced by the aircraft coordinated in its body axis system. These forces and moments are generated by three sources, which are aerodynamics, thrust and gravity. The total forces (X, Y, Z) and moments (L, M, N) acting on the aircraft is the sum of these aerodynamic, thrust and gravitational components, as shown below.

$$X = X^A + X^T + X^G \tag{4.24}$$
$$Y = Y^A + Y^T + Y^G \tag{4.25}$$
$$Z = Z^A + Z^T + Z^G \tag{4.26}$$
$$L = L^A + L^T + L^G \tag{4.27}$$
$$M = M^A + M^T + M^G \tag{4.28}$$
$$N = N^A + N^T + N^G \tag{4.29}$$

The A, T, and G superscripts indicate the aerodynamic, thrust, and gravitational components, respectively.

The forces and moments from these sources must now be modelled individually so their affect on the aircraft can be captured. This will be performed in the remainder of this section.

4.3.1 Aerodynamic Model

The aerodynamic model captures the effect of the atmosphere on the aircraft. Figure 4.7 shows the inputs and outputs of the aerodynamic model.

Figure 4.7: Block diagram showing the aerodynamic model.

The inputs are $V_{B-\text{polar}}$ which is the body frame velocity in polar form, ω_B which is the angular rates, and $\delta_\square$ which is the control surface deflection angles $(\delta_A, \delta_E, \delta_R, \delta_F)$.

Bernoulli's equation and the continuity principle for incompressible fluids can be used to show that the aerodynamic forces and moments in subsonic flight are proportional to the dynamic pressure. This pressure (q) is defined as,

$$q = \frac{1}{2}\rho\overline{V}^2 \tag{4.30}$$

where ρ is the air density and $\overline{V}$ is the aircraft airspeed magnitude. This allows the following equations to used to express the aerodynamic forces and moments.

$$X^A = qSC_{X_B} \tag{4.31}$$
$$Y^A = qSC_{Y_B} \tag{4.32}$$
$$Z^A = qSC_{Z_B} \tag{4.33}$$
$$L^A = qSbC_{l_B} \tag{4.34}$$
$$M^A = qS\bar{c}C_{m_B} \tag{4.35}$$
$$N^A = qSbC_{n_B} \tag{4.36}$$

which in vector form is,

$$\begin{aligned}\mathbf{F}^A &= \begin{bmatrix} X^A & Y^A & Z^A \end{bmatrix}^\intercal \\ &= \begin{bmatrix} qSC_{X_B} & qSC_{Y_B} & qSC_{Z_B} \end{bmatrix}^\intercal \end{aligned} \tag{4.37}$$

$$\begin{aligned}\mathbf{M}^A &= \begin{bmatrix} L^A & M^A & N^A \end{bmatrix}^\intercal \\ &= \begin{bmatrix} qSbC_{l_B} & qS\bar{c}C_{m_B} & qSbC_{n_B} \end{bmatrix}^\intercal \end{aligned} \tag{4.38}$$

where S is the area of the aircraft's wing, b is the aircraft's wing span, $\bar{c}$ is the mean aerodynamic chord and $C_{\square_B}$ are the non-dimensional coefficients for the aerodynamic forces and moments in the body frame. These coefficients represent specific aerodynamic properties of an airframe and they are independent of the aircraft size and flight speed. These coefficients are normally modelled in the wind frame with sideslip angle (β) set to zero, therefore, a conversion is required to obtain their values in the body axes. This is done with the following equations,

$$C_{X_B} = -C_D \cos(\alpha) + C_L \sin(\alpha) \tag{4.39}$$

$$C_{Y_B} = C_Y \tag{4.40}$$

$$C_{Z_B} = -C_L \cos(\alpha) - C_D \sin(\alpha) \tag{4.41}$$

$$C_{l_B} = C_l \cos(\alpha) - C_n \sin(\alpha) \tag{4.42}$$

$$C_{m_B} = C_m \tag{4.43}$$

$$C_{n_B} = C_n \cos(\alpha) + C_l \sin(\alpha) \tag{4.44}$$

where C_L is the lift coefficient, C_D is the drag coefficient, C_Y is the lateral force coefficient, C_l is the rolling moment coefficient, C_m is the pitching moment coefficient and C_n is the yawing moment coefficient, all defined in the wind frame. α is the angle of attack of the aircraft. Since the β is equal to zero, the lateral force and moment coefficients do not change from the wind axes to the body axes. Expanding these wind axes coefficients

results in,

$$C_D = C_{D_0} + \frac{C_L^2}{\pi A e} \tag{4.45}$$

$$C_y = C_{y_\beta}\,\beta + C_{y_P}\,\frac{b}{2\overline{V}}\,P_S + C_{y_R}\,\frac{b}{2\overline{V}}\,R_S + C_{y_{\delta_A}}\,\delta_A + C_{y_{\delta_R}}\,\delta_R \tag{4.46}$$

$$C_L = C_{L_0} + C_{L_\alpha}\,\alpha + C_{L_Q}\,\frac{\bar{c}}{2\overline{V}}\,Q_S + C_{L_{\delta_E}}\,\delta_E + C_{L_{\delta_F}}\,\delta_F \tag{4.47}$$

$$C_l = C_{l_\beta}\,\beta + C_{l_P}\,\frac{b}{2\overline{V}}\,P_S + C_{l_R}\,\frac{b}{2\overline{V}}\,R_S + C_{l_{\delta_A}}\,\delta_A + C_{l_{\delta_R}}\,\delta_R \tag{4.48}$$

$$C_m = C_{m_0} + C_{m_\alpha}\,\alpha + C_{m_Q}\,\frac{\bar{c}}{2\overline{V}}\,Q_S + C_{m_{\delta_E}}\,\delta_E + C_{m_{\delta_F}}\,\delta_F \tag{4.49}$$

$$C_n = C_{n_\beta}\,\beta + C_{n_P}\,\frac{b}{2\overline{V}}\,P_S + C_{n_R}\,\frac{b}{2\overline{V}}\,R_S + C_{n_{\delta_A}}\,\delta_A + C_{n_{\delta_R}}\,\delta_R \tag{4.50}$$

where C_{D_0} is the parasitic drag coefficient, C_{L_0} is the static lift coefficient, C_{m_0} is the static pitching moment coefficient, A is the aspect ratio($\frac{\text{length}}{\text{breadth}}$) of the wing, and e is the Oswald efficiency factor. To use the angular rates in these equations, they first have be transformed to the wind axes which is done with,

$$P_S = P\,\cos(\alpha) + R\,\sin(\alpha) \tag{4.51}$$

$$Q_S = Q \tag{4.52}$$

$$R_S = -P\,\sin(\alpha) + R\,\cos(\alpha) \tag{4.53}$$

The remaining coefficients from equations 4.45 to 4.50 are non-dimensional stability and control derivatives of the form,

$$C_{A_B} = \frac{\partial C_A}{\partial B'} \qquad (B' = fB) \tag{4.54}$$

where A represents a force or moment in the wind axes, B is an aircraft state or deflection angle and f is an appropriate normalising coefficient. The value of f is 1 for the angle of attack, sideslip and control surface deflections. For the pitch rate, the f value is $\frac{\bar{c}}{2\overline{V}}$, and for the roll and yaw rates, it is $\frac{b}{2\overline{V}}$. These coefficients describe the forces and moments applied to the aircraft as a function of the aircraft states and control surface deflections. Explanations regarding the meaning of the coefficients can be found in Cook [55]. Appendix A contains the values used for the constants in equations 4.45 to 4.50 for the aircraft used in this project.

4.3.2 Thrust Model

The thrust model is used to capture the effect of the motor producing thrust on the aircraft. Figure 4.8 shows the input (thrust command) and outputs of the thrust model.

The motor used on the UAV for this project is an electric brushless motor which is

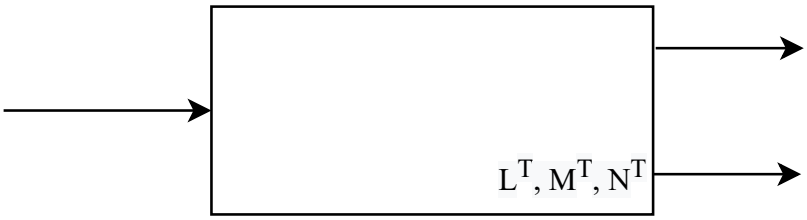


Figure 4.8: Block diagram showing the thrust model.

attached to the nose of the airframe and therefore it is operated in a puller configuration. Many complex propulsion models exist that are used for thrust modeling. However, due to the simple configuration of the motor used, this not required. A first-order lag model is sufficient to capture the band-limited nature of the propulsion source [AA833 document], and it is modelled with the following equation.

$$\dot{T} = -\frac{K_T}{\tau_e}T + \frac{K_{T_c}}{\tau_e}T_c \tag{4.55}$$

where T is the thrust magnitude, T_c is the thrust command, τ_e is the lag time constant of the motor, K_T is the thrust magnitude constant and K_{T_c} is the thrust command constant. It is normally assumed that the thrust command is equal to thrust produced and hence $K_T = K_{T_c} = 1$. This will be assumed for the classical controllers but will not be assumed for the MPC. This will be discussed further when the MPC design is described in section 5.3.2.

There is a maximum thrust that can be produced by the motor and this is used to limit the thrust commanded to the model. The maximum thrust that the motor can produce was experimentally determined. The experiment that was performed is described in Appendix A.2, and the maximum thrust was determined to be 40 N.

The thrust vector is assumed to be aligned with the X-axis of the body axis system and therefore its effect is only producing a force in the axial direction. The thrust vector is assumed to act through the centre of mass, and therefore does not produce any thrust moments. This is represented by the following,

$$\begin{aligned}\mathbf{F}^T &= \begin{bmatrix} X^T & Y^T & Z^T \end{bmatrix}^\intercal \\ &= \begin{bmatrix} T & 0 & 0 \end{bmatrix}^\intercal \end{aligned} \tag{4.56}$$

$$\begin{aligned}\mathbf{M}^T &= \begin{bmatrix} L^T & M^T & N^T \end{bmatrix}^\intercal \\ &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\intercal \end{aligned} \tag{4.57}$$

On an aircraft with a single propeller there is a yaw and roll moment generated due to the propeller rotation. This effect is negligible for this project's UAV and therefore the flight control system can treat this as disturbance and compensate appropriately.

4.3.3 Gravitational Model

The gravitational model captures the effect of gravity on the aircraft. Figure 4.9 shows the input and outputs of the model.

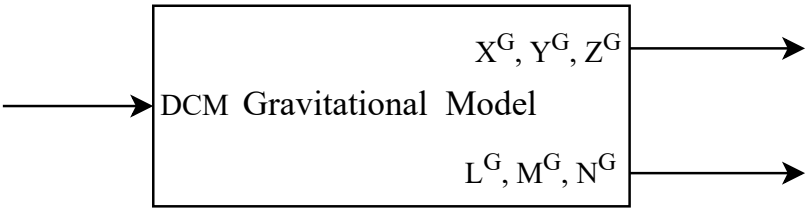


Figure 4.9: Block diagram showing the gravitational model.

Since the earth is modeled as a flat surface, it assumed that the effect of gravity is a constant force acting at the aircraft's centre of mass in the down (D) direction in the inertial frame. This corresponds to the following force vector defined in the inertial frame,

$$\mathbf{F}_I^G = \begin{bmatrix} 0 & 0 & mg \end{bmatrix}^\intercal \tag{4.58}$$

where m is the mass of the aircraft and g is the gravitational acceleration. These values can be found in appendix A.

The gravitational force must be coordinated in the body axes, which can be done by transforming the inertial vector using the DCM from Equation 4.19. This yields,

$$\begin{aligned} \mathbf{F}^G &= \begin{bmatrix} X^G & Y^G & Z^G \end{bmatrix}^\intercal \\ &= \mathbf{DCM}_{I\to B}\mathbf{F}_I^G \\ &= \begin{bmatrix} -\sin(\Theta) \\ \cos(\Theta)\sin(\Phi) \\ \cos(\Theta)\cos(\Phi) \end{bmatrix} mg \end{aligned} \tag{4.59}$$

As the UAV is assumed to be a rigid body with a fixed mass in a uniform gravitational field, the gravitational force at the centre of mass produces no moments on the aircraft. Therefore,

$$\begin{aligned} \mathbf{M}^G &= \begin{bmatrix} L^G & M^G & N^G \end{bmatrix}^\intercal \\ &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\intercal \end{aligned} \tag{4.60}$$

4.4. Wind Model

One of the major factors that causes an aircraft's flight performance to vary is wind. A tail wind can cause an aircraft to stall and crash during landing if the wind speed is too high. Since the UAV airframe used in this project is smaller and lighter then a manned

fixed-wing aircraft, it is much more susceptible to the wind. It is therefore important to model the effect of wind on the UAV.

The aerodynamic force and moments acting on the aircraft are a function of the airspeed, which is the speed of the aircraft relative to the air. The aerodynamic model should therefore use the velocity magnitude $\bar{V}$, the angle of attack α, and the sideslip angle β, that are calculated using the airspeed vector, and not the ground speed vector. The dynamic pressure should also be calculated using the airspeed, and not the ground speed. However, the equations of motion describe the motion of the vehicle relative to the ground, and not relative to the air. Given the ground speed vector provided by the equations of motion, the airspeed vector must first be calculated before it is provided as an input to the aerodynamic model. The airspeed vector is the ground speed vector minus the wind speed vector, which is expressed mathematically as,

$$\mathbf{V}^{AIR} = \mathbf{V}^{GND} - \mathbf{V}^{WND} \tag{4.61}$$

where $\mathbf{V}^{GND}$ is the ground speed vector, $\mathbf{V}^{AIR}$ is the airspeed vector and $\mathbf{V}^{WND}$ is the wind speed vector.

The wind is treated as an unknown force and moment disturbance, with the control system designed to reject these disturbances. The effect of wind is therefore not added to the aircraft model. The airspeed is assumed to be equal to the ground speed when the linearised model of the aircraft dynamics is derived. The effect of wind is still of interest when testing the control system's performance in non-linear simulation. The full wind model will therefore be implemented during the non-linear simulation. There are four wind effects that will be modelled in the preceding subsections, namely discrete gust, turbulence, wind shear and ground effect.

4.4.1 Discrete Gust Model

A discrete wind gust is described as airflow that gradually increases in speed until it reaches a certain maximum magnitude. The gust model achieves this gradual build-up using a "1-cosine" profile and for gust fade-out it uses an inverted "1-cosine" profile as shown in figure 4.10.

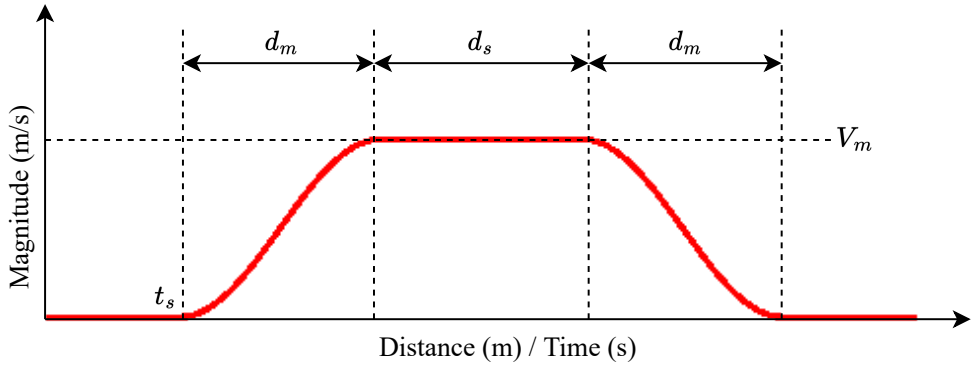


Figure 4.10: Illustration of the discrete gust wind profile.

The mathematical representation which is derived from the MIL-F-8785C military specification [59] with an addition of gust fading is given as,

$$V_{\text{gust}} = \begin{cases} 0 & t < t_s \\ \frac{V_m}{2}(1 - \cos(\frac{\pi x}{d_m})) & 0 \leq x \leq d_m \\ V_m & x > d_m \, , \, x < (d_m + d_s) \\ \frac{V_m}{2}(1 + \cos(\frac{\pi x}{d_m})) & (d_m + d_s) \leq x \leq (2d_m + d_s) \\ 0 & x > (2d_m + d_s) \end{cases} \tag{4.62}$$

where V_{gust} is the gust model output, V_m is the gust amplitude, t is the simulation time, t_s is the gust start time, x is the distanced the aircraft travelled since the gust activation at t_s, d_m is the gust build and fade distance, and d_s is the distance for maximum gust before it fades out.

4.4.2 Turbulence Model

Turbulence can be described as random forces and moments applied to an aircraft due to various factors which results in a disturbance in the linear and angular rates on the aircraft. Turbulence is modelled by passing white noise through shaping filters. Many forms of these filters exist, however for this project the Dryden spectral form is used, since it gives a good approximation of turbulence while not being too computationally expensive.

The Dryden filter equations obtained from the MIL-HDBK-1797 document [60] is shown in table 4.2. In the table, b is the aircraft wing span, V is current aircraft airspeed, σ is the turbulence intensities, and L is the turbulence scale lengths. For altitudes under 1000ft (304.8m), the scale lengths are defined as,

$$2L_w = h \tag{4.63}$$

$$L_u = 2L_v = \frac{h}{(0.177 + 0.000823h)^{1.2}} \tag{4.64}$$

and the turbulence intensities are defined as,

$$\sigma_w = 0.1V_{20} \tag{4.65}$$

$$\frac{\sigma_u}{\sigma_w} = \frac{\sigma_v}{\sigma_w} = \frac{1}{(0.177 + 0.000823h)^{0.4}} \tag{4.66}$$

where h is the aircraft altitude, and V_{20} is the average wind speed at an altitude of 20ft. The altitude lower limit is set to 20ft to prevent overaggressive non-linear scaling. For higher altitudes, the method for obtaining the turbulence scale lengths and intensities change. This is not of concern for this project as the UAV flies well under the maximum limit of 1000ft.

Table 4.2: Dryden turbulence filter form

Dryden Filter Functions	
Direction	**Filter Transfer Function**
Longitudinal	
Linear : $H_u(s)$	$\sigma_u\sqrt{\frac{2L_u}{\pi V}} \cdot \frac{1}{1+\frac{L_u}{V}s}$
Angular : $H_p(s)$	$\sigma_w\sqrt{\frac{0.8}{V}} \cdot \frac{(\frac{\pi}{4b})^{\frac{1}{6}}}{(2L_w)^{\frac{1}{3}}(1+(\frac{4b}{\pi V})s)}$
Lateral	
Linear : $H_v(s)$	$\sigma_v\sqrt{\frac{2L_v}{\pi V}} \cdot \frac{1+\frac{2\sqrt{3}L_v}{V}s}{(1+\frac{2L_v}{V}s)^2}$
Angular : $H_r(s)$	$\frac{\mp\frac{s}{V}}{(1+(\frac{3b}{\pi V})s)} \cdot H_v(s)$
Vertical	
Linear : $H_w(s)$	$\sigma_w\sqrt{\frac{2L_w}{\pi V}} \cdot \frac{1+\frac{2\sqrt{3}L_w}{V}s}{(1+\frac{2L_w}{V}s)^2}$
Angular : $H_q(s)$	$\frac{\pm\frac{s}{V}}{(1+(\frac{4b}{\pi V})s)} \cdot H_w(s)$

4.4.3 Wind Shear Model

Wind shear describes the change in wind speed as a function of altitude and the surrounding terrain. Wind shear magnitude significantly increases during extreme weather conditions such as in storms. Figure 4.11 shows how the wind speed increases significantly with altitude until it reaches a plateau. Since the UAV will be flown at a low altitude, close to the ground, it will therefore be operating in the steep gradient section of the graph.

This means that the UAV is significantly affected by shear which is why it is important to model it.

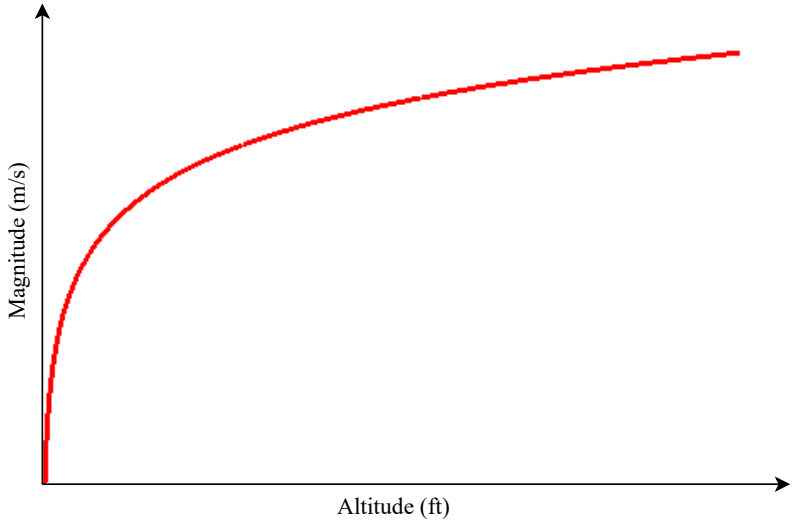


Figure 4.11: Illustration of the wind shear profile.

The mathematical representation of wind shear, which was obtained from the MIL-HDBK-1797 specification [60], is as follows,

$$\overline{v}_{\text{shear}} = V_{20} \frac{ln(\frac{h}{z_0})}{ln(\frac{20}{z_0})} \qquad 3\,\text{ft} < h < 1000\,\text{ft} \tag{4.67}$$

where $\overline{v}_{\text{shear}}$ is the average wind speed, V_{20} is the wind speed at an altitude of 20ft, h is the aircraft's current altitude, and z_0 is a constant which depends on the phase of flight. z_0 is 0.15 for flight phases labeled as Category C (takeoff, approach and landing) and is 2.0 for all the other phases. Since the UAV will be prominently tested during the landing phase, z_0 is set to 0.15. This formula is sufficient to model shear since the aircraft will be flown within the given altitude limits in equation 4.67.

4.4.4 Ground Effect

The ground effect describes the reduced wing tip vortices produced by an aircraft when it is close to the earth's surface. This results in an increase in lift for the same angle of attack and a reduction in drag on a aircraft. This disturbance in the aerodynamics can significantly affect the aircraft's landing accuracy as the aircraft would stay aloft longer than the autopilot expects. It is therefore important to model this effect to determine if the controllers adequately reject this disturbance and produce an accurate landing. The implementation of the ground effect consists of modifying the aerodynamic model by adding a multiplier to the lift (C_L) and drag (C_D) coefficients. When the aircraft is in the

ground effect these coefficients are adjusted to become,

$$C_L = G_L(h)C_L \tag{4.68}$$

$$C_D = C_{D_0} + G_D(h)\left(\frac{C_L^2}{\pi Ae}\right) \tag{4.69}$$

where $G_L(h)$ and $G_D(h)$ are altitude varying gains describe by Hull [61] as,

$$G_L(h) = 1.0 + (0.00211 - 0.0003(A - 3.0))e^{5.2(1-h/b)} \tag{4.70}$$

$$G_D(h) = 1.111 + 5.55\left(\frac{h}{b}\right) - \sqrt{29.8\left(\frac{h}{b} + 0.02\right)^2 + 0.817} \tag{4.71}$$

where h is the aircraft's altitude, A is the wing aspect ratio, and b is the wing span. The $G_D(h)$ equation is only used when $h < 0.9b$, else $G_D(h) = 1$.

4.5. Moving Platform Model

The moving platform is modelled as a point mass moving nominally in a straight line at a constant velocity, but with small in-track and cross-track velocity disturbances. This may represent, for example, an aircraft carrier keeping a constant speed and heading while a fixed-wing UAV comes in for a landing. For this project, the moving platform will be an RC car which will be manually controlled by a human driver. The RC car will be given a constant throttle setting, which should result in a constant speed (3 m/s) on level ground. The human driver will manually steer the vehicle to follow the centre line of the runway. This should be adequately modelled by the point mass model mentioned above. The in-track and cross-track velocity disturbances represent variations in the RC car speed and deviations from the centre line of the runway.

No control system is developed for the RC car as it is manually controlled. Therefore, a complex model capturing the full dynamics of the RC car is not required. The point mass model is used to simulate the moving platform in non-linear simulations when testing the control systems.

To describe the motion of the moving platform, it is helpful to introduce the runway frame or runway axis system, shown in Figure 4.12. The runway frame is the same as the inertial frame, except that it is rotated about the Z-axis by an angle Ψ_r to align the its X-axis with the centre line of the runway. Ψ_r is the heading of the runway's centre line relative to true North. The position coordinates of the moving platform in the runway frame represents its position along the centre line, its lateral deviation from the centre line, and its vertical position relative to the runway surface. The application of the runway axis system is discussed in more detail when the landing position predictor is introduced

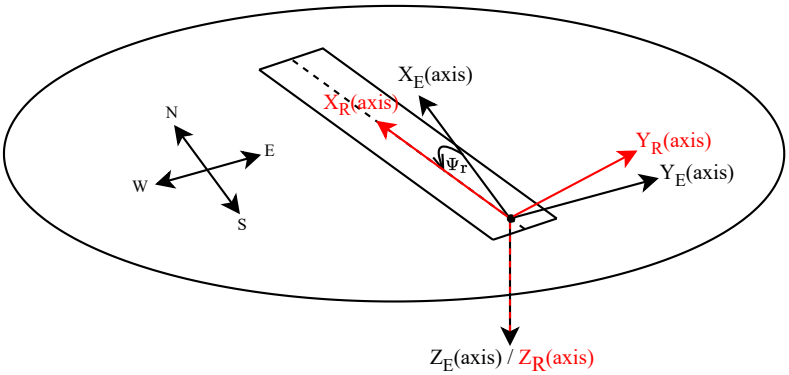


Figure 4.12: Illustration of the runway frame.

in Section 6.3.

As the moving platform moves at constant velocity in a straight line, its velocity coordinated in the runway frame can be expressed as,

$$\dot{x}_{\text{mp}} = V_{\text{mp}} + \eta_{vx} \tag{4.72}$$

$$\dot{y}_{\text{mp}} = \eta_{vy} \tag{4.73}$$

$$\dot{z}_{\text{mp}} = 0 \tag{4.74}$$

where V_{mp} is the nominal speed of the platform, and η_{vx} and η_{vy} are in-track and cross-track velocity disturbances, respectively. η_{vx} and η_{vy} are modelled as zero-mean, Gaussian noise with standard deviations of σ_{vx} and σ_{vy}, respectively.

The platform velocity can be transformed from the runway frame to the inertial frame using the following equation

$$\begin{bmatrix} \dot{N}_{\text{mp}} \\ \dot{E}_{\text{mp}} \\ \dot{D}_{\text{mp}} \end{bmatrix} = \begin{bmatrix} \cos\Psi_{\text{r}} & -\sin\Psi_{\text{r}} & 0 \\ \sin\Psi_{\text{r}} & \cos\Psi_{\text{r}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_{\text{mp}} \\ \dot{y}_{\text{mp}} \\ \dot{z}_{\text{mp}} \end{bmatrix} \tag{4.75}$$

where Ψ_{r} is the heading of the runway's center line relative to true north. Finally, the instantaneous position of the moving platform can be obtained by integrating the platform velocity with respect to time, starting at its initial position, using

$$\mathbf{p}_{\text{mp}}(t) = \int_0^t \mathbf{V}_{\text{mp}}(\tau)\, d\tau + \mathbf{p}_{\text{mp}}(0) \tag{4.76}$$

where $\mathbf{p}_{\text{mp}}(t)$ and $\mathbf{V}_{\text{mp}}(t)$ are the instantaneous position and velocity of the moving platform at time t, and $\mathbf{p}_{\text{mp}}(0)$ is the initial position of the moving platform at time $t = 0$.

4.6. Linearisation of Aircraft Model

A linear model is required to analyse the aircraft model dynamics and to design the control system. The non-linear aircraft model, which comprises of the equation of motion combined with the force and moment models, can be linearised around an equilibrium point to achieve this linear model.

4.6.1 Obtaining the Trim Variables

The equilibrium point chosen is the state when the aircraft is in straight and level flight. At this point the sum of all the forces and moments acting on the aircraft equals to zero. To maintain its equilibrium point, the aircraft will have trim values for its states. Since the aircraft is symmetrical around its XZ-plane, and the equilibrium point is level flight, all the trim lateral motion and control parameters are zero. This means that the aircraft only needs to be considered for the longitudinal and normal axes (XZ-plane). The trim variables that need to be calculated are $(\overline{V}, \alpha, Q, \Theta, \delta_E, T)_T$. For trimmed flight, the flap deflection angle (δ_{F_T}) will be assumed to be zero. The $(\Psi, D)_T$ variables are omitted from the trim variables as the aircraft can be at any altitude and heading for level flight. The (N, E) variables are also not included, because they do not remain constant for forward flight, and are the kinematic result of the equilibrium state. Figure 4.13 shows the trim force and moment diagram in XZ-plane for an aircraft in level flight.

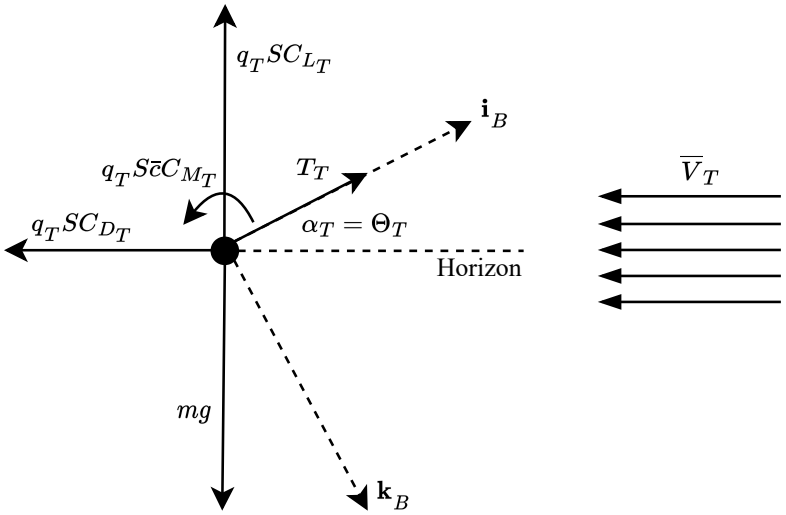


Figure 4.13: Diagram showing the trim forces and moments in the XZ-plane.

It is common to specify the desired trim airspeed $\overline{V}_T$ and trim altitude, and then to calculate the required trim angle of attack, trim elevator deflection, and trim thrust $(\alpha, \delta_E, T)_T$ to achieve the straight and level equilibrium condition. The trim altitude must be specified, because it determines the trim air density ρ_T, and thus the trim dynamic pressure q_T. The desired trim airspeed must be chosen between the stall speed and the

maximum airspeed. De Bruin [16] obtained these values with practical experiments, and since he used the same airframe and motor as the UAV in this project, his values can be used. His values were $V_{\text{stall}} = 10.8$ m/s and $V_{\text{max}} = 25$ m/s and therefore $\overline{V}_T$ is chosen to be 18 m/s. This speed was chosen as it is significantly above the stall speed while not being so high that it will strain the motor.
The trim dynamic pressure q_T is calculated as,

$$q_T = \frac{1}{2}\rho_T \overline{V}_T^2 \tag{4.77}$$

The trim air density is approximated as $\rho_T = 1.225$ kg/m^3 which the air density at sea level at 15°C. The aircraft will be flown at airfield close to sea level and therefore this air density is safe to assume. Given, the trim airspeed and trim dynamic pressure, the trim angle of attack and trim elevator setting are calculated using the following equations,

$$\begin{bmatrix} \alpha_T \\ \delta_{E_T} \end{bmatrix} = \begin{bmatrix} C_{L_\alpha} & C_{L_{\delta_E}} \\ C_{m_\alpha} & C_{m_{\delta_E}} \end{bmatrix}^{-1} \begin{bmatrix} \frac{mg}{q_T S} - C_{L_0} \\ -C_{m_0} \end{bmatrix} \tag{4.78}$$

The trim thrust setting is then calculated using the following equation,

$$T_T = q_T S C_{D_T} cos(\alpha_T) - q_T S C_{L_T} sin(\alpha_T) + mg\, sin(\alpha_T) \tag{4.79}$$

where,

$$C_{D_T} = C_{D_0} + \frac{C_{L_T}^2}{\pi A e} \tag{4.80}$$

The derivation of these equations are provided in Appendix B.4.

Using the specifications of the UAV used in this project (see appendix A), the trim values are calculated as,

$$\alpha_T = 0.0649 \text{ rad} \tag{4.81}$$

$$\delta_{E_T} = -0.0558 \text{ rad} \tag{4.82}$$

$$T_T = 26.5513 \text{ N} \tag{4.83}$$

4.6.2 Linearising Equations of Motion around Trim

The nonlinear aircraft model can now be linearised around the calculated trim state and trim control inputs. The assumptions made during the linearisation process are:

- The products of small perturbations are small and are therefore ignored
- The cosine of a small angle is one

- The sine of a small angle is the angle itself in radians

The nonlinear aircraft flight dynamics consists of the kinetic equations, the kinematic equations, and the force and moment models for the aerodynamics, gravity, and thrust.

The kinetic and kinematic equations are rearranged so that the state derivatives are the subject of the equations, as follows,

$$\dot{U} = \frac{X}{m} + VR - WQ \tag{4.84}$$

$$\dot{V} = \frac{Y}{m} - UR + WP \tag{4.85}$$

$$\dot{W} = \frac{Z}{m} + UQ - VP \tag{4.86}$$

$$\dot{P} = \frac{L}{I_{xx}} - QR\frac{I_{zz} - I_{yy}}{I_{xx}} \tag{4.87}$$

$$\dot{Q} = \frac{M}{I_{yy}} - PR\frac{I_{xx} - I_{zz}}{I_{yy}} \tag{4.88}$$

$$\dot{R} = \frac{N}{I_{zz}} - PQ\frac{I_{yy} - I_{xx}}{I_{zz}} \tag{4.89}$$

$$\dot{\Phi} = P + Q\sin\Phi\tan\Theta + R\cos\Phi\tan\Theta \tag{4.90}$$

$$\dot{\Theta} = Q\cos\Phi - R\sin\Phi \tag{4.91}$$

This forms a coupled set of differential equations which capture the primary dynamics of the aircraft. The dynamic states Ψ, N, E and D are not used for linearisation as they not part of the primary flight dynamics and are instead the kinematic results of them. The primary dynamic equations can be written in a nonlinear state space form as,

$$\dot{\mathbf{x}} = \mathbf{f(x,u)} \tag{4.92}$$

where,

$$\mathbf{x} = \begin{bmatrix} U & V & W & P & Q & R & \Phi & \Theta \end{bmatrix}^\intercal \tag{4.93}$$

$$\mathbf{u} = \begin{bmatrix} \delta_E & \delta_F & \delta_A & \delta_R & T \end{bmatrix}^\intercal \tag{4.94}$$

and $\mathbf{f}$ is the vector function of the corresponding dynamic equation. Each state and control variable can be written as a sum of the trim value and perturbation around trim which gives,

$$\mathbf{x} = \mathbf{x}_T + \Delta\mathbf{x} \tag{4.95}$$

$$\mathbf{u} = \mathbf{u}_T + \Delta\mathbf{u} \tag{4.96}$$

where,

$$\Delta x = \begin{bmatrix} u & v & w & p & q & r & \phi & \theta \end{bmatrix}^\intercal \tag{4.97}$$

$$\Delta u = \begin{bmatrix} \delta_e & \delta_f & \delta_a & \delta_r & \Delta T \end{bmatrix}^\intercal \tag{4.98}$$

The linearised aircraft model is obtained by taking the partial derivatives of the nonlinear differential equations with respect to the states and the control inputs. The following linear state space model for the aircraft dynamics is then obtained,

$$\Delta \dot{\mathbf{x}} \approx \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_T \Delta \mathbf{x} + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_T \Delta \mathbf{u} \tag{4.99}$$

Solving equation 4.99 while grouping the longitudinal and lateral dynamic states together and then expanding yields,

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \\ \dot{v} \\ \dot{p} \\ \dot{r} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \frac{\partial \dot{U}}{\partial U} & \frac{\partial \dot{U}}{\partial W} & \frac{\partial \dot{U}}{\partial Q} & \frac{\partial \dot{U}}{\partial \Theta} & \frac{\partial \dot{U}}{\partial V} & \frac{\partial \dot{U}}{\partial P} & \frac{\partial \dot{U}}{\partial R} & \frac{\partial \dot{U}}{\partial \Phi} \\ \frac{\partial \dot{W}}{\partial U} & \frac{\partial \dot{W}}{\partial W} & \frac{\partial \dot{W}}{\partial Q} & \frac{\partial \dot{W}}{\partial \Theta} & \frac{\partial \dot{W}}{\partial V} & \frac{\partial \dot{W}}{\partial P} & \frac{\partial \dot{W}}{\partial R} & \frac{\partial \dot{W}}{\partial \Phi} \\ \frac{\partial \dot{Q}}{\partial U} & \frac{\partial \dot{Q}}{\partial W} & \frac{\partial \dot{Q}}{\partial Q} & \frac{\partial \dot{Q}}{\partial \Theta} & \frac{\partial \dot{Q}}{\partial V} & \frac{\partial \dot{Q}}{\partial P} & \frac{\partial \dot{Q}}{\partial R} & \frac{\partial \dot{Q}}{\partial \Phi} \\ \frac{\partial \dot{\Theta}}{\partial U} & \frac{\partial \dot{\Theta}}{\partial W} & \frac{\partial \dot{\Theta}}{\partial Q} & \frac{\partial \dot{\Theta}}{\partial \Theta} & \frac{\partial \dot{\Theta}}{\partial V} & \frac{\partial \dot{\Theta}}{\partial P} & \frac{\partial \dot{\Theta}}{\partial R} & \frac{\partial \dot{\Theta}}{\partial \Phi} \\ \frac{\partial \dot{V}}{\partial U} & \frac{\partial \dot{V}}{\partial W} & \frac{\partial \dot{V}}{\partial Q} & \frac{\partial \dot{V}}{\partial \Theta} & \frac{\partial \dot{V}}{\partial V} & \frac{\partial \dot{V}}{\partial P} & \frac{\partial \dot{V}}{\partial R} & \frac{\partial \dot{V}}{\partial \Phi} \\ \frac{\partial \dot{P}}{\partial U} & \frac{\partial \dot{P}}{\partial W} & \frac{\partial \dot{P}}{\partial Q} & \frac{\partial \dot{P}}{\partial \Theta} & \frac{\partial \dot{P}}{\partial V} & \frac{\partial \dot{P}}{\partial P} & \frac{\partial \dot{P}}{\partial R} & \frac{\partial \dot{P}}{\partial \Phi} \\ \frac{\partial \dot{R}}{\partial U} & \frac{\partial \dot{R}}{\partial W} & \frac{\partial \dot{R}}{\partial Q} & \frac{\partial \dot{R}}{\partial \Theta} & \frac{\partial \dot{R}}{\partial V} & \frac{\partial \dot{R}}{\partial P} & \frac{\partial \dot{R}}{\partial R} & \frac{\partial \dot{R}}{\partial \Phi} \\ \frac{\partial \dot{\Phi}}{\partial U} & \frac{\partial \dot{\Phi}}{\partial W} & \frac{\partial \dot{\Phi}}{\partial Q} & \frac{\partial \dot{\Phi}}{\partial \Theta} & \frac{\partial \dot{\Phi}}{\partial V} & \frac{\partial \dot{\Phi}}{\partial P} & \frac{\partial \dot{\Phi}}{\partial R} & \frac{\partial \dot{\Phi}}{\partial \Phi} \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \\ v \\ p \\ r \\ \phi \end{bmatrix} + \begin{bmatrix} \frac{\partial \dot{U}}{\partial \delta_E} & \frac{\partial \dot{U}}{\partial \delta_F} & \frac{\partial \dot{U}}{\partial T} & \frac{\partial \dot{U}}{\partial \delta_A} & \frac{\partial \dot{U}}{\partial \delta_R} \\ \frac{\partial \dot{W}}{\partial \delta_E} & \frac{\partial \dot{W}}{\partial \delta_F} & \frac{\partial \dot{W}}{\partial T} & \frac{\partial \dot{W}}{\partial \delta_A} & \frac{\partial \dot{W}}{\partial \delta_R} \\ \frac{\partial \dot{Q}}{\partial \delta_E} & \frac{\partial \dot{Q}}{\partial \delta_F} & \frac{\partial \dot{Q}}{\partial T} & \frac{\partial \dot{Q}}{\partial \delta_A} & \frac{\partial \dot{Q}}{\partial \delta_R} \\ \frac{\partial \dot{\Theta}}{\partial \delta_E} & \frac{\partial \dot{\Theta}}{\partial \delta_F} & \frac{\partial \dot{\Theta}}{\partial T} & \frac{\partial \dot{\Theta}}{\partial \delta_A} & \frac{\partial \dot{\Theta}}{\partial \delta_R} \\ \frac{\partial \dot{V}}{\partial \delta_E} & \frac{\partial \dot{V}}{\partial \delta_F} & \frac{\partial \dot{V}}{\partial T} & \frac{\partial \dot{V}}{\partial \delta_A} & \frac{\partial \dot{V}}{\partial \delta_R} \\ \frac{\partial \dot{P}}{\partial \delta_E} & \frac{\partial \dot{P}}{\partial \delta_F} & \frac{\partial \dot{P}}{\partial T} & \frac{\partial \dot{P}}{\partial \delta_A} & \frac{\partial \dot{P}}{\partial \delta_R} \\ \frac{\partial \dot{R}}{\partial \delta_E} & \frac{\partial \dot{R}}{\partial \delta_F} & \frac{\partial \dot{R}}{\partial T} & \frac{\partial \dot{R}}{\partial \delta_A} & \frac{\partial \dot{R}}{\partial \delta_R} \\ \frac{\partial \dot{\Phi}}{\partial \delta_E} & \frac{\partial \dot{\Phi}}{\partial \delta_F} & \frac{\partial \dot{\Phi}}{\partial T} & \frac{\partial \dot{\Phi}}{\partial \delta_A} & \frac{\partial \dot{\Phi}}{\partial \delta_R} \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_f \\ \Delta T \\ \delta_a \\ \delta_r \end{bmatrix} \tag{4.100}$$

This equation represents the entire linear model with the longitudinal and lateral components coupled together. It is more convenient to work with the velocity magnitude, angle of attack and sideslip angle deviations from trim $(\bar{v}, \alpha, \beta)$ in the state vector instead of the axial, lateral and normal velocity deviations (u, v, w) currently in the vector. To replace these states, equations 4.6 to 4.8 are simplified for straight and level flight using the assumptions mentioned at the beginning of this subsection. Assuming the angle of attack and sideslip angle are small, the simplifications are,

$$U = \overline{V}_T \, cos(\alpha) \, cos(\beta) \approx \overline{V}_T \tag{4.101}$$

$$V = \overline{V}_T \, sin(\beta) \approx \overline{V}_T \, \beta \tag{4.102}$$

$$W = \overline{V}_T \, sin(\alpha) \, cos(\beta) \approx \overline{V}_T \, \alpha \tag{4.103}$$

where $\overline{V}_T$ is the trim velocity magnitude. As the aircraft is symmetrical, the linear aircraft model can be simplified by decoupling the longitudinal and lateral dynamics. The coupled

linear model in equation 4.100 can be written as,

$$\begin{bmatrix} \Delta\dot{\mathbf{x}}_{Long} \\ \Delta\dot{\mathbf{x}}_{Lat} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{T_{11}} & \mathbf{A}_{T_{12}} \\ \mathbf{A}_{T_{21}} & \mathbf{A}_{T_{22}} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}_{Long} \\ \Delta\mathbf{x}_{Lat} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_{T_{11}} & \mathbf{B}_{T_{12}} \\ \mathbf{B}_{T_{21}} & \mathbf{B}_{T_{22}} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u}_{Long} \\ \Delta\mathbf{u}_{Lat} \end{bmatrix} \tag{4.104}$$

where T_{12} and T_{21} are the matrices that couple the longitudinal and lateral dynamics. As the aircraft is symmetrical in the XZ-plane, the $\mathbf{A}_{T_{21}}$ and $\mathbf{B}_{T_{21}}$ terms are equal to zero. Since the deviations from trim are required to be small for the linearisation assumption, it can therefore be approximated that $\mathbf{A}_{T_{12}}$ and $\mathbf{B}_{T_{12}}$ are equal zero. Using these assumptions, the full linearised model is decoupled into the longitudinal and lateral dynamic models which are given as,

$$\Delta\dot{\mathbf{x}}_{Long} = \mathbf{A}_{T_{11}}\Delta\mathbf{x}_{Long} + \mathbf{B}_{T_{11}}\Delta\mathbf{u}_{Long} \tag{4.105}$$

$$\Delta\dot{\mathbf{x}}_{Lat} = \mathbf{A}_{T_{22}}\Delta\mathbf{x}_{Lat} + \mathbf{B}_{T_{22}}\Delta\mathbf{u}_{Lat} \tag{4.106}$$

Expanding these equations while also using the simplifications in equations 4.101 to 4.103; the longitudinal linear dynamic model is given as,

$$\begin{bmatrix} \dot{\bar{v}} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{\partial\dot{U}}{\partial U} & \overline{V}_T\frac{\partial\dot{U}}{\partial W} & \frac{\partial\dot{U}}{\partial Q} & \frac{\partial\dot{U}}{\partial\Theta} \\ \frac{1}{\overline{V}_T}\frac{\partial\dot{W}}{\partial U} & \frac{\partial\dot{W}}{\partial W} & \frac{1}{\overline{V}_T}\frac{\partial\dot{W}}{\partial Q} & \frac{1}{\overline{V}_T}\frac{\partial\dot{W}}{\partial\Theta} \\ \frac{\partial\dot{Q}}{\partial U} & \overline{V}_T\frac{\partial\dot{Q}}{\partial W} & \frac{\partial\dot{Q}}{\partial Q} & \frac{\partial\dot{Q}}{\partial\Theta} \\ \frac{\partial\dot{\Theta}}{\partial U} & \overline{V}_T\frac{\partial\dot{\Theta}}{\partial W} & \frac{\partial\dot{\Theta}}{\partial Q} & \frac{\partial\dot{\Theta}}{\partial\Theta} \end{bmatrix} \begin{bmatrix} \bar{v} \\ \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\partial\dot{U}}{\partial\delta_E} & \frac{\partial\dot{U}}{\partial\delta_F} & \frac{\partial\dot{U}}{\partial T} \\ \frac{1}{\overline{V}_T}\frac{\partial\dot{W}}{\partial\delta_E} & \frac{1}{\overline{V}_T}\frac{\partial\dot{W}}{\partial\delta_F} & \frac{1}{\overline{V}_T}\frac{\partial\dot{W}}{\partial T} \\ \frac{\partial\dot{Q}}{\partial\delta_E} & \frac{\partial\dot{Q}}{\partial\delta_F} & \frac{\partial\dot{Q}}{\partial T} \\ \frac{\partial\dot{\Theta}}{\partial\delta_E} & \frac{\partial\dot{\Theta}}{\partial\delta_F} & \frac{\partial\dot{\Theta}}{\partial T} \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_f \\ \Delta T \end{bmatrix} \tag{4.107}$$

and the lateral linear dynamic model is given as,

$$\begin{bmatrix} \dot{\beta} \\ \dot{p} \\ \dot{r} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \frac{\partial\dot{V}}{\partial V} & \frac{1}{\overline{V}_T}\frac{\partial\dot{V}}{\partial P} & \frac{1}{\overline{V}_T}\frac{\partial\dot{V}}{\partial R} & \frac{1}{\overline{V}_T}\frac{\partial\dot{V}}{\partial\Phi} \\ \overline{V}_T\frac{\partial\dot{P}}{\partial V} & \frac{\partial\dot{P}}{\partial P} & \frac{\partial\dot{P}}{\partial R} & \frac{\partial\dot{P}}{\partial\Phi} \\ \overline{V}_T\frac{\partial\dot{R}}{\partial V} & \frac{\partial\dot{R}}{\partial P} & \frac{\partial\dot{R}}{\partial R} & \frac{\partial\dot{R}}{\partial\Phi} \\ \overline{V}_T\frac{\partial\dot{\Phi}}{\partial V} & \frac{\partial\dot{\Phi}}{\partial P} & \frac{\partial\dot{\Phi}}{\partial R} & \frac{\partial\dot{\Phi}}{\partial\Phi} \end{bmatrix} \begin{bmatrix} \beta \\ p \\ r \\ \phi \end{bmatrix} + \begin{bmatrix} \frac{1}{\overline{V}_T}\frac{\partial\dot{V}}{\partial\delta_A} & \frac{1}{\overline{V}_T}\frac{\partial\dot{V}}{\partial\delta_R} \\ \frac{\partial\dot{P}}{\partial\delta_A} & \frac{\partial\dot{P}}{\partial\delta_R} \\ \frac{\partial\dot{R}}{\partial\delta_A} & \frac{\partial\dot{R}}{\partial\delta_R} \\ \frac{\partial\dot{\Phi}}{\partial\delta_A} & \frac{\partial\dot{\Phi}}{\partial\delta_R} \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \tag{4.108}$$

The MATLAB Symbolic Toolbox is used to evaluate these equations and solve the partial derivatives. The resulting longitudinal and lateral linear models are compared to Etkin and Reid's [56] approximation of these models and it is found that the results are almost identical. This gives confidence that the derived linear models are indeed correct. The

longitudinal linear model derived after using the Symbolic toolbox is,

$$\begin{bmatrix} \dot{\bar{v}} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -0.4306 & 10.5352 & -1.1570 & -9.7894 \\ -0.0622 & -4.1956 & 0.9070 & -0.0353 \\ 0 & -40.2578 & -6.6282 & 0 \\ 0 & 0 & 1.0000 & 0 \end{bmatrix} \begin{bmatrix} \bar{v} \\ \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 0.0624 & 0.1394 & 0.1699 \\ -0.5485 & -1.2256 & 0 \\ -97.4349 & 11.8157 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_f \\ \Delta T \end{bmatrix} \tag{4.109}$$

and the lateral linear model is,

$$\begin{bmatrix} \dot{\beta} \\ \dot{p} \\ \dot{r} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} -0.2757 & 0.0717 & -0.9891 & 0.5439 \\ -32.1681 & -12.2162 & 3.0505 & 0 \\ 10.5809 & -0.8892 & -1.0324 & 0 \\ 0 & 1.0000 & 0.0650 & 0 \end{bmatrix} \begin{bmatrix} \beta \\ p \\ r \\ \phi \end{bmatrix} + \begin{bmatrix} 0.0010 & 0.1513 \\ -140.5221 & 2.2681 \\ -2.9175 & -15.2938 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \tag{4.110}$$

4.7. Natural Modes of Motion

To design an effective flight control system, the linear aircraft models need to be analysed to identify the natural modes of motion. The modes of motion are obtained by calculating the poles of the linearised models.

4.7.1 Longitudinal Modes of Motion

The poles of the longitudinal dynamics are obtained by calculating the eigenvalues of the longitudinal system matrix in Equation 4.109. The pole plot of the longitudinal dynamics is shown in Figure 4.14.

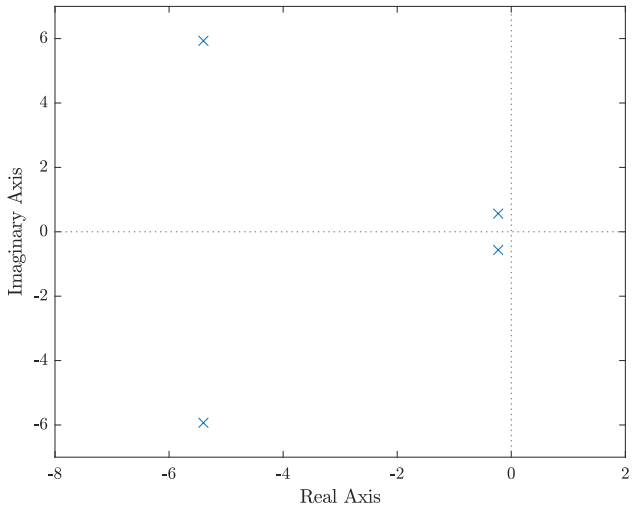


Figure 4.14: Pole Plot for the Linear Longitudinal Model.

The pole plot shows two complex pole pairs which define the longitudinal modes of

motion. The high frequency pair represents the Short Period Mode, and the low frequency pair represents the Phugoid Mode.

4.7.1.1 Short Period Mode

The Short Period Mode poles for the UAV are calculated as,

$$\mathbf{p}_{sp} = -5.3978 \pm 5.9312i \tag{4.111}$$

$$\zeta = 0.6731 \tag{4.112}$$

$$\omega_n = 8.0197\,\text{rad/s} \tag{4.113}$$

The pole locations indicate that the short period mode is stable, well damped, and has a high natural frequency. This mode describes the aircraft's ability to realign itself with its velocity vector when disturbed. A stable aircraft that experiences a disturbance in trimmed level flight will exhibit a restoring pitching moment produced by C_{m_α} due to it having a change in angle of attack. The pitch rate motion induces damping, produced by C_{m_Q}, which removes energy from the system, causing stable oscillatory behaviour. For short period motions, the aircraft can be modeled as only rotating about the Y_B axis, with an aerodynamic spring and damping torques that cause the aircraft to realign itself with incident airflow.

4.7.1.2 Phugoid Mode

The Phugoid Mode poles for the UAV are calculated as,

$$\mathbf{p}_{ph} = -0.2294 \pm 0.5650i \tag{4.114}$$

$$\zeta = 0.3762 \tag{4.115}$$

$$\omega_n = 0.6098\,\text{rad/s} \tag{4.116}$$

The pole locations indicate that the phugoid mode is stable, underdamped, and has a low natural frequency. This mode describes the aircraft's tendency to exchange potential and kinetic energy when it is disturbed from trimmed level flight. For example, if an aircraft in level flight experiences a sudden increase in velocity, it will have an increase in lift. This will cause it to pitch up increasing its altitude and thus gaining in potential energy. At the same time, it loses kinetic energy, causing its airspeed to decrease which results in a decrease in lift. This causes the aircraft to pitch down and lose altitude. This sinusoidal behaviour repeats itself continuously and is governed by this mode's damping which removes energy via aerodynamic drag.

4.7.2 Lateral Modes of Motion

The poles of the lateral dynamics are obtained by calculating the eigenvalues of the lateral system matrix in Equation 4.110. The pole plot of the longitudinal dynamics is shown in Figure 4.15.

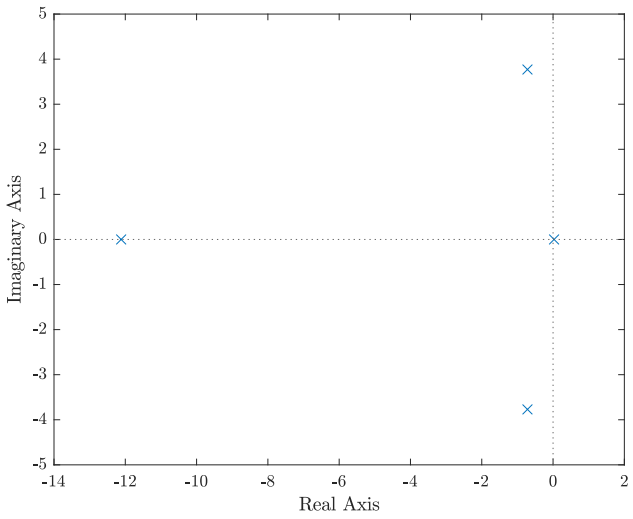


Figure 4.15: Pole Plot for the Linear Lateral Model.

The pole plot shows two real poles and one complex pole pair which define the lateral modes of motion. The high frequency real pole represents the Roll Mode, the complex pole pair represents the Dutch Roll Mode, and the unstable real pole represents the Spiral Mode.

4.7.2.1 Roll Mode

The Roll Mode pole for the UAV is calculated as,

$$p_{rl} = -12.1163 \tag{4.117}$$

The pole has a high frequency and is critically damped with a first order exponential response. This mode describes the aircraft's roll rate dynamics when deviating from trim. When a roll moment disturbance is applied to an aircraft, its roll rate will grow quickly. A counter moment, which acts as damping, is then generated by the wings due to the differential lift it experiences as it rolls. The differential lift is caused by the angle of incidence being greater on the lower wing then the higher wing. This form of damping is governed by the C_{l_P} stability derivative. Due to the pole's high frequency, the roll dynamic response is very fast, which results in the aircraft appearing to operate at a constant roll rate when a control moment is applied.

4.7.2.2 Dutch Roll Mode

The Dutch Roll Mode poles for the UAV are calculated as,

$$\mathbf{p}_{dr} = -0.7183 \pm 3.7701i \tag{4.118}$$

$$\zeta = 0.1872 \tag{4.119}$$

$$\omega_n = 3.8379\,\text{rad/s} \tag{4.120}$$

The pole pair is stable, fast, and poorly damped. The Dutch Roll Mode describes the aircraft's tendency to laterally realign itself with incident airflow due to changes in sideslip angle. This mode is the lateral direction equivalent of Short Period Mode, but with worse natural damping. When an aircraft under trim experiences a deviation in sideslip, the vertical stabiliser will produce a restoring yaw moment defined by C_{n_β} to align the nose back into oncoming airflow. The yaw rate associated with this motion provides damping and is defined by C_{n_R}. During this yaw motion, the wings experience different velocities which causes differential lift and drag on wings defined by C_{l_R} and C_{n_R}. This differential lift causes deviations in roll rate which causes further deviations in lift and drag through the C_{l_P} and C_{n_P} stability derivatives. The deviations in drag help dampen the yaw rate motions. The net effect of this is oscillation in both the yaw and roll axes which causes the wingtips to translate in an elliptical pattern as the aircraft flies.

4.7.2.3 Spiral Mode

The Spiral Mode pole for the UAV is calculated as,

$$p_{sp} = 0.0284 \tag{4.121}$$

This pole is unstable and extremely slow. This mode describes the aircraft's tendency to restore itself to or diverge away from wings level flight, when experiencing a disturbance in sideslip which causes a deviation in roll angle. When a trimmed aircraft experiences a disturbance in sideslip a yaw moment, defined by C_{n_β}, is produced which turns the aircraft in the direction of the sideslip. This results in differential lift producing a rolling moment, defined by C_{l_R}, which causes the wing in the direction of the turn to drop even further. This effect can be unstable if the restoring moment, defined by C_{l_β}, which is dependent on the wing dihedral, causes the aircraft to diverge from level flight. The UAV used for this project has an unstable Spiral Mode, as calculated using the airframe properties summaries in Appendix A, and will therefore naturally tend to diverge from level flight. This means that the flight control system must be designed to stablise the spiral mode, by using feedback to make the corresponding closed-loop pole stable.

4.8. Summary

A non-linear model of the fixed-wing UAV was presented by using the aircraft's equation of motion, and force and moment models. The wind and moving platform models used in non-linear simulations were then described. The non-linear model was then linearised around an equilibrium point to produce a linear model that is used to design the control systems. The UAV's natural modes of motion were also analysed to identify key characteristics of the aircraft's motion so that an effective flight control system could be designed. Now that the linearised fixed-wing UAV model has been developed, the flight control system can be designed using this model, which is presented in the next chapter.

Chapter 5

Flight Control System Development

Now that the linear model for the fixed-wing UAV has been developed, the flight control system (FCS) design can be presented. First, an overview of the entire flight control system architecture will be discussed, to provide an understanding of the FCS and highlight the interactions between the different controllers. Thereafter, the classical controller design procedure will be described, for both the longitudinal and lateral controller groups. The MPC design procedure will then be described, by first discussing the MPC's theory in general and then detailing the design of the MPC used for this research project.

5.1. Flight Control System Overview

The flight control system (FCS) combines two distinct forms of control which are classical control and model predictive control (MPC). The reason for using this hybrid approach is so that FCS can have the predictability of the classical control while also having the high accurate landing performance of the MPC. The classical controllers are adapted from De Bruin [16] and Le Roux [14] who themselves iterated on the controllers of previous students. This was one of the main reasons why this classical control architecture was chosen as it had a proven track record in successfully controlling the UAV in different scenarios. The aircraft specifications for the fixed-wing UAV in this project are different from De Bruin's and Le Roux's UAVs. Therefore, the entire design process for the classical controllers had followed to ensure that the controllers were within their design requirements. The design process will be thoroughly discussed in this chapter. The MPC architecture is base on those described by Wang [62] and Amadi [38] however, it was altered to control a fixed-wing aircraft. Figure 5.1 shows the layout of the complete FCS.

The references for the FCS are provided by the guidance control system (GCS) which consists of a state machine and a guidance algorithm. The GCS will be discussed in chapter 6.

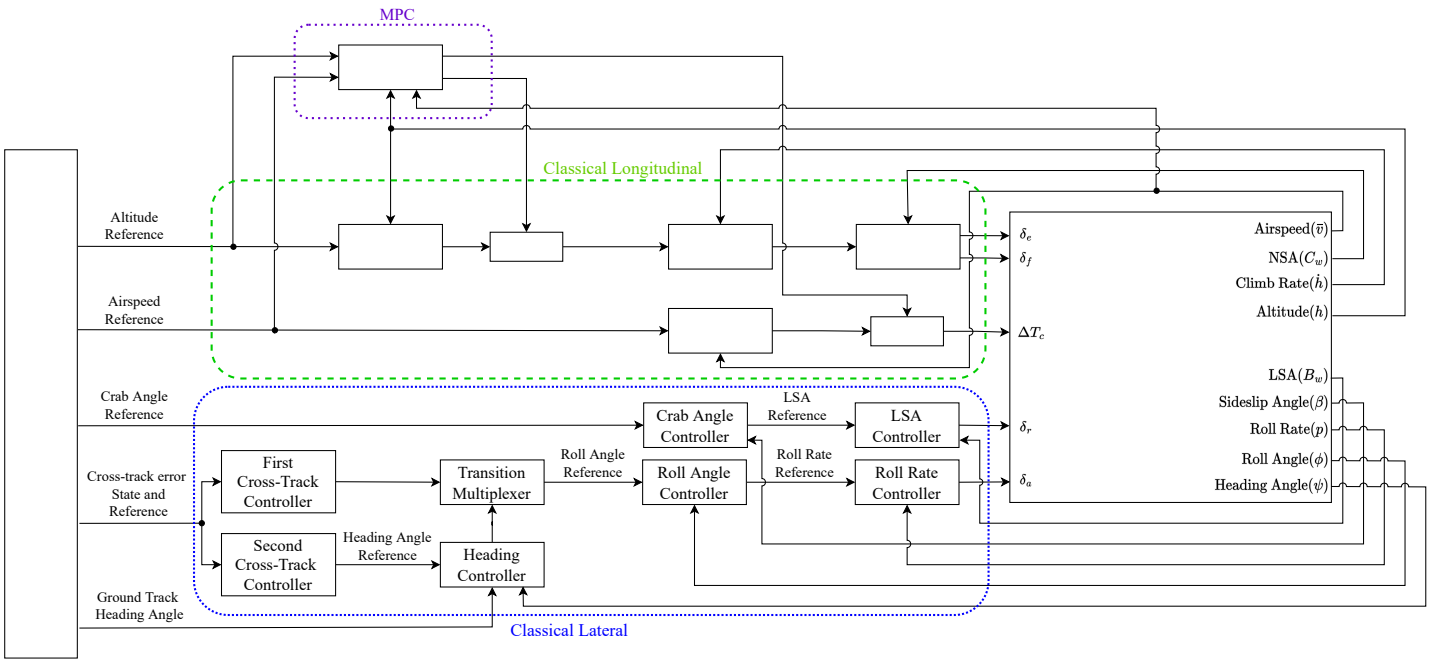


Figure 5.1: Block diagram showing the overview of the FCS.

5.1.1 Classical Control Overview

The classical controllers use a successive loop closure design where the outer loops feed the inner loops. The inner-loop controllers are used to command the aircraft control inputs, and are mainly used for stability augmentation. The stability of the inner-loop controllers are passed on to the outer-loop controllers. The outer-loop controllers are used to control the aircraft's translational motion and attitude in the inertial frame. The inner loops use acceleration-based control as it has numerous advantages, such as attitude independence, disturbance rejection, and practical feasibility due to computational efficiency.

The classical controllers are split into two groups, namely the longitudinal controllers and the lateral controllers. These control groups were formed because the aircraft's longitudinal and lateral dynamics are uncoupled, or at least weakly coupled. This allows the two classical controller groups to be designed independently. The longitudinal controllers consist of:

- The airspeed controller which controls the aircraft's calibrated airspeed by providing the thrust command to the brushless DC motor.
- The normal specific acceleration direct lift control (NSADLC) controller which controls the normal specific acceleration (NSA) by commanding the aircraft's elevator and flap control surface deflections. The NSA reference for this controller comes from the outer climb rate controller.
- The climb rate controller which controls the climb rate by providing NSA references to the NSADLC controller. The climb rate reference comes from the outer altitude

controller.

- The altitude controller which controls the aircraft's altitude by commanding the climb rate reference.

The lateral controllers consist of:

- The lateral specific acceleration (LSA) controller which controls the lateral acceleration as well provides dutch roll damping. This controller's reference is usually set to zero for conventional flight (waypoint navigation) to allow for coordinated turns. The LSA reference only changes during the landing phase, when the de-crab manoeuvre is executed by the crab angle controller, which provides this reference.
- The crab angle controller is only activated during the landing phase of flight to execute the de-crab manoeuvre so that the aircraft lands with its landing gear aligned with the runway or moving platform. During this stage, the crab angle controller controls the aircraft's crab angle. The crab angle is the yaw angle with respect to the waypoint track (which, for the landing phase, is the centre line of the runway). The controller uses the LSA controller, by providing references to it, to obtain its desired crab angle.
- The roll rate controller controls the roll rate by actuating the aileron deflection angle.
- The roll angle controller controls the roll angle by providing commands to the roll rate controller.
- The transition multiplexer chooses which roll angle reference to give to the roll angle controller based on the aircraft's distance from the ground track. The roll angle references are generated by the either the first cross-track controller or the heading controller.
- The first cross-track controller generates the roll angle command based on the aircraft's cross-track error from the ground track. The algorithm used to obtain the cross-track error will be described in section 6.1.
- The heading controller regulates the aircraft heading by providing roll angle commands.
- The second cross-track controller also regulates the cross-track error. However, it generates heading references to achieve this goal.

5.1.2 Model Predictive Control Overview

The model predictive control (MPC) controller uses Hildreth's Quadatric Programming as the optimiser to obtain the optimal control actions to apply to the aircraft. The MPC is operated in a multiple-input-multiple-output (MIMO) configuration to control both the airspeed and altitude simultaneously. The MPC does this by providing the climb rate reference to the climb rate controller and the thrust command to the motor. The advantage of this is that, since airspeed is coupled to climb rate and hence effects altitude, the MPC can generate the optimal control actions to minimise both state errors. The classical airspeed and climb rate controllers are designed independently of each other, and the effect of this is that any actions performed by one of the controllers will treated as a disturbance in the other. This decoupled design method would work as long as the airspeed is maintained. However, during the descent phase for landing, the airspeed tends to increase, which causes an oscillation in altitude when tracking the glideslope. The MPC would be able to reduce this oscillation as it would be able keep the airspeed in check while tracking the glideslope altitude references.

The MPC replaces the classical airspeed and altitude controllers to obtain a more precise landing. This is executed by selecting the appropriate sub-mode in the ground control station, which will then prompt the PX4 autopilot software to forward the MPC commands to the climb rate controller and the motor. The classical airspeed and altitude controllers will be designed and tested for the runway and moving platform landing scenarios, so that their results can serve as a baseline for the MPC. The baseline will determine if there is an improvement when using the MPC for the the landing scenarios. Besides serving as a baseline, the classical airspeed and altitude controllers will also take over control if the MPC on the Jetson Nano fails.

It was initially considered to let the MPC regulate the climb rate as well. However, this idea was rejected due to the MPC being too model dependent. It was instead decided to keep the classical climb rate controller so that uncertainties in the aircraft model can be abstracted from the MPC. This allows any deficiencies in the aircraft model in capturing the practical UAV's dynamics to be encapsulated by the lower level controllers. This gives the MPC the best chance of working in a real world environment where uncertainties exist.

5.2. Classical Controller Design

5.2.1 Longitudinal Controllers Design

5.2.1.1 Airspeed Controller

The airspeed controller controls the calibrated airspeed to follow the airspeed references received from the guidance system, by actuating the thrust command using feedback from

an airspeed sensor. The calibrated airspeed feedback on the practical vehicle is provided by the PX4 autopilot software which obtains this value by reading the differential pressure in the pitot tube. Figure 5.2 shows the airspeed controller architectural diagram,

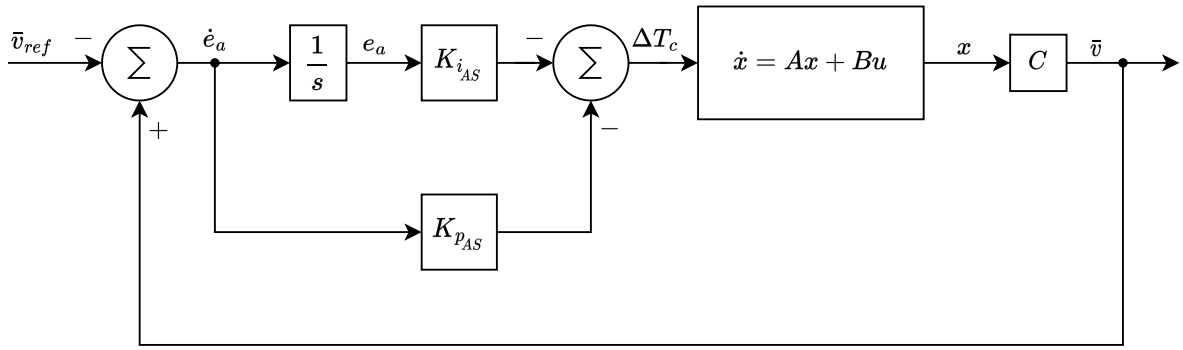


Figure 5.2: Block diagram of the airspeed controller.

where $\bar{v}_{ref}$ is the airspeed reference from trim, $\bar{v}$ is the airspeed from trim, ΔT_c is the thrust command deviation from trim, $K_{p_{AS}}$ is the proportional airspeed controller gain, $K_{i_{AS}}$ is the integral airspeed controller gain, s is the Laplace transform variable, A and B are the plant matrices, and C is the airspeed output matrix.

The longitudinal model in Equation 4.107 should ideally be used as the plant for the control design. However, since the airspeed controller can only regulate the airspeed deviation state ($\bar{v}$) in this model, the other longitudinal state variables would not be controlled, and would disturb the airspeed. To address this issue, a reduced-order model can be used that only considers the states that are relevant to the airspeed controller.

The motor trust has been previously modelled with equation 4.55, and assuming that the actual thrust follows the commanded thrust, this equation can be written as,

$$\dot{T} = -\frac{1}{\tau_e}T + \frac{1}{\tau_e}T_c \tag{5.1}$$

where T_c is the thrust command, T is the thrust magnitude and τ_e is the lag time constant of the motor. De Bruin [16] used a simplification from Peddle [9] to derived the linearised simplified velocity dynamics as,

$$\begin{bmatrix} \Delta\dot{T} \\ \dot{\bar{v}} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau_e} & 0 \\ \frac{1}{m} & 0 \end{bmatrix} \begin{bmatrix} \Delta T \\ \bar{v} \end{bmatrix} + \begin{bmatrix} \frac{1}{\tau_e} \\ 0 \end{bmatrix} \Delta T_c \tag{5.2}$$

where m is the mass of the UAV, and the states are deviations from trim including ΔT, which is the thrust magnitude from trim. This model represents the plant block in Figure 5.2.

A PI control architecture was chosen for the airspeed controller as shown in Figure 5.2. This was chosen due to the integrator being able to compensate for uncertainty in

steady-state drag and motor thrust offsets [9]. This is an important feature as the drag on the aircraft and the thrust produced by the motor both depend on the atmospheric conditions, which can vary due to numerous factors. The thrust test performed in appendix A.2 was done in the ESL laboratory where the conditions are ideal with no wind. The maximum thrust value was also conservatively chosen as the thrust produced by the motor also depends on battery voltage. The integrator compensates for these deviations, which makes it suitable for practical controller implementations. The disadvantage of using the integrator is that it does make the controller slower. However, this not a major issue as the primary function of the airspeed controller is to maintain a constant trim airspeed (18 m/s), which does not require a fast response.

The PI control law in figure 5.2 is defined as,

$$\Delta T_c = -K_{p_{AS}} \dot{e}_a - K_{i_{AS}} e_a \tag{5.3}$$

with,

$$\dot{e}_a = \bar{v} - \bar{v}_{ref} \tag{5.4}$$

where $\dot{e}_a$ is the airspeed error and e_a is the time integral of the airspeed error. The integrator term can be augmented into the dynamic model in equation 5.2 which results in,

$$\begin{bmatrix} \Delta\dot{T} \\ \dot{\bar{v}} \\ \dot{e}_a \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau_e} & 0 & 0 \\ \frac{1}{m} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta T \\ \bar{v} \\ e_a \end{bmatrix} + \begin{bmatrix} \frac{1}{\tau_e} \\ 0 \\ 0 \end{bmatrix} \Delta T_c + \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \bar{v}_{ref} \tag{5.5}$$

Performing the substitution of equation 5.3 and simplifying results in,

$$\begin{bmatrix} \Delta\dot{T} \\ \dot{\bar{v}} \\ \dot{e}_a \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau_e} & -\frac{K_{p_{AS}}}{\tau_e} & -\frac{K_{i_{AS}}}{\tau_e} \\ \frac{1}{m} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta T \\ \bar{v} \\ e_a \end{bmatrix} + \begin{bmatrix} \frac{K_{p_{AS}}}{\tau_e} \\ 0 \\ -1 \end{bmatrix} \bar{v}_{ref} \tag{5.6}$$

The close-loop characteristic equation can be obtained as,

$$p(s) = s^3 + \frac{1}{\tau_e} s^2 + \frac{K_{p_{AS}}}{m\tau_e} s + \frac{K_{i_{AS}}}{m\tau_e} \tag{5.7}$$

The desired close-loop poles can then be placed using,

$$\alpha_c(s) = (s + a)(s^2 + 2\zeta\omega_n s + \omega_n^2) \tag{5.8}$$

Coefficient matching is used with two equations to give the airspeed controller gains as,

$$K_{i_{AS}} = m\tau_e\omega_n^2 a \tag{5.9}$$

$$K_{p_{AS}} = m\tau_e(2\zeta\omega_n a + \omega_n^2) \tag{5.10}$$

where the natural frequency is,

$$\omega_n = \frac{\frac{1}{\tau_e} - a}{2\zeta} \tag{5.11}$$

To obtain the gains, the damping ratio ζ and the close-loop integrator pole a must be chosen. The values chosen for these parameters will effect the step response of the controller. This step response has to meet the requirements obtained from Peddle [8] and De Bruin [16] which are,

- Rise time of less than 3s
- Overshoot of less than 20%
- Zero steady state error

The damping ratio and integrator pole location were manually altered until a desirable step response was obtained that satisfied these requirements. The resulting values were found to be,

$$\zeta_{cl} = 0.85 \tag{5.12}$$

$$a = 0.65\text{rad/s} \tag{5.13}$$

Substituting these values into the gain equations (Equations 5.9 and 5.10) with this project's aircraft specifications (from Appendix A), and calculating the gains results in,

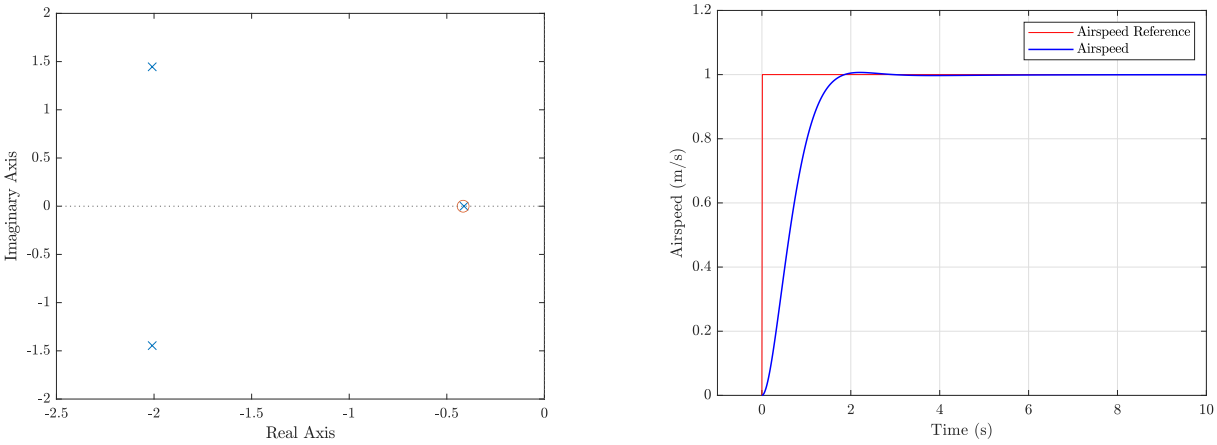
$$K_{i_{AS}} = 3.7136 \tag{5.14}$$

$$K_{p_{AS}} = 8.9168 \tag{5.15}$$

Figure 5.3 shows the pole zero plot and step response of the airspeed controller when using these gains.

The airspeed controller introduces a zero near the dominant pole, as shown in figure 5.3a, which causes an overshoot in the step response, therefore ζ and a had to be chosen carefully to ensure the requirements were met. The airspeed controller step response in figure 5.3b is well within the requirements for the controller. The response has a rise time of 1.26s, an overshoot of 0.7%, zero steady state error tracking and a 2% settling time of 1.63s.

The airspeed controller was designed with a reduced-order model. However, it needs to be added to the full longitudinal linear model in equation 4.107 so that the remaining



(a) Pole-Zero Plot **(b)** Step Response

Figure 5.3: Airspeed controller pole-zero plot and step response using the reduced-order model

longitudinal controllers can be designed taking into account the effect of the airspeed controller. The full longitudinal linear model does not include the thrust command needed by airspeed controller therefore the thrust dynamics from equation 5.1 is augmented into the model which results in,

$$\begin{bmatrix} \dot{\bar{v}} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \\ \Delta\dot{T} \end{bmatrix} = \begin{bmatrix} \frac{\partial \dot{U}}{\partial U} & \overline{V}_T \frac{\partial \dot{U}}{\partial W} & \frac{\partial \dot{U}}{\partial Q} & \frac{\partial \dot{U}}{\partial \Theta} & \frac{\partial \dot{U}}{\partial T} \\ \frac{1}{\overline{V}_T}\frac{\partial \dot{W}}{\partial U} & \frac{\partial \dot{W}}{\partial W} & \frac{1}{\overline{V}_T}\frac{\partial \dot{W}}{\partial Q} & \frac{1}{\overline{V}_T}\frac{\partial \dot{W}}{\partial \Theta} & \frac{1}{\overline{V}_T}\frac{\partial \dot{W}}{\partial T} \\ \frac{\partial \dot{Q}}{\partial U} & \overline{V}_T \frac{\partial \dot{Q}}{\partial W} & \frac{\partial \dot{Q}}{\partial Q} & \frac{\partial \dot{Q}}{\partial \Theta} & \frac{\partial \dot{Q}}{\partial T} \\ \frac{\partial \dot{\Theta}}{\partial U} & \overline{V}_T \frac{\partial \dot{\Theta}}{\partial W} & \frac{\partial \dot{\Theta}}{\partial Q} & \frac{\partial \dot{\Theta}}{\partial \Theta} & \frac{\partial \dot{\Theta}}{\partial T} \\ 0 & 0 & 0 & 0 & -\frac{1}{\tau_e} \end{bmatrix} \begin{bmatrix} \bar{v} \\ \alpha \\ q \\ \theta \\ \Delta T \end{bmatrix} + \begin{bmatrix} \frac{\partial \dot{U}}{\partial \delta_E} & \frac{\partial \dot{U}}{\partial \delta_F} & 0 \\ \frac{1}{\overline{V}_T}\frac{\partial \dot{W}}{\partial \delta_E} & \frac{1}{\overline{V}_T}\frac{\partial \dot{W}}{\partial \delta_F} & 0 \\ \frac{\partial \dot{Q}}{\partial \delta_E} & \frac{\partial \dot{Q}}{\partial \delta_F} & 0 \\ \frac{\partial \dot{\Theta}}{\partial \delta_E} & \frac{\partial \dot{\Theta}}{\partial \delta_F} & 0 \\ 0 & 0 & \frac{1}{\tau_e} \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_f \\ \Delta T_c \end{bmatrix} \tag{5.16}$$

Writing this equation in a compact state space form results in,

$$\dot{\boldsymbol{x}}_{Long2} = \boldsymbol{A}_{Long2}\boldsymbol{x}_{Long2} + \boldsymbol{B}_{Long2}\boldsymbol{u}_{Long2} \tag{5.17}$$

where $\boldsymbol{A}_{Long2}$ and $\boldsymbol{B}_{Long2}$ are the augmented system and input matrices respectively. The controller integrator state is augmented into this matrix and then the control law from equation 5.3 is substituted in to give the closed-loop system as,

$$\dot{\boldsymbol{x}}_{AS} = \boldsymbol{A}_{AS}\boldsymbol{x}_{AS} + \boldsymbol{B}_{AS}\boldsymbol{u}_{AS} \tag{5.18}$$

with,

$$\bar{v} = \boldsymbol{C}_{AS}\boldsymbol{x}_{AS} \tag{5.19}$$

where $\boldsymbol{A}_{AS}$ and $\boldsymbol{B}_{AS}$ are the system and input matrices augmented with the airspeed controller. The full derivation of the closed-loop system was performed by De Bruin [16]. The closed-loop transfer function which relates the airspeed reference to airspeed is given

as,

$$\frac{\bar{v}(s)}{\bar{v}_{ref}(s)} = \boldsymbol{C}_{AS}(s\boldsymbol{I} - \boldsymbol{A}_{AS})^{-1}\boldsymbol{B}_{\bar{v}_{ref}} \tag{5.20}$$

where,

$$\boldsymbol{C}_{AS} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \; ; \; \boldsymbol{B}_{\bar{v}_{ref}} = \boldsymbol{B}_{AS}\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\mathsf{T}} \tag{5.21}$$

5.2.1.2 Normal Specific Acceleration Direct Lift Control Controller

The normal specific acceleration direct lift (NSADLC) controller controls the normal acceleration by commanding the elevator and flap deflections based on references it receives from the climb rate controller. The NSADLC controller is based on the design by De Bruin [16] who achieved high precision runway landing performance using this controller. De Bruin added the direct lift control (DLC) component to the standard normal specific acceleration (NSA) controller, which improved the step response performance. The normal acceleration feedback on the practical vehicle is obtained from a 3-axis accelerator which gives the acceleration in the body axes. As shown in Figure 5.4, the NSADLC controller is formed using two separate controllers, which are the NSA controller and the DLC controller.

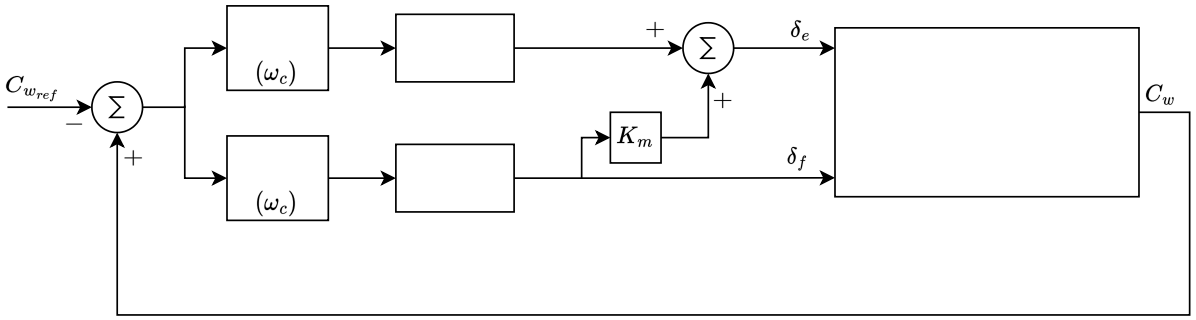


Figure 5.4: Block diagram of entire NSADLC controller.

This architecture constrains the NSA controller to only operate on the low frequency components of the error signal, while the DLC controller only operates on the high frequency components. The filter centre frequency ω_c is chosen to be close to the closed-loop bandwidth of the NSA controller. This provides two advantages which are: the removal of the low pass filter (LPF) for the NSA controller, and being able to design the NSA and DLC controllers independently of each other. The K_m gain is used to allow the elevator to cancel out undesired pitching moments produced by the flap deflections.

DLC Controller Design

Figure 5.5 shows the block diagram of the DLC controller.

A reduced-order model, obtained from De Bruin [16], is used to represent the DLC plant, which describes the effect of the flaps (δ_f) on the angle of attack (α) and pitch rate

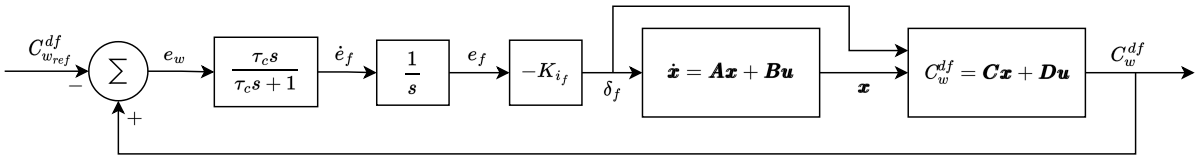


Figure 5.5: Block diagram of DLC controller.

deviations from trim (q). This model is given as,

$$\begin{bmatrix}\dot{\alpha}\\ \dot{q}\end{bmatrix} = \begin{bmatrix}-\frac{L_\alpha}{m\overline{V}_T} & 1\\ \frac{M_\alpha}{I_{yy}} & \frac{M_Q}{I_{yy}}\end{bmatrix}\begin{bmatrix}\alpha\\ q\end{bmatrix} + \begin{bmatrix}-\frac{L_{\delta_F}}{m\overline{V}_T}\\ 0\end{bmatrix}\delta_f \tag{5.22}$$

$$C^{df}_w = \begin{bmatrix}-\frac{L_\alpha}{m} & 0\end{bmatrix}\begin{bmatrix}\alpha\\ q\end{bmatrix} + \begin{bmatrix}-\frac{L_{\delta_F}}{m}\end{bmatrix}\delta_f \tag{5.23}$$

where C^{df}_w is the normal specific acceleration produced by the DLC controller in the wind axis system, I_{yy} is the principle moment of inertia about the y-axis, and $\overline{V}_T$ is the trim airspeed. The $L_\alpha, L_Q, L_{\delta_F}, M_\alpha$ and M_Q are stability and control derivatives which depend on the aircraft's characteristics. The equations for these derivatives can be found in appendix A.3.3.

The DLC controller uses an integral control law as shown in figure 5.5 and the equations that define this law are written as,

$$\delta_f = -K_{i_f} e_f \tag{5.24}$$

$$\dot{e}_f = \frac{\tau_c s}{\tau_c s + 1}\, e_w \tag{5.25}$$

$$e_w = C^{df}_w - C^{df}_{w_{ref}} \tag{5.26}$$

where e_w is the NSA error, $\dot{e}_f$ is the high-pass-filtered NSA error, e_f is the time integral of the high-pass-filtered NSA error, T_c is the HPF time constant and $C^{df}_{w_{ref}}$ is the normal specific acceleration reference to the DLC controller. The reduced-order model in equation 5.22 is augmented with the DLC controller integrator state and the control law. Thereafter, the closed-loop characteristic equation is obtained to perform coefficient matching with the desired close-loop pole equation, so that the integrator gain equation can be calculated. The gain equation is calculated to be,

$$K_{i_f} = -\frac{m}{L_{\delta_F}}\left(2\zeta\omega_n + a - \frac{L_\alpha}{\overline{V}_T m} + \frac{M_Q}{I_{yy}} - \frac{1}{\tau_c}\right) \tag{5.27}$$

where ζ is the closed-loop damping ratio, w_n is closed-loop natural frequency, and a is the closed-loop integrator pole location.

To calculate the filter time constant, the NSA controller's natural frequency needs to

be determined. The natural frequency is chosen to be close to the non-minimum phase (NMP) upper bound which is calculated using the formula,

$$\omega_n < \frac{1}{3}\left|\sqrt{\frac{L_\alpha}{I_{yy}}(l_T - l_N)}\right| \tag{5.28}$$

where l_T and l_N are the effective lengths from the CG location to the tailplane and neutral point respectively. These lengths are calculated using,

$$l_T \equiv -\frac{M_{\delta_E}}{L_{\delta_E}} \tag{5.29}$$

$$l_N \equiv -\frac{M_\alpha}{L_\alpha} \tag{5.30}$$

where $L_\alpha, L_{\delta_E}, M_\alpha$ and M_{δ_E} are stability and control derivatives whose equations can be found in appendix A.3.3. Substituting the aircraft constants into these equations produces the NMP upper bound as,

$$\omega_{n_{max}} < 8.6073 \text{ rad/s} \tag{5.31}$$

The NSA closed-loop natural frequency is therefore chosen as,

$$\omega_{n_{pmb}} = 8.5 \text{ rad/s} \tag{5.32}$$

The filter centre frequency is chosen to be equal to the closed-loop bandwidth of the NSA controller, as previously discussed, which results in,

$$\omega_c = \omega_{n_{pmb}} = 8.5 \text{ rad/s} \tag{5.33}$$

and therefore the filter time constant is calculated as,

$$\tau_c = \frac{1}{\omega_c} = 0.1176 \tag{5.34}$$

The DLC controller damping ratio and natural frequency were set equal to those of the aircraft's natural short period mode, and then the DLC controller closed-loop integrator pole location was calculated as,

$$\zeta_{cl} = \zeta_{sp} = 0.6731 \tag{5.35}$$

$$\omega_{n_{cl}} = \omega_{n_{sp}} = 8.0197 \text{ rad/s} \tag{5.36}$$

$$a = 2\zeta_{cl}\,\omega_{n_{cl}} = 10.7961 \text{ rad/s} \tag{5.37}$$

These values can then be substituted into equation 5.27 to calculate the DLC controller

gain as,

$$K_{i_f} = -0.1131 \tag{5.38}$$

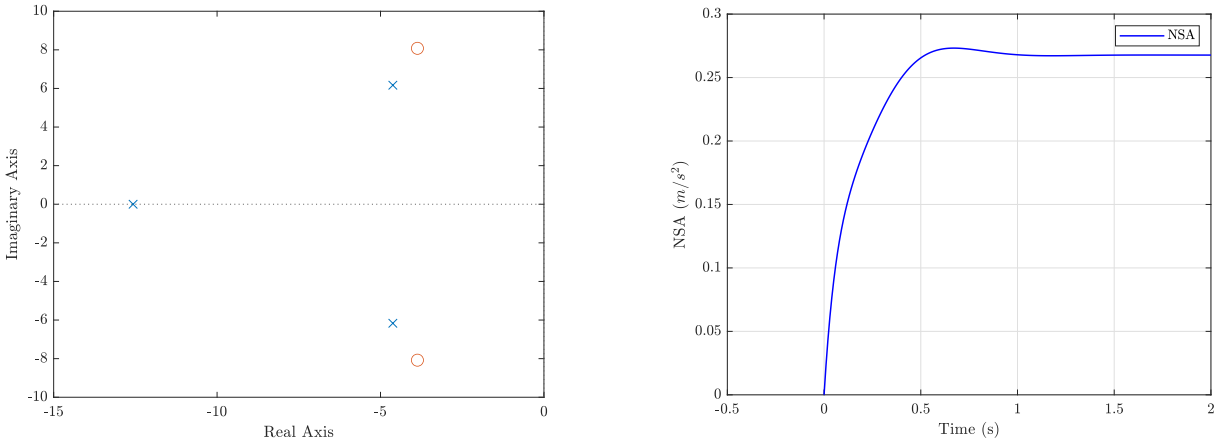
The K_m gain can be calculated using the ratio between the pitching moment flap deflection coefficient and the pitching moment elevator deflection coefficient which is written as,

$$K_m = -\frac{C_{m_{\delta_F}}}{C_{m_{\delta_E}}} \tag{5.39}$$

giving,

$$K_m = 0.1213 \tag{5.40}$$

Figure 5.6 shows the pole-zero plot and step response of the DLC controller for the calculated gains. The step response's magnitude is the normal specific acceleration produced by the flap deflection caused by the DLC controller. The step response reference has a magnitude of 1.0. In Figure 5.6a, the closed-loop integrator pole was chosen to



(a) Pole-Zero Plot **(b)** Step Response

Figure 5.6: DLC controller pole-zero plot and step response using the reduced order model

be about double the real component of the short period mode poles for high bandwidth. The short period mode poles are kept close to their open-loop location as they will be manipulated by the NSA controller. The DLC step response in Figure 5.6b has a large steady state error due to the close-loop system not containing a free integrator. However, this is not of concern since the NSA controller will handle the low frequency components of the NSA error signal. The DLC controller is mainly used to increase the initial response time of the NSA state.

NSA Controller Design

The NSA Controller block diagram is shown in Figure 5.7.

The NSA controller also uses a reduced-order dynamic model, obtained from Peddle [9],

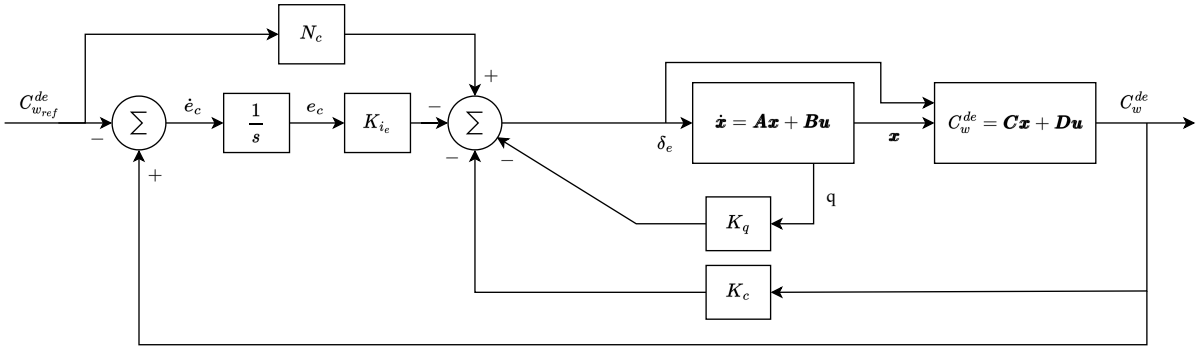


Figure 5.7: Block diagram of NSA controller.

which relates the effect of the elevator deflection (δ_e) on the angle of attack (α) and pitch rate deviations from trim (q). The model is derived as,

$$\begin{bmatrix}\dot{\alpha}\\ \dot{q}\end{bmatrix} = \begin{bmatrix}-\frac{L_\alpha}{m\bar{V}_T} & 1\\ \frac{M_\alpha}{I_{yy}} & \frac{M_Q}{I_{yy}}\end{bmatrix}\begin{bmatrix}\alpha\\ q\end{bmatrix} + \begin{bmatrix}0\\ \frac{M_{\delta_E}}{I_{yy}}\end{bmatrix}\delta_e \tag{5.41}$$

$$C^{de}_w = \begin{bmatrix}-\frac{L_\alpha}{m} & 0\end{bmatrix}\begin{bmatrix}\alpha\\ q\end{bmatrix} + \begin{bmatrix}0\end{bmatrix}\delta_e \tag{5.42}$$

where C^{de}_w is the normal specific acceleration produced by the NSA controller in the wind axis system. M_{δ_E} and L_{δ_E} (used in closed-loop derivation) are stability and control derivatives which depend on the aircraft's characteristics. The formula for these derivatives can be found in appendix A.3.3.

The PI control law, shown in figure 5.7, was derived by Peddle [9] as,

$$\delta_e = -K_q\, q - K_c\, C^{de}_w - K_{i_e}\, e_c + N_c\, C^{de}_{w_{ref}} \tag{5.43}$$

$$\dot{e}_c = C^{de}_w - C^{de}_{w_{ref}} \tag{5.44}$$

where $\dot{e}_c$ is the NSA error, e_c is the time integral of the NSA error, and $C^{de}_{w_{ref}}$ is the normal specific acceleration reference for the NSA controller. The flap mixing component with the K_m gain is ignored for the NSA control law design to allow for the independent controller design, and it is instead treated as a disturbance. The feedforward gain N_c is included in the control law by placing a zero at,

$$s = -\frac{K_{i_e}}{N_c} \tag{5.45}$$

which is placed close to the closed-loop integrator pole to minimise its transient effects. The reduced-order model in equation 5.41 is augmented with the NSA integrator state and control law, and thereafter the closed-loop characteristic equation is obtained. This equation is then used for coefficient matching with the desired close-loop pole equation so

that the NSA controller gains can be calculated resulting in,

$$K_q = \frac{I_{yy}}{M_{\delta_E}}\left(2\zeta\omega_n + a + \frac{M_Q}{I_{yy}} - \frac{L_\alpha}{m\overline{V}_T}\right) \tag{5.46}$$

$$K_c = -\frac{mI_{yy}}{L_\alpha M_{\delta_E}}\left(2\zeta\omega_n a + \omega_n^2 + \frac{M_\alpha}{I_{yy}} - \frac{L_\alpha}{m\overline{V}_T}\left(2\zeta\omega_n + a - \frac{L_\alpha}{m\overline{V}_T}\right)\right) \tag{5.47}$$

$$K_{i_e} = -\frac{mI_{yy}}{L_\alpha M_{\delta_E}}\left(\omega_n^2 a\right) \tag{5.48}$$

To obtain these gains, the closed-loop integrator pole location a, as well as the short period mode poles' damping ratio ζ and natural frequency ω_n must be selected. The feedforward zero z_f also needs to be selected to calculate the feedforward gain N_c. The closed-loop damping ratio is chosen for optimal damping and is therefore selected as,

$$\zeta_{cl} = 0.707 \tag{5.49}$$

The NSA closed-loop natural frequency was previously selected in equation 5.32 and is restated as,

$$\omega_{n_{cl}} = 8.5 \text{ rad/s} \tag{5.50}$$

The closed-loop integrator location is obtained as,

$$a = \zeta_{cl}\,\omega_{n_{cl}} = 6.0095 \text{ rad/s} \tag{5.51}$$

The feedforward zero is selected to be at the closed-loop integrator pole location,

$$z_f = a = 6.0095 \text{ rad/s} \tag{5.52}$$

The NSA gains are then calculated by using these values, resulting in

$$K_q = -0.0762 \tag{5.53}$$

$$K_c = 0.0069 \tag{5.54}$$

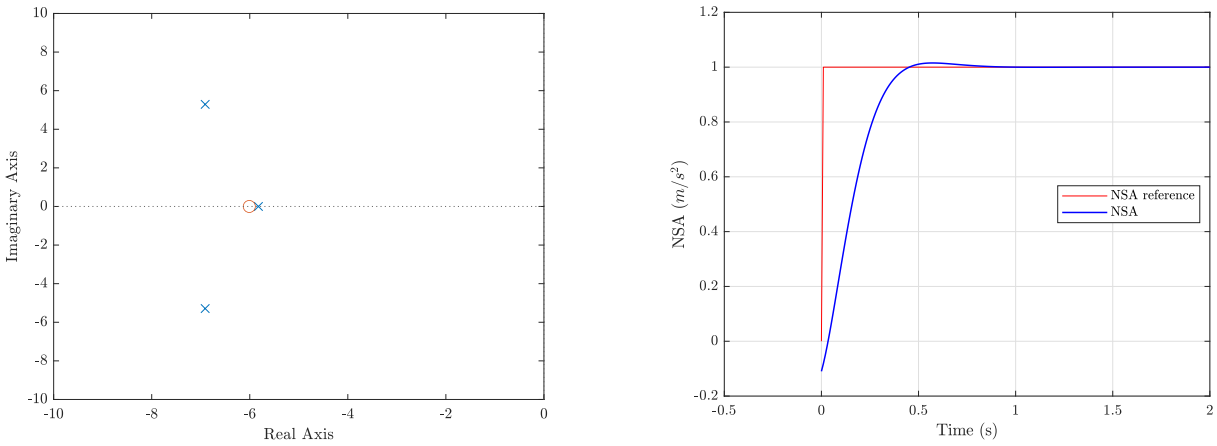
$$K_{i_e} = 0.0623 \tag{5.55}$$

The feedforward gain is calculated using equation 5.45 which yields,

$$N_c = \frac{-K_{i_e}}{-z_f} = 0.0104 \tag{5.56}$$

The pole-zero plot and step response for these gains are shown in figure 5.8.

As shown in the pole-zero plot, the short period mode poles are adequately damped and are placed close to the NMP upper bound (8.6073 rad/s). The integrator pole location



(a) Pole-Zero Plot **(b)** Step Response

Figure 5.8: NSA controller pole-zero plot and step response using the reduced order model

is placed close to the short period mode poles as to not constrain the NSA controller bandwidth. The zero is placed by the integrator to reduce its negative transient effects. The NSA step response has a very fast rise time and almost no overshoot indicating sufficient damping. The initial negative value of the response is believed to be cause by approximations in the model.

NSADLC Controller Closed-loop System

To allow the outer controllers to be designed in a cascaded configuration, the NSADLC controller needs to be augmented into the full linear longitudinal model. The airspeed controller was already augmented into the model, therefore its closed-loop model from equation 5.18 can be used as the plant for the NSADLC controller and it is restated as,

$$\dot{\boldsymbol{x}}_{AS} = \boldsymbol{A}_{AS}\boldsymbol{x}_{AS} + \boldsymbol{B}_{AS}\boldsymbol{u}_{AS} \tag{5.57}$$

The DLC integrator state can be augmented into this model to form,

$$\begin{bmatrix} \dot{\boldsymbol{x}}_{AS} \\ \dot{e}_f \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{AS} & & & & \boldsymbol{0}_{6\times1} \\ 0 & -\frac{L_\alpha}{m} & -\frac{L_Q}{m} & \boldsymbol{0}_{1\times3} & -\frac{1}{\tau_c} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{AS} \\ e_f \end{bmatrix} + \begin{bmatrix} & \boldsymbol{B}_{AS} & \\ 0 & -\frac{L_{\delta_F}}{m} & 0 \end{bmatrix} \boldsymbol{u}_{AS} + \begin{bmatrix} \boldsymbol{0}_{6\times1} \\ -1 \end{bmatrix} C^{df}_{w_{ref}} \tag{5.58}$$

The $\boldsymbol{A}$ matrix in this equation can be represented more compactly as $\boldsymbol{A}_{\dot{e}_f}$. Substituting the DLC control law from equation 5.24 into this equation while simplifying results in,

$$\begin{bmatrix} \dot{\boldsymbol{x}}_{AS} \\ \dot{e}_f \end{bmatrix} = \left[\boldsymbol{A}_{\dot{e}_f} + \begin{bmatrix} \boldsymbol{B}_{\delta_f} \\ -\frac{L_{\delta_F}}{m} \end{bmatrix} \begin{bmatrix} \boldsymbol{0}_{1\times6} & -K_{i_f} \end{bmatrix} \right] \begin{bmatrix} \boldsymbol{x}_{AS} \\ e_f \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_{\delta_e} & \boldsymbol{0}_{6\times1} & \boldsymbol{B}_{\bar{v}_{ref}} \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \delta_e \\ C^{df}_{w_{ref}} \\ \bar{v}_{ref} \end{bmatrix} \tag{5.59}$$

where,

$$\boldsymbol{B}_{\delta_e} = \boldsymbol{B}_{AS}\begin{bmatrix}1 & 0 & 0\end{bmatrix}^\intercal \;;\; \boldsymbol{B}_{\delta_f} = \boldsymbol{B}_{AS}\begin{bmatrix}0 & 1 & 0\end{bmatrix}^\intercal \;;\; \boldsymbol{B}_{\bar{v}_{ref}} = \boldsymbol{B}_{AS}\begin{bmatrix}0 & 0 & 1\end{bmatrix}^\intercal \tag{5.60}$$

This equation can be written in a compact state space form as,

$$\dot{\boldsymbol{x}}_{DLC} = \boldsymbol{A}_{DLC}\boldsymbol{x}_{DLC} + \boldsymbol{B}_{DLC}\boldsymbol{u}_{DLC} \tag{5.61}$$

with,

$$C_w^{df} = \boldsymbol{C}_{DLC}\boldsymbol{x}_{DLC} \tag{5.62}$$

The close-loop transfer function which relates the DLC controller NSA reference to the DLC controller output is calculated as,

$$\frac{C_w^{df}(s)}{C_{w_{ref}}^{df}(s)} = \boldsymbol{C}_{DLC}(s\boldsymbol{I} - \boldsymbol{A}_{DLC})^{-1}\boldsymbol{B}_{C_{w_{ref}}}^{df} \tag{5.63}$$

where,

$$\boldsymbol{C}_{DLC} = \begin{bmatrix}0 & -\frac{L_\alpha}{m} & -\frac{L_Q}{m} & 0 & 0 & 0 & \frac{K_{i_f}L_{\delta_F}}{m}\end{bmatrix} \;;\; \boldsymbol{B}_{C_{w_{ref}}}^{df} = \boldsymbol{B}_{DLC}\begin{bmatrix}0 & 1 & 0\end{bmatrix}^\intercal \tag{5.64}$$

The DLC controller has now been augmented into the full longitudinal model. All that remains is to add the NSA controller to form the full hybrid NSADLC closed-loop model. The NSA controller integrator state is augmented into the DLC state space equation(5.61) which results in,

$$\begin{aligned}\begin{bmatrix}\dot{\boldsymbol{x}}_{DLC}\\ \dot{e}_c\end{bmatrix} = &\begin{bmatrix}\boldsymbol{A}_{DLC} & & & & & \boldsymbol{0}_{7\times1}\\ 0 & -\frac{L_\alpha}{m} & -\frac{L_Q}{m} & \boldsymbol{0}_{1\times3} & \frac{K_{i_f}L_{\delta_F}}{m} & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_{DLC}\\ e_c\end{bmatrix}\\ &+\begin{bmatrix}\boldsymbol{B}_{DLC} & & \\ -\frac{L_{\delta_E}}{m} & 0 & 0\end{bmatrix}\boldsymbol{u}_{DLC} + \begin{bmatrix}\boldsymbol{0}_{7\times1}\\ -1\end{bmatrix}C_{w_{ref}}^{de}\end{aligned} \tag{5.65}$$

The $\boldsymbol{A}$ matrix in this equation can be represented in a more compact form as $\boldsymbol{A}_{\dot{e}_c}$. Assuming that the input NSA reference to the NSA and DLC controllers are the same, the NSA control law (5.43) can be substituted into this equation to give,

$$\begin{aligned}\begin{bmatrix}\dot{\boldsymbol{x}}_{DLC}\\ \dot{e}_c\end{bmatrix} = &\left[\boldsymbol{A}_{\dot{e}_c} + \begin{bmatrix}\boldsymbol{B}_{\delta_e}\\ -\frac{L_{\delta_E}}{m}\end{bmatrix}\begin{bmatrix}0 & \frac{K_cL_\alpha}{m-K_cL_{\delta_E}} & \frac{K_cL_Q-mK_q}{m-K_cL_{\delta_E}} & \boldsymbol{0}_{1\times3} & -\frac{K_cK_{i_f}L_{\delta_F}}{m-K_cL_{\delta_E}} & -\frac{mK_{i_e}}{m-K_cL_{\delta_E}}\end{bmatrix}\right]\begin{bmatrix}\boldsymbol{x}_{DLC}\\ e_c\end{bmatrix}\\ &+\begin{bmatrix}\left(\frac{mN_c}{m-K_cL_{\delta_E}}\boldsymbol{B}_{\delta_e} + \boldsymbol{B}_{C_{w_{ref}}}^{df}\right) & \boldsymbol{B}_{\bar{v}_{ref}}\\ -\left(1+\frac{N_cL_{\delta_E}}{m-K_cL_{\delta_E}}\right) & 0\end{bmatrix}\begin{bmatrix}C_{w_{ref}}\\ \bar{v}_{ref}\end{bmatrix}\end{aligned} \tag{5.66}$$

where,

$$\boldsymbol{B}_{\delta_e} = \boldsymbol{B}_{DLC}\begin{bmatrix}1 & 0 & 0\end{bmatrix}^\intercal \; ; \; \boldsymbol{B}^{df}_{C_{w_{ref}}} = \boldsymbol{B}_{DLC}\begin{bmatrix}0 & 1 & 0\end{bmatrix}^\intercal \; ; \; \boldsymbol{B}_{\bar{v}_{ref}} = \boldsymbol{B}_{DLC}\begin{bmatrix}0 & 0 & 1\end{bmatrix}^\intercal \quad (5.67)$$

This equation represents the closed-model of the full NSADLC hybrid system. This model can be written in a compact state space form as,

$$\dot{\boldsymbol{x}}_{Hyb} = \boldsymbol{A}_{Hyb}\boldsymbol{x}_{Hyb} + \boldsymbol{B}_{Hyb}\boldsymbol{u}_{Hyb} \quad (5.68)$$

and,

$$C_w = \boldsymbol{C}_{Hyb}\boldsymbol{x}_{Hyb} + \boldsymbol{D}_{Hyb}\boldsymbol{u}_{Hyb} \quad (5.69)$$

where,

$$\boldsymbol{C}_{Hyb} = \begin{bmatrix}\boldsymbol{C}_{DLC} & 0\end{bmatrix} - \frac{L_{\delta_E}}{m}\begin{bmatrix}0 & \frac{K_c L_\alpha}{m - K_c L_{\delta_E}} & \frac{K_c L_Q - mK_q}{m - K_c L_{\delta_E}} & \boldsymbol{0}_{1\times 3} & -\frac{K_c K_{i_f} L_{\delta_F}}{m - K_c L_{\delta_E}} & -\frac{mK_{i_e}}{m - K_c L_{\delta_E}}\end{bmatrix} \quad (5.70)$$

$$\boldsymbol{D}_{Hyb} = \begin{bmatrix}-\frac{N_c L_{\delta_E}}{m - K_c L_{\delta_E}} & 0\end{bmatrix} \quad (5.71)$$

This model can be used to form the plant of the outer-loop controllers, and their control laws can be augmented into this model. The closed-loop transfer function, which relates the NSA reference to the NSA output of the entire NSADLC controller, can be calculated as,

$$\frac{C_w(s)}{C_{w_{ref}}(s)} = \boldsymbol{C}_{Hyb}(s\boldsymbol{I} - \boldsymbol{A}_{Hyb})^{-1}\boldsymbol{B}_{C_{w_{ref}}} + \boldsymbol{D}_{Hyb} \quad (5.72)$$

where,

$$\boldsymbol{B}_{C_{w_{ref}}} = \boldsymbol{B}_{Hyb}\begin{bmatrix}1 & 0\end{bmatrix}^\intercal \quad (5.73)$$

The step response of the NSADLC closed-loop model, shown in Figure 5.9, has a very fast rise time and minimal overshoot. The overshoot period and hence the 2% settling time of the NSADLC controller is longer then the standard NSA controller on the reduced-order dynamics (Figure 5.8b). This is expected as the NSADLC controller operates on the full longitudinal dynamics. Its performance is still within specification. The fast rise time of the NSADLC controller is very important for it to not limit the performance of the outer (climb rate and altitude) controllers, which use it as an inner-loop controller.

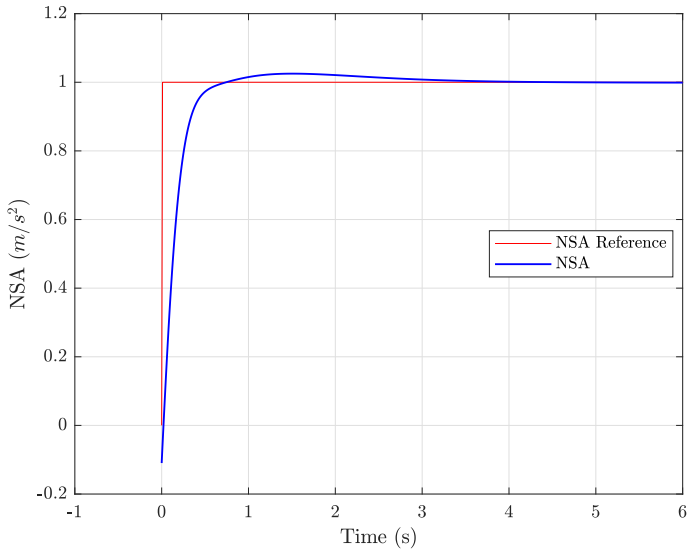


Figure 5.9: Step response of the full NSADLC controller.

5.2.1.3 Climb Rate Controller

The climb rate controller controls the aircraft's climb rate state by commanding the NSA reference for the NSADLC controller based on climb rate references it receives from the altitude controller. The climb rate is defined as the aircraft's upwards velocity in the inertial frame. This means that the climb rate is positive in the upwards direction, which is the opposite direction to the Z-axis (down axis) of the inertial frame. On the practical vehicle, the aircraft's inertial velocity is obtained from the Extended Kalman Filter (EKF) which uses the IMU's data combined with corrections obtained from the DGPS. The climb rate can be obtained from the EKF data by simply multiplying the Z inertial velocity component by -1 which is mathematically expressed as,

$$\dot{h} = -\dot{D} \tag{5.74}$$

The block diagram outlining the climb rate architecture is shown in figure 5.10.

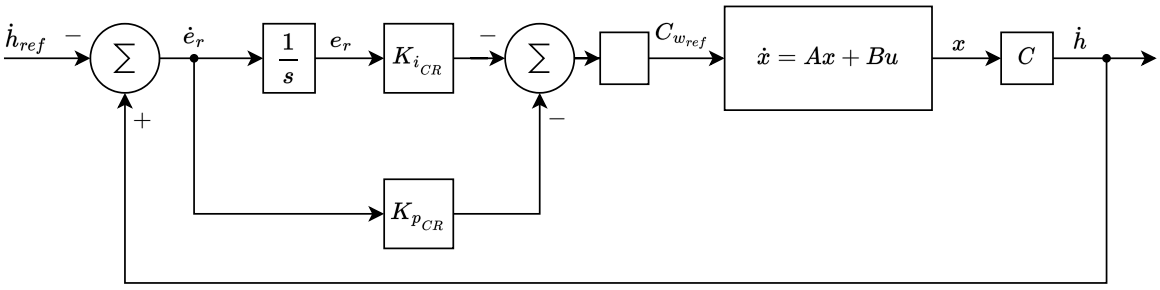


Figure 5.10: Block diagram of the climb rate controller.

$\dot{h}_{ref}$ is the climb rate reference, $\dot{h}$ is the climb rate, and $C_{w_{ref}}$ is the NSA reference command. The -1 block converts the climb rate controller command to the down axis used by the NSADLC controller. The climb rate controller uses a PI control law as the

integrator is used to compensate for biases in the normal acceleration measurements. The climb rate controller can be designed on the full longitudinal model dynamics which allows the climb rate plant to use the NSADLC closed-loop model from equation 5.68. To obtain the climb rate plant, the relationship between the NSA reference and climb rate has to be derived. This requires the climb rate to be extracted from the NSADLC closed-loop state vector $\boldsymbol{x}_{Hyb}$. The downwards inertial velocity ($\dot{D}$) is related to the body axis velocities via the inverse DCM as,

$$\dot{D} = -U\sin(\Theta) + V\cos(\Theta)\,\sin(\Phi) + W\cos(\Theta)\,\cos(\Phi) \tag{5.75}$$

The climb rate is related to the inertial downwards velocity via equation 5.74. Therefore, using small angle approximations, and a roll angle (Φ) of zero, the climb rate can be written as,

$$\dot{h} = -\dot{D} = U\Theta - W \tag{5.76}$$

During straight and level flight it can be assumed that $U \approx \overline{V}_T$ and $W \approx \overline{V}_T\,\alpha$ which allows the climb rate to be simplified to,

$$\dot{h} = \overline{V}_T(\Theta - \alpha) \tag{5.77}$$

This equation extracts the climb rate from the $\boldsymbol{x}_{Hyb}$ state vector. Therefore, the transfer function from the NSA reference to the climb rate state, which is the climb rate controller plant, is written as,

$$\frac{\dot{h}(s)}{C_{w_{ref}}(s)} = \boldsymbol{C}_{\dot{h}}(s\boldsymbol{I} - \boldsymbol{A}_{Hyb})^{-1}\boldsymbol{B}_{C_{w_{ref}}} \tag{5.78}$$

where,

$$\boldsymbol{C}_{\dot{h}} = \begin{bmatrix} 0 & -\overline{V}_T & 0 & \overline{V}_T & \boldsymbol{0}_{1\times 4} \end{bmatrix} \tag{5.79}$$

The PI control law for figure 5.10 is written as,

$$C_{w_{ref}} = -K_{p_{CR}}\dot{e}_r - K_{i_{CR}}e_r \tag{5.80}$$

with,

$$\dot{e}_r = \dot{h} - \dot{h}_{ref} \tag{5.81}$$

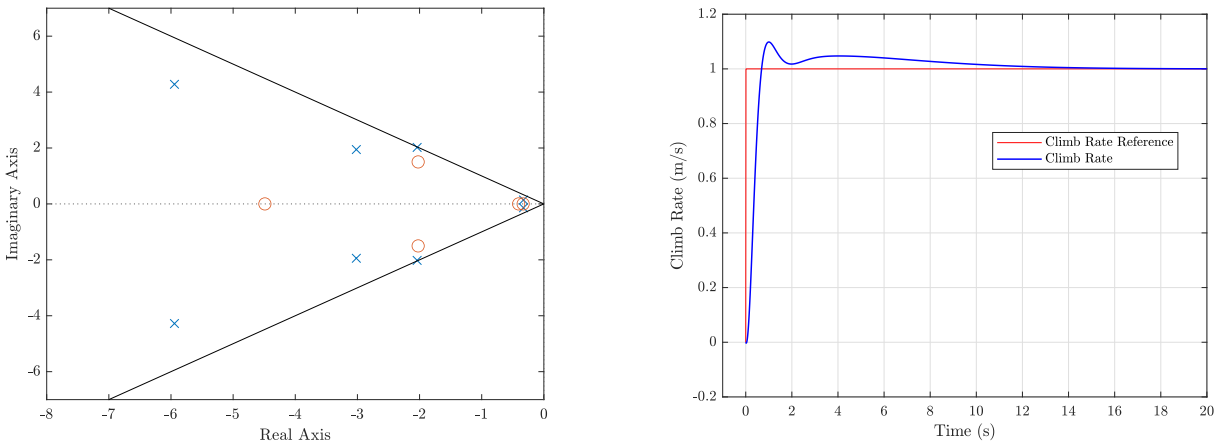
where $\dot{e}_r$ is the climb rate error and e_r is the time integral of the climb rate error. The climb rate controller gains are selected so that the dominant closed-loop poles are optimally damped, having a damping ratio (ζ) of 0.707. The gains are also selected to ensure the step response has a rise time of less than 3 seconds, an overshoot of less than 20% and zero steady-state error. The step response requirements were obtained from Peddle [8], who choose them based on what would be reasonable for conventional flight. The controller

gains selected that meet these requirements are,

$$K_{i_{CR}} = 0.9 \tag{5.82}$$

$$K_{p_{CR}} = 2.70 \tag{5.83}$$

Figure 5.11 shows the pole-zero plot and step response for the climb rate controller using the selected gains.



(a) Pole-Zero Plot **(b)** Step Response

Figure 5.11: Climb Rate controller pole-zero plot and step response

The two black lines in the pole-zero map represents the damping ratio (ζ) boundary at 0.707 and it can be seen that the dominant closed-loop poles are optimally damped. The step response has a fast rise time of 0.59 seconds, an overshoot of 10%, and zero steady-state error, which are within the requirements for the controller. The fast rise means that the climb rate controller should not hamper the outer altitude controller's response time. The step response has an undesirable second peak which is caused by the slower complex pole pair near the closed-loop zeros. This causes the 2% settling time to be much slower at 9.4 seconds. However, this is still acceptable as the outer altitude controller contains a limited integrator which would minimise this effect.

The closed-loop model for the climb rate controller can now be derived so it can be used for the altitude controller design. The NSADLC closed-loop model from Equation 5.68 can be augmented with the climb rate controller integrator to give,

$$\begin{bmatrix} \dot{\boldsymbol{x}}_{Hyb} \\ \dot{e}_r \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{Hyb} & & & & \boldsymbol{0}_{8\times1} \\ 0 & -\overline{V_T} & 0 & \overline{V_T} & \boldsymbol{0}_{1\times5} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{Hyb} \\ e_r \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_{Hyb} \\ \boldsymbol{0}_{1\times2} \end{bmatrix} \boldsymbol{u}_{Hyb} + \begin{bmatrix} \boldsymbol{0}_{8\times1} \\ -1 \end{bmatrix} \dot{h}_{ref} \tag{5.84}$$

The $\boldsymbol{A}$ matrix can be written more compactly as $\boldsymbol{A}_{\dot{e}_r}$ and the climb rate control law

(Equation 5.80) can be substituted into the equation to form,

$$\begin{bmatrix} \dot{\boldsymbol{x}}_{Hyb} \\ \dot{e}_r \end{bmatrix} = \left[\boldsymbol{A}_{\dot{e}_r} + \begin{bmatrix} \boldsymbol{B}_{C_{w_{ref}}} \\ 0 \end{bmatrix} \begin{bmatrix} 0 & K_{p_{CR}} \overline{V_T} & 0 & -K_{p_{CR}} \overline{V_T} & \boldsymbol{0}_{1\times 4} & -K_{i_{CR}} \end{bmatrix} \right] \begin{bmatrix} \boldsymbol{x}_{Hyb} \\ e_r \end{bmatrix} + \begin{bmatrix} K_{p_{CR}} \boldsymbol{B}_{C_{w_{ref}}} & \boldsymbol{B}_{\bar{v}_{ref}} \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \dot{h}_{ref} \\ \bar{v}_{ref} \end{bmatrix} \tag{5.85}$$

where,

$$\boldsymbol{B}_{C_{w_{ref}}} = \boldsymbol{B}_{Hyb} \begin{bmatrix} -1 & 0 \end{bmatrix}^\intercal \; ; \; \boldsymbol{B}_{\bar{v}_{ref}} = \boldsymbol{B}_{Hyb} \begin{bmatrix} 0 & 1 \end{bmatrix}^\intercal \tag{5.86}$$

The -1 multiplication used for the calculation of the $\boldsymbol{B}_{C_{w_{ref}}}$ vector is used to account for the opposite direction between the NSA and climb rate. Equation 5.85 can be written in a more compact form as,

$$\dot{\boldsymbol{x}}_{\dot{h}} = \boldsymbol{A}_{\dot{h}} \boldsymbol{x}_{\dot{h}} + \boldsymbol{B}_{\dot{h}} \boldsymbol{u}_{\dot{h}} \tag{5.87}$$

with,

$$\dot{h} = \boldsymbol{C}_{\dot{h}} \boldsymbol{x}_{\dot{h}} \tag{5.88}$$

where,

$$\boldsymbol{C}_{\dot{h}} = \begin{bmatrix} 0 & -\overline{V_T} & 0 & \overline{V_T} & \boldsymbol{0}_{1\times 5} \end{bmatrix} \tag{5.89}$$

The transfer function that relates the climb rate reference to climb rate is calculated as,

$$\frac{\dot{h}(s)}{\dot{h}_{ref}(s)} = \boldsymbol{C}_{\dot{h}} (s\boldsymbol{I} - \boldsymbol{A}_{\dot{h}})^{-1} \boldsymbol{B}_{\dot{h}_{ref}} \tag{5.90}$$

where,

$$\boldsymbol{B}_{\dot{h}_{ref}} = \boldsymbol{B}_{\dot{h}} \begin{bmatrix} 1 & 0 \end{bmatrix}^\intercal \tag{5.91}$$

5.2.1.4 Altitude Controller

The altitude controller controls the aircraft's altitude, based on the references it receives from the guidance system, by commanding the climb rate controller which itself commands the NSADLC controller. As with the climb rate, the altitude is positive in the upward direction in the inertial frame. On the practical vehicle, the aircraft position in the inertial frame is obtained from the EKF which uses the IMU and DGPS measurements. The altitude is then obtained by multiplying the Z-component of the position (D) by -1 which is mathematically expressed as,

$$h = -D \tag{5.92}$$

It is important for the altitude controller to have very good glide slope tracking during the landing phase, as any deviation from the altitude reference will cause a large longitudinal position error due to the small glide slope angle. Another issue is that it takes a significant

period of time for the altitude controller to produce the correct climb rate reference to track the glide slope. This can cause the aircraft to have a steady-state error when tracking the glide slope, or to not have settled in time for the landing. A common method used to address this issue is to pre-inject the desired climb rate reference into the climb rate controller and then have the altitude controller work around this set point to reject any disturbances. This pre-injected climb rate reference ($\dot{\bar{h}}_{ref}$) is only added to the altitude controller's climb rate reference command during the glide slope tracking phase. The pre-injected climb rate reference is provided by the guidance system and its derivation is discussed in section 6.4. Figure 5.12 shows the block diagram of the classical altitude controller,

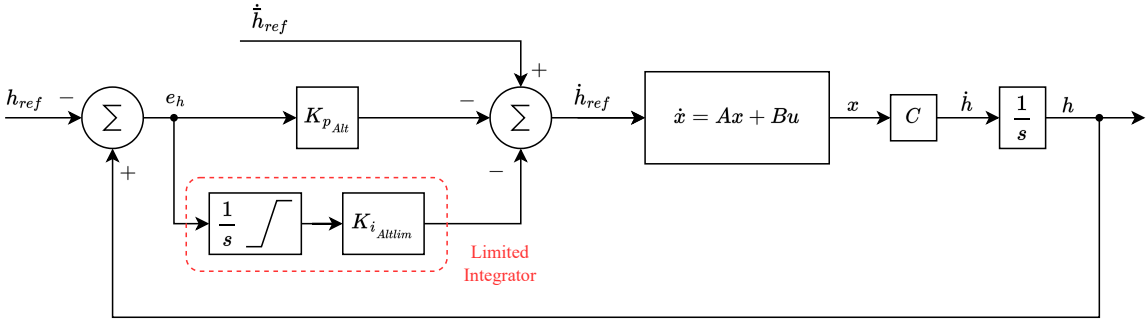


Figure 5.12: Block diagram of the altitude controller.

where h_{ref} is the altitude reference, h is the altitude, $\dot{h}_{ref}$ is the climb rate reference command, and $\dot{\bar{h}}_{ref}$ is the pre-injected climb rate reference. The limited integrator is added to compensate for climb rate biases that can cause glide slope tracking errors. The limited integrator does not form part of the altitude controller and is instead an external component which, similar to the $\dot{\bar{h}}_{ref}$ component, helps the altitude controller. The parameters of the limited integrator are manually tuned and their selection are discussed later in this subsection.

The altitude plant can be formed using the closed-loop model of the climb rate controller from equation 5.87. The altitude can be easily obtained from the climb rate model state vector $\boldsymbol{x}_{\dot{h}}$ by extracting the climb rate and then integrating it. The altitude controller plant relates the climb rate reference to the altitude state, and this is calculated as,

$$\frac{h(s)}{\dot{h}_{ref}(s)} = \frac{1}{s}\boldsymbol{C}_{\dot{h}}(s\boldsymbol{I} - \boldsymbol{A}_{\dot{h}})^{-1}\boldsymbol{B}_{\dot{h}_{ref}} \tag{5.93}$$

where $\boldsymbol{C}_{\dot{h}}$ and $\boldsymbol{B}_{\dot{h}_{ref}}$ have been defined previously in equations 5.89 and 5.91 respectively.

For the design of the altitude controller, the pre-injected climb rate reference ($\dot{\bar{h}}_{ref}$) is set to zero. The altitude controller only consists of a proportional component as the addition of an integrator makes the controller response too slow. The altitude proportional

control law, with reference to figure 5.12, is define as,

$$\dot{h}_{ref} = -K_{p_{Alt}} e_h \tag{5.94}$$

with,

$$e_h = h - h_{ref} \tag{5.95}$$

where e_h is the altitude error. The altitude controller has to have very good step response performance to successfully command the aircraft to land on the moving platform. To achieve this, it is crucial that the controller eliminates any altitude error before touchdown. Therefore, the controller needs to have a 2% settling time that is less than the time it takes the aircraft to complete its glide slope. The glide slope time can be calculated using,

$$t_{GS} = \frac{L_{GS}}{V_{\text{ground}}} \tag{5.96}$$

where L_{GS} is the glide slope longitudinal distance and V_{ground} is the aircraft ground speed. As will be shown in chapter 6, the glide slope length is chosen to be 250 m and the ideal ground speed is chosen as 18 m/s for the moving platform landing. The glide slope time is therefore calculated as,

$$t_{GS} = \frac{250}{18} = 13.89\,\text{s} \tag{5.97}$$

The 2% settling time requirement can therefore be specified as,

$$t_s < 13\,\text{s} \tag{5.98}$$

Additional requirements that the controller should meet are: a rise time of less than 6 seconds, less than 20% overshoot, and zero steady state error. The additional requirements are obtained from Peddle [8], who choose these requirements based on the physical capabilities of the aircraft. The altitude gain that is selected to meet these requirements is,

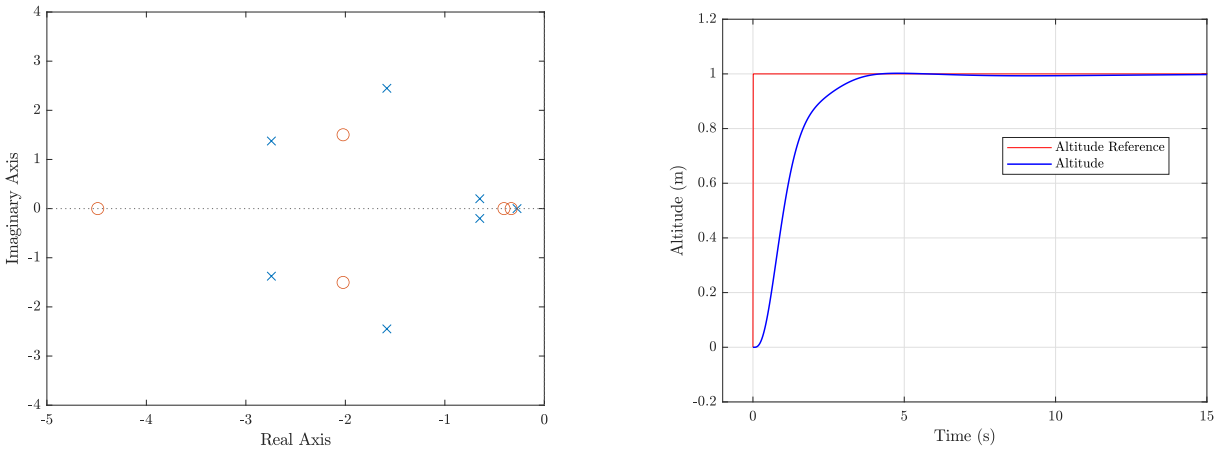
$$K_{p_{Alt}} = 0.8 \tag{5.99}$$

The pole-zero plot and step response of the altitude controller for this gain are shown in Figure 5.13.

The damping ratio of the most dominant complex pole pair in Figure 5.13a is,

$$\zeta_{cl} = 0.956 \tag{5.100}$$

which is quite high. This ensures that there is minimal overshoot in the step response. The altitude step response in Figure 5.13b has a 2% settling time of 3.39 seconds, a rise time of 2.26 seconds, an overshoot of less than 1%, and zero steady-state error. These characteristics are within the requirements for the controller. The settling time is much



(a) Pole-Zero Plot **(b)** Step Response

Figure 5.13: Altitude controller pole-zero plot and step response

faster then the requirement, which is good, especially since the ground speed could increase significantly due to a tail wind. The altitude limited integrator gain is found through manual tuning using the non-linear simulation model. This gain is found to be,

$$K_{i_{Altlim}} = 0.5 \tag{5.101}$$

The saturation limits for the integrator are set to $\pm 0.1\,\mathrm{m/s}$ which is believed to be the maximum climb rate bias.

To create the altitude response closed-loop model, it is first required to augment the altitude state h into the climb rate closed-loop model from equation 5.87 which results in,

$$\begin{bmatrix} \dot{\boldsymbol{x}}_{\dot{h}} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{\dot{h}} & & & & \boldsymbol{0}_{9\times 1} \\ 0 & -\overline{V_T} & 0 & \overline{V_T} & \boldsymbol{0}_{1\times 6} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{\dot{h}} \\ h \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_{\dot{h}} \\ \boldsymbol{0}_{1\times 2} \end{bmatrix} \boldsymbol{u}_{\dot{h}} \tag{5.102}$$

where the climb rate ($\dot{h}$) state is extracted from the $\boldsymbol{x}_{\dot{h}}$ vector using equation 5.77. The $\boldsymbol{A}$ matrix from this equation can be more compactly written as $\boldsymbol{A}_{\dot{h}_2}$, and the altitude control law from equation 5.94 can be substituted in to give,

$$\begin{bmatrix} \dot{\boldsymbol{x}}_{\dot{h}} \\ \dot{h} \end{bmatrix} = \left[\boldsymbol{A}_{\dot{h}_2} + \begin{bmatrix} \boldsymbol{B}_{\dot{h}_{ref}} \\ 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{0}_{9\times 1} & -K_{p_{Alt}} \end{bmatrix} \right] \begin{bmatrix} \boldsymbol{x}_{\dot{h}} \\ h \end{bmatrix} + \begin{bmatrix} K_{p_{Alt}} \boldsymbol{B}_{\dot{h}_{ref}} & \boldsymbol{B}_{\bar{v}_{ref}} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} h_{ref} \\ \bar{v}_{ref} \end{bmatrix} \tag{5.103}$$

where,

$$\boldsymbol{B}_{\dot{h}_{ref}} = \boldsymbol{B}_{\dot{h}} \begin{bmatrix} 1 & 0 \end{bmatrix}^{\mathsf{T}} \; ; \; \boldsymbol{B}_{\bar{v}_{ref}} = \boldsymbol{B}_{\dot{h}} \begin{bmatrix} 0 & 1 \end{bmatrix}^{\mathsf{T}} \tag{5.104}$$

Equation 5.103 can be written in a more compact form as,

$$\dot{\boldsymbol{x}}_h = \boldsymbol{A}_h \boldsymbol{x}_h + \boldsymbol{B}_h \boldsymbol{u}_h \tag{5.105}$$

with,

$$h = \boldsymbol{C}_h \boldsymbol{x}_h \tag{5.106}$$

where,

$$\boldsymbol{C}_h = \begin{bmatrix} \mathbf{0}_{1\times 9} & 1 \end{bmatrix} \tag{5.107}$$

The transfer function that relates the altitude reference to altitude is calculated as,

$$\frac{h(s)}{h_{ref}(s)} = \boldsymbol{C}_h (s\boldsymbol{I} - \boldsymbol{A}_h)^{-1} \boldsymbol{B}_{h_{ref}} \tag{5.108}$$

where,

$$\boldsymbol{B}_{h_{ref}} = \boldsymbol{B}_h \begin{bmatrix} 1 & 0 \end{bmatrix}^\intercal \tag{5.109}$$

5.2.2 Lateral Controllers Design

5.2.2.1 Lateral Specific Acceleration(LSA) Controller

The lateral specific acceleration (LSA) controller controls the aircraft's lateral acceleration by commanding the rudder deflection based on references it receives from the yaw controller. To design the LSA controller, it is assumed that the lateral-directional dynamics can be decoupled to form the lateral dynamics and directional dynamics which are independent of each other, as was done by Peddle [9]. The LSA controller is designed for the directional dynamics while the roll rate controller, which is introduced later, is designed for the lateral dynamics. To perform this decoupling, it is assumed, that the side forces and yawing moments produced by the aircraft roll rate and aileron defection are significantly smaller than those produced by the yaw rate, rudder deflection, and sideslip. It is also assumed that the rolling moments produced by the yaw rate and rudder deflection are significantly smaller than those produced by the roll rate, aileron deflection, and sideslip. These assumptions are not entirely true, as the rudder deflection influences the lateral dynamics, while the aileron deflection influences the directional dynamics [9]. These effects are known as the adverse yaw and rudder-induced roll and are treated as disturbances to the appropriate controllers. The lateral acceleration measurements on the practical vehicle are obtained from a three-axis accelerometer which measures the acceleration in the body axes. The LSA reference is set to zero for conventional flight (waypoint navigation) to allow for coordinated turns, and therefore the yaw controller is disabled in this stage. The LSA reference is no longer zero when the yaw controller is activated during the de-crab stage of landing. The LSA controller is formed by combining two separate controllers together, as shown in figure 5.14, with the inner controller used for stability augmentation and the outer controller used to regulate the lateral specific acceleration. The inner controller is only used to place the dutch roll mode poles at a desired location.

In figure 5.14, B_w is the lateral specific acceleration in the wind axes, $B_{w_{ref}}$ is the

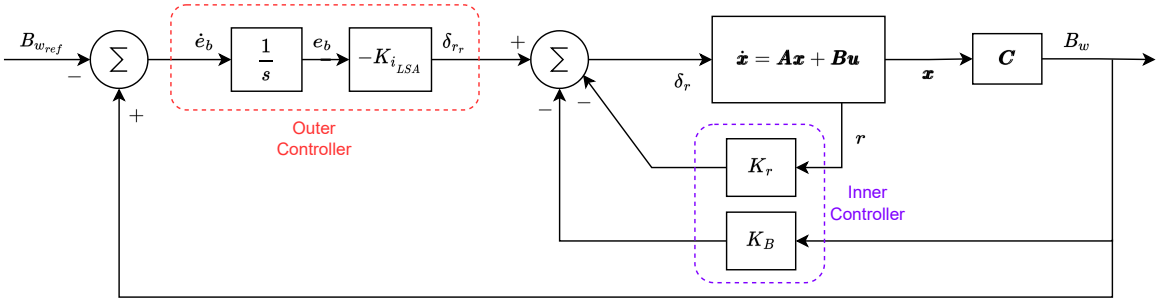


Figure 5.14: Block diagram of the LSA controller.

lateral specific acceleration reference, and δ_r is the rudder deflection from trim.

Stability Augmentation Controller Design

The LSA plant is represented by the reduced-order directional dynamics which is given as,

$$\begin{bmatrix} \dot{\beta} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} -\frac{Y_\beta}{m\bar{V}_T} & -1 \\ \frac{N_\beta}{I_{zz}} & \frac{N_R}{I_{zz}} \end{bmatrix} \begin{bmatrix} \beta \\ r \end{bmatrix} + \begin{bmatrix} \frac{Y_{\delta_R}}{m\bar{V}_T} \\ \frac{N_{\delta_R}}{I_{zz}} \end{bmatrix} \delta_r \tag{5.110}$$

$$B_w = \begin{bmatrix} \frac{Y_\beta}{m} & \frac{Y_R}{m} \end{bmatrix} \begin{bmatrix} \beta \\ r \end{bmatrix} + \begin{bmatrix} \frac{Y_{\delta_R}}{m} \end{bmatrix} \delta_r \tag{5.111}$$

where I_{zz} is the principle moment of inertia about the z-axis, β is the slideslip angle, r is the yaw rate from trim, and $Y_\beta, Y_R, Y_{\delta_R}, N_\beta, N_R$, and N_{δ_R} are stability and control derivatives which are defined in appendix A.3.3. The control law for the stability augmentation controller with respect to figure 5.14 is given as,

$$\delta_r = -K_r r - K_B B_w + \delta_{r_r} \tag{5.112}$$

where δ_{r_r} is the rudder command from the outer controller. This control law is substituted into the directional dynamics of equation 5.110 and the characteristic equation can be derived. This characteristic equation, which is obtained from Peddle [9], is only valid if the following constraints are met:

$$\left| \frac{K_B}{K_r} \right| \ll \left| \frac{ml_F}{Y_R(l_D - l_F)} \right| \tag{5.113}$$

$$|K_r| \ll \left| \frac{m\overline{V}_T l_W}{Y_{\delta_R}(l_W - l_F)} \right| \tag{5.114}$$

where,

$$l_W = -\frac{N_\beta}{Y_\beta} \quad ; \quad l_D = -\frac{N_R}{Y_R} \quad ; \quad l_F = -\frac{N_{\delta_R}}{Y_{\delta_R}} \tag{5.115}$$

Coefficient matching is used on the characteristic equation to obtain the controller gain equations as,

$$K_B = \frac{\frac{Y_\beta N_R}{m\overline{V}_T I_{zz}} + \frac{N_\beta}{I_{zz}} - \omega_{n_{cl}}^2}{\frac{Y_{\delta_R}}{m}\left[\omega_{n_{cl}}^2 - \frac{Y_\beta}{I_{zz}}\left(\frac{N_\beta}{Y_\beta} - \frac{N_{\delta_R}}{Y_{\delta_R}}\right)\right]} \tag{5.116}$$

$$K_r = \frac{I_{zz}}{N_{\delta_R}}\left[\frac{Y_\beta}{m\overline{V}_T} + \frac{N_R}{I_{zz}} + 2\zeta_{cl}\omega_{n_{cl}}\left(1 + K_B\frac{Y_{\delta_R}}{m}\right)\right] \tag{5.117}$$

To calculate the gains, it is first required to choose the closed-loop damping ratio ζ_{cl} and natural frequency $\omega_{n_{cl}}$ of the dutch roll mode poles. The damping ratio was chosen to be 0.9 and the closed-loop natural frequency was chosen as 1.2 times the open-loop natural frequency of the dutch roll mode poles(equation 4.120). The selected values are therefore,

$$\zeta_{cl} = 0.9 \tag{5.118}$$

$$\omega_{n_{cl}} = 1.2\omega_{n_{ol}} = 4.60548 \text{ rad/s} \tag{5.119}$$

The gains are therefore calculated as,

$$K_B = -0.0899 \tag{5.120}$$

$$K_r = -0.3176 \tag{5.121}$$

For these gains to be valid, the constraints of equations 5.113 and 5.114 must be satisfied. Therefore evaluating the constraints results in,

$$\left|\frac{K_B}{K_r}\right| = 0.2831 \ll \left|\frac{ml_F}{Y_R(l_D - l_F)}\right| = 115.7929 \tag{5.122}$$

$$|K_r| = 0.3176 \ll \left|\frac{m\overline{V}_T l_W}{Y_{\delta_R}(l_W - l_F)}\right| = 4.7772 \tag{5.123}$$

The inner controller gains are therefore valid.

LSA Regulation Controller Design

The transfer function which represents the plant for the LSA regulation controller is defined by Peddle [9] as,

$$B_w \approx K_{ss}\delta_{r_r} \tag{5.124}$$

where K_{ss} is the steady-state gain of the transfer function and is given as,

$$K_{ss} = \frac{Y_{\delta_R}Y_\beta}{mI_{zz}\omega_{n_{cl}}^2}\left(\frac{N_\beta}{Y_\beta} - \frac{N_{\delta_R}}{Y_{\delta_R}}\right)\left(1 + K_B\frac{Y_{\delta_R}}{m}\right)^{-1} \tag{5.125}$$

The control law for the outer controller with respect to Figure 5.14 is given as,

$$\delta_{r_r} = -K_{i_{LSA}} e_b \tag{5.126}$$

and,

$$\dot{e}_b = B_w - B_{w_{ref}} \tag{5.127}$$

To obtain a practically feasible controller, Peddle [9] constrains the natural frequency of the closed-loop integrator pole with the following equation,

$$\omega_{n_i} < \frac{1}{3} \left| \sqrt{\frac{-Y_\beta (l_F - l_W)}{I_{zz}}} \right| \tag{5.128}$$

The control law of the outer controller is substituted into the plant from Equation 5.124. Coefficient matching is then performed with a characteristic equation to derive the integrator gain equation as,

$$K_{i_{LSA}} = \frac{a}{K_{ss}} \tag{5.129}$$

where a is the integrator pole location which can be selected. The steady-state gain can be calculated by evaluating Equation 5.125 with the stability augmentation gain (K_B) and closed-loop natural frequency ($\omega_{n_{cl}}$) resulting in,

$$K_{ss} = -2.7604 \tag{5.130}$$

The constraint from Equation 5.128 is evaluated to form,

$$\omega_{n_i} < 1.3429 \text{ rad/s} \tag{5.131}$$

The integrator pole location needs to be lower then this frequency. Therefore, it is chosen as,

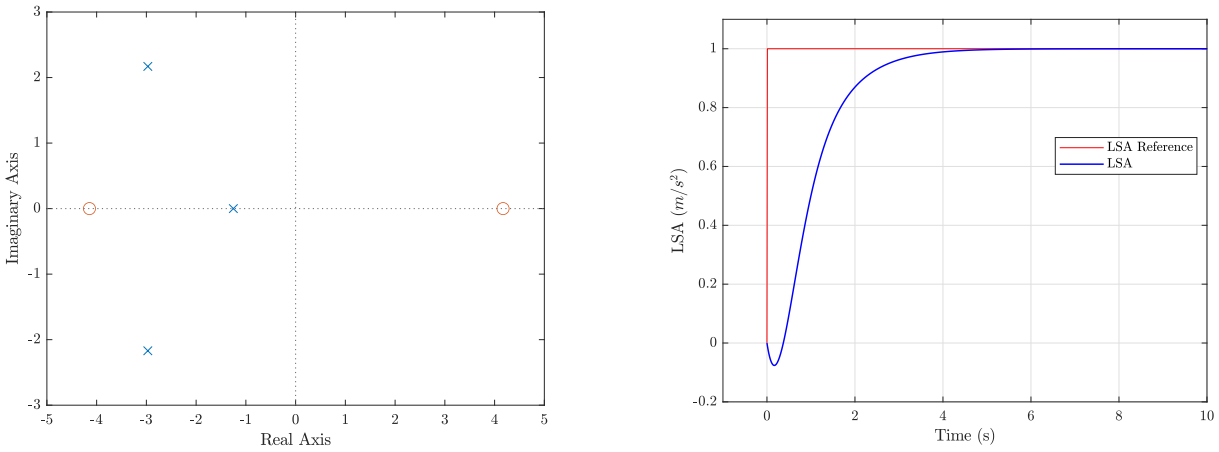
$$a = 0.75 \text{ rad/s} \tag{5.132}$$

The integrator gain can then be calculated using Equation 5.129 to give,

$$K_{i_{LSA}} = -0.2717 \tag{5.133}$$

The pole-zero plot and step response of the LSA controller using these gains are shown in Figure 5.15.

The pole-zero plot shows that the dutch role mode poles are sufficiently separated from the dominant integrator pole of the outer controller. This means that the LSA response depends on the outer controller which is desired since this controller regulates the LSA. The step response has a rise time of 2.22 seconds which is slower then the NSADLC



(a) Pole-Zero Plot **(b)** Step Response

Figure 5.15: LSA controller pole-zero plot and step response

controller. However, this is expected, as the rudder is smaller then the elevator, reducing its effectiveness. This slower rise time is still acceptable for the crab angle controller as it only has to de-crab the aircraft on landing where there will be sufficient time to perform this manoeuvre.

LSA Closed-Loop Model

The LSA controller was designed on the reduced-order directional dynamics. However to design the outer controllers, the LSA controller has to be augmented into the full lateral dynamic model from Equation 4.108. The full lateral model can be more compactly written in state space form as,

$$\dot{\boldsymbol{x}}_{Lat} = \boldsymbol{A}_{Lat}\boldsymbol{x}_{Lat} + \boldsymbol{B}_{Lat}\boldsymbol{u}_{Lat} \tag{5.134}$$

where $\boldsymbol{A}_{Lat}$ and $\boldsymbol{B}_{Lat}$ are system and input matrices, respectively, for the lateral model. The control law of the stability augmentation controller from Equation 5.112 is substituted into this model and the resulting compact state space system is given as,

$$\dot{\boldsymbol{x}}_s = \boldsymbol{A}_s\boldsymbol{x}_s + \boldsymbol{B}_s\boldsymbol{u}_s \tag{5.135}$$

The LSA regulation controller integrator from Equation 5.127 is augmented into this model, and then the control law for the LSA regulation controller from Equation 5.126 is substituted in, resulting in the following compact state space system:

$$\dot{\boldsymbol{x}}_{bw} = \boldsymbol{A}_{bw}\boldsymbol{x}_{bw} + \boldsymbol{B}_{bw}\boldsymbol{u}_{bw} \tag{5.136}$$

with,

$$B_w = \boldsymbol{C}_{bw}\boldsymbol{x}_{bw} \tag{5.137}$$

The closed-loop transfer function that relates the LSA reference to the LSA is given as,

$$\frac{B_w(s)}{B_{w_{ref}}(s)} = \boldsymbol{C}_{bw}(s\boldsymbol{I} - \boldsymbol{A}_{bw})^{-1}\boldsymbol{B}_{B_{w_{ref}}} \tag{5.138}$$

The complete derivation of the closed-loop system for the LSA controller was performed by De Bruin [16].

5.2.2.2 Roll Rate Controller

The role rate controller controls the aircraft's roll rate by commanding aileron deflections based on references it receives from the roll angle controller. The roll rate controller operates on the decoupled lateral dynamics as discussed in the previous section. On the practical vehicle, the roll rate feedback is received from the EKF, which uses a three-axis gyroscope to get the measurements. Figure 5.16 shows the roll rate controller architecture.

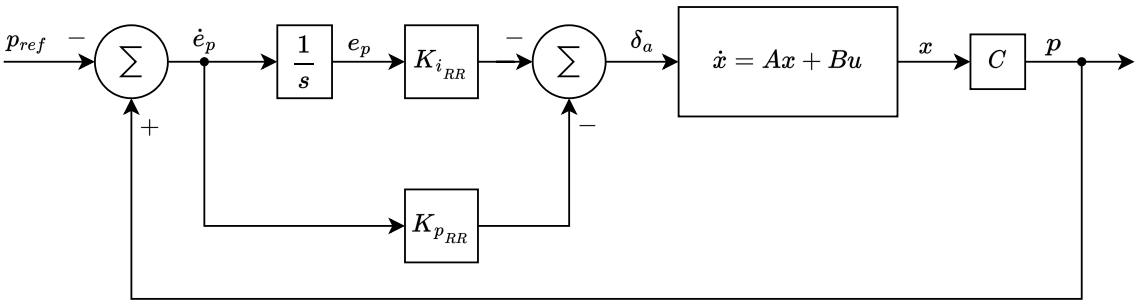


Figure 5.16: Block diagram of the Roll Rate controller.

In the figure, p is roll rate, p_{ref} is the roll rate reference, and δ_a is the aileron deflection from trim. The plant used for the roll rate controller is the reduced-order lateral dynamics, which is represented as,

$$\dot{p} = \left[\frac{L_P}{I_{xx}}\right] p + \left[\frac{L_{\delta_A}}{I_{xx}}\right] \delta_a \tag{5.139}$$

where I_{xx} is the principle moment of inertia about the x-axis, and L_P and L_{δ_A} are control and stability derivatives defined in appendix A.3.3. The roll rate controller has a PI architecture, as shown in Figure 5.16, and therefore the control law with respect to this figure is written as,

$$\delta_a = -K_{p_{RR}}\dot{e}_p - K_{i_{RR}}e_p \tag{5.140}$$

with,

$$\dot{e}_p = p - p_{ref} \tag{5.141}$$

where $\dot{e}_p$ is the roll rate error and e_p is the time integral of the roll rate error. The integrator is state is augmented into the roll rate controller plant and then the roll rate control law is substituted in. Coefficient matching is then performed with a characteristic

equation to derive the roll rate gain equations as,

$$K_{p_{RR}} = \frac{\alpha_1 I_{xx} + L_P}{L_{\delta_A}} \tag{5.142}$$

$$K_{i_{RR}} = \frac{\alpha_0 I_{xx}}{L_{\delta_A}} \tag{5.143}$$

where α_0 and α_1 are the characteristic equation coefficients which are defined based on the desired pole locations. To calculate the gains, the closed-loop integrator and roll mode pole locations needs to be selected. The roll mode pole position was chosen to be kept at its open-loop location while the integrator pole was selected to be slightly slower then the roll mode pole. The pole locations are obtained as,

$$p_1 = \left|\frac{L_P}{I_{xx}}\right| = 12.0775 \text{ rad/s} \tag{5.144}$$

$$p_2 = \left(\frac{3}{4}\right) p_1 = 9.0581 \text{ rad/s} \tag{5.145}$$

where p_1 is the roll mode pole and p_2 is the integrator pole. The characteristic coefficients are therefore calculated as,

$$\alpha_0 = p_1 \times p_2 = 109.3997 \tag{5.146}$$

$$\alpha_1 = p_1 + p_2 = 21.1356 \tag{5.147}$$

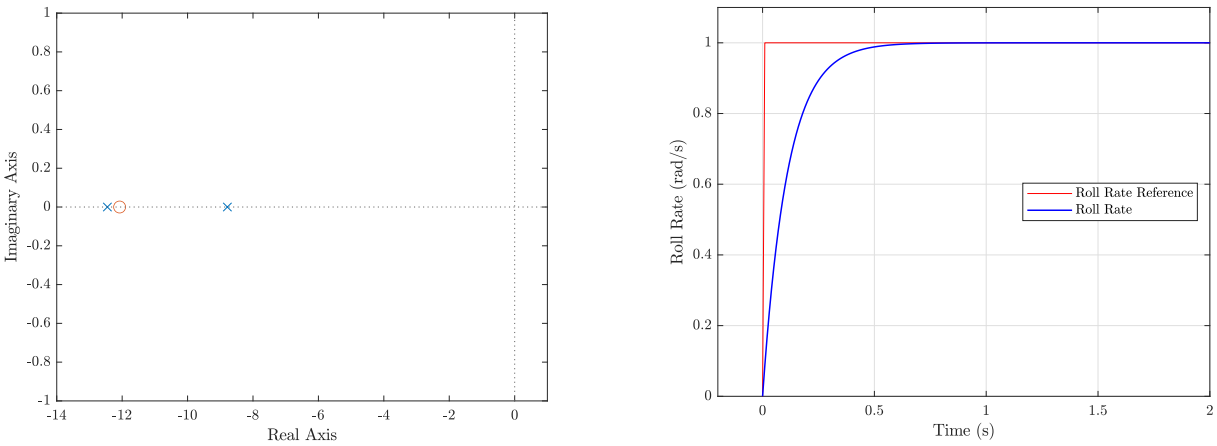
Substituting these coefficient values into the roll rate gain equations results in,

$$K_{p_{RR}} = -0.0644 \tag{5.148}$$

$$K_{i_{RR}} = -0.7783 \tag{5.149}$$

The pole-zero plot and step response of the roll rate rate controller with these gains are shown in Figure 5.17.

The pole-zero plot shows that the integrator pole is dominant, but is still close to the roll mode pole. The roll rate controller step response has a very fast rise and settling time, as well as no overshoot, which is desired to not limit the roll angle controller.



(a) Pole-Zero Plot **(b)** Step Response

Figure 5.17: Roll Rate controller pole-zero plot and step response

Roll Rate Closed-Loop Model

The roll rate controller was designed on the reduced-order lateral dynamics. However, to design the outer controllers, the roll rate controller needs to be augmented into the full lateral dynamic model. The closed-loop model of the LSA controller from Equation 5.136 contains the full lateral model and is used to augment the roll rate controller. The roll rate controller's integrator state is first augmented into the model, and then its control law is substituted in to create the roll rate closed-loop model. The roll rate closed-loop model is represented in a compact state space form as,

$$\dot{\boldsymbol{x}}_p = \boldsymbol{A}_p \boldsymbol{x}_p + \boldsymbol{B}_p \boldsymbol{u}_p \tag{5.150}$$

with,

$$p = \boldsymbol{C}_p \boldsymbol{x}_p \tag{5.151}$$

where,

$$\boldsymbol{C}_p = \begin{bmatrix} 0 & 1 & \boldsymbol{0}_{1\times 4} \end{bmatrix} \tag{5.152}$$

The closed-loop transfer function that relates the roll rate reference to the roll rate is given as,

$$\frac{p(s)}{p_{ref}(s)} = \boldsymbol{C}_p (s\boldsymbol{I} - \boldsymbol{A}_p)^{-1} \boldsymbol{B}_{p_{ref}} \tag{5.153}$$

where,

$$\boldsymbol{B}_{p_{ref}} = \boldsymbol{B}_p \begin{bmatrix} 1 & 0 \end{bmatrix}^\intercal \tag{5.154}$$

The complete derivation of the closed-loop system for the roll rate controller is performed by De Bruin [16].

5.2.2.3 Roll Angle Controller

The roll angle controller controls the roll angle by commanding the roll rate controller based on references it receives from the transition multiplexer. On the practical vehicle, the roll angle is obtained by converting quaternion estimates received from the EKF. The EKF uses a three-axis accelerometer and magnetometer to provide the quaternion estimates. Figure 5.18 shows the roll angle controller architecture, where ϕ is the roll angle from trim, ϕ_{ref} is the reference roll angle, and p_{ref} is the roll rate reference.

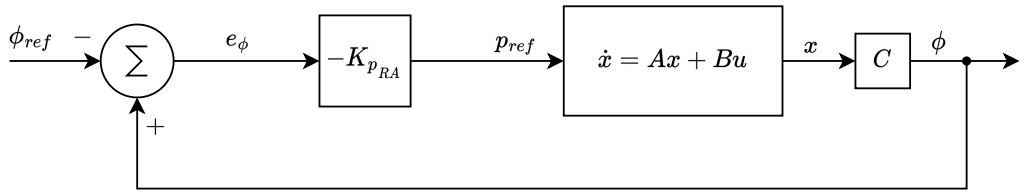


Figure 5.18: Block diagram of the Roll Angle controller.

The roll angle reference is limited to $\pm\frac{\pi}{6}$ rad ($\pm 30°$) to prevent excessive banking which can cause the aircraft to stall. The roll angle controller is designed on the full lateral dynamic model that is augmented with the LSA and roll rate controllers. This model is represented by the closed-loop roll rate controller model, from Equation 5.150, which is used to design the roll angle controller plant. The roll angle plant consists of a transfer function which relates the roll rate reference to the roll angle, and is given as,

$$\frac{\phi(s)}{p_{ref}(s)} = \boldsymbol{C}_\phi(s\boldsymbol{I} - \boldsymbol{A}_p)^{-1}\boldsymbol{B}_{p_{ref}} \tag{5.155}$$

where,

$$\boldsymbol{C}_\phi = \begin{bmatrix}\boldsymbol{0}_{1\times 3} & 1 & \boldsymbol{0}_{1\times 2}\end{bmatrix} \tag{5.156}$$

The roll angle controller uses a proportional controller architecture as shown in Figure 5.18 and therefore its control law with respect to this figure is given as,

$$p_{ref} = -K_{p_{RA}} e_\phi \tag{5.157}$$

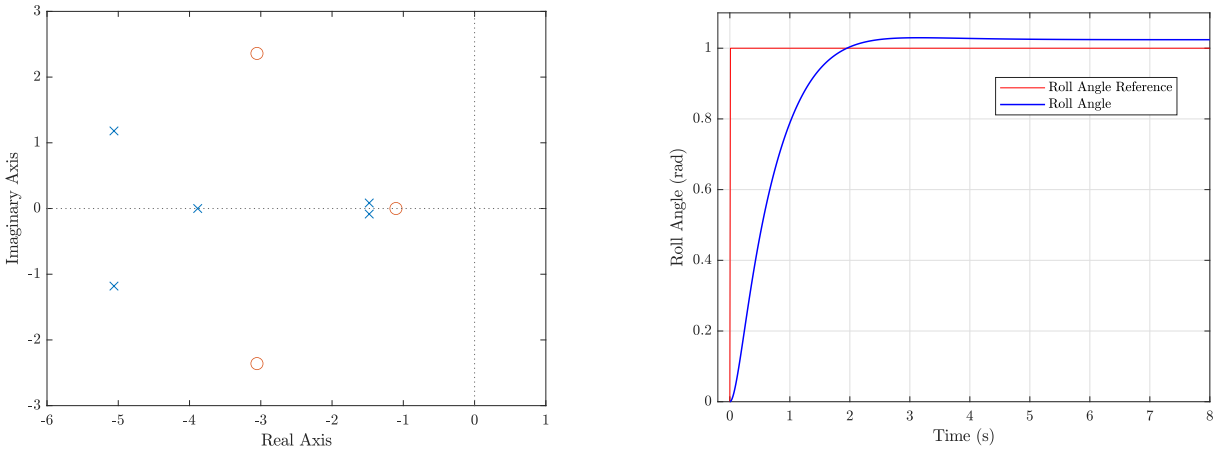
with,

$$e_\phi = \phi - \phi_{ref} \tag{5.158}$$

where, e_ϕ is the roll angle error. The roll angle controller's step response needs to have minimal overshoot and a 2% settling time of less than 3 seconds. These requirements were obtained from De Bruin [16], who chose them to ensure that the roll angle controller's response would be acceptable for the outer cross-track controller. The proportional gain $K_{p_{RA}}$ is adjusted to meet these requirements with the resulting gain found to be,

$$K_{p_{RA}} = 1.5 \tag{5.159}$$

The pole zero plot and step response for the roll angle controller with this gain value is shown in Figure 5.19.



(a) Pole-Zero Plot **(b)** Step Response

Figure 5.19: Roll Angle controller pole-zero plot and step response

The step response has less than 1% overshoot and a 2% settling time of 2 seconds, which are within the requirements for the controller. Unfortunately, there is a steady-state error in the step response, which is due to the roll angle controller not containing an integrator. An integrator was considered to be added to the controller however the response time was too slow and would not meet the controller transient response requirements. The steady-state error will instead be handled by the outer cross-track controller.

Roll Angle Closed-Loop Model

To design the outer controllers, the closed-loop model for the roll angle controller needs to be derived. The control law of the roll angle controller is substituted into the roll rate closed-loop model from Equation 5.150. The compact state space form of the resulting model is given as,

$$\dot{\boldsymbol{x}}_\phi = \boldsymbol{A}_\phi \boldsymbol{x}_\phi + \boldsymbol{B}_\phi \boldsymbol{u}_\phi \tag{5.160}$$

with,

$$\phi = \boldsymbol{C}_\phi \boldsymbol{x}_\phi \tag{5.161}$$

where,

$$\boldsymbol{C}_\phi = \begin{bmatrix} \boldsymbol{0}_{1\times 3} & 1 & \boldsymbol{0}_{1\times 2} \end{bmatrix} \tag{5.162}$$

The closed-loop transfer function that relates the roll angle reference to the roll angle is given as,

$$\frac{\phi(s)}{\phi_{ref}(s)} = \boldsymbol{C}_\phi (s\boldsymbol{I} - \boldsymbol{A}_\phi)^{-1} \boldsymbol{B}_{\phi_{ref}} \tag{5.163}$$

where,

$$\boldsymbol{B}_{\phi_{ref}} = \boldsymbol{B}_\phi \begin{bmatrix} 1 & 0 \end{bmatrix}^\intercal \tag{5.164}$$

The complete derivation of the closed-loop system for the roll angle controller is performed by De Bruin [16].

5.2.2.4 First Cross-Track Controller

The first cross-track controller is used to manoeuvre the aircraft along the ground track to allow it to follow waypoints around the airfield. The ground track is a straight-line segment between two waypoints and it is define in the XY-plane of the inertial frame. The first cross-track controller manoeuvres the aircraft by regulating the cross-track error (y), which it does by sending commands to the roll angle controller based on the references it receives from the guidance system. The cross-track error (y) is the perpendicular distance between the aircraft position and the ground track, as shown in Figure 5.20. The cross-track error and in-track distance (x) is provided by the guidance algorithm which is further discussed in section 6.1.

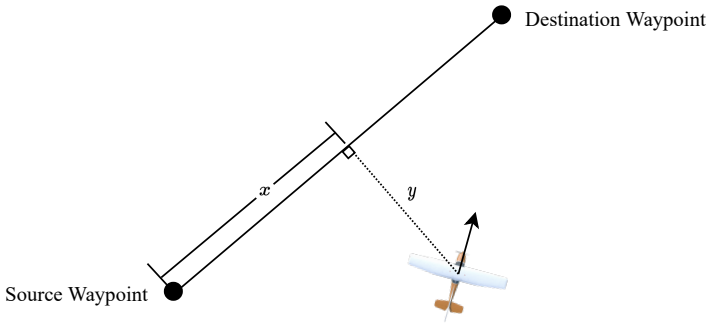


Figure 5.20: Diagram showing the cross-track error measurement.

As mentioned in the roll angle controller design section, the roll angle reference is limited to $\pm\frac{\pi}{6}$ rad ($\pm 30°$) to be within the physical capabilities of the aircraft. A major drawback of using the first cross-track controller is that, when the controller is activated while the aircraft is far from the ground track, it causes the aircraft to circle continuously around a point, resulting in the aircraft being unable to follow the ground track. This is caused by the first cross-track controller saturating due to the limit placed on the roll angle reference. This issue can be solved by using a second cross-track controller combined with a heading controller which can control the aircraft's heading. Unfortunately, these two controllers have a very slow response time which would cause a large cross-track being present during landing. The first cross-track controller has a faster response time, and it should therefore have a lower cross-track error. It is therefore desired to use the first cross-track controller when the aircraft is close to the ground track, and the second cross-track controller when the aircraft is far from the ground track. This can be achieved by using a transition multiplexer that selects which roll angle reference to give based on the aircraft's distance to the ground track. The transition multiplexer is discussed in a later section. Figure 5.21 shows the architecture of the first cross-track controller.

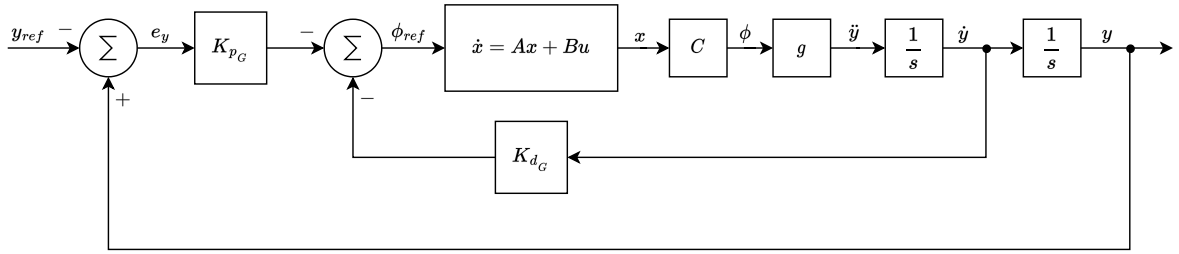


Figure 5.21: Block diagram of the first cross-track controller.

In the figure, y is the cross-track error, $\dot{y}$ is the cross-track error rate, $\ddot{y}$ is the cross-track error acceleration, and y_{ref} is the cross-track error reference. The cross-track error and cross-track error rate are produced by the guidance algorithm for the simulation and practical vehicle, with the cross-track error rate being calculated using the vehicle ground speed. The cross-track error reference is normally set to zero by the guidance control system, which causes the aircraft to follow the ground track as accurately as possible. This reference only changes during the landing stage when the aircraft tracks the moving platform. This will be discussed further in chapter 6.

For an aircraft that is in a steady turn with a non-zero constant roll angle ϕ, the lift vector acting on the aircraft counters the aircraft's weight and centripetal acceleration used to initiate the turn. This centripetal acceleration is the lateral acceleration acting on the aircraft and it can be expressed as,

$$a_{_L} = g\ \tan(\phi) \tag{5.165}$$

where g is the gravitational acceleration value on earth. Assuming that the roll angle is small, $\tan(\phi) \approx \phi$, and therefore Equation 5.165 can be simplified to,

$$\ddot{y} = g\phi \tag{5.166}$$

The first cross-track controller is designed on the full lateral linear model that is augmented with the LSA, roll rate, and roll angle controllers. The first cross-track controller plant can be represented as a transfer function that relates the roll angle reference to the cross-track error, and this transfer function is derived as,

$$\frac{y(s)}{\phi_{ref}(s)} = \frac{g}{s^2}\boldsymbol{C}_{\phi}(s\boldsymbol{I} - \boldsymbol{A}_{\phi})^{-1}\boldsymbol{B}_{\phi_{ref}} \tag{5.167}$$

The first cross-track controller uses a PD control architecture and its control law, with reference to Figure 5.21, is given as,

$$\phi_{ref} = -K_{p_{_G}}e_y - K_{d_{_G}}\dot{y} \tag{5.168}$$

with,

$$e_y = y - y_{ref} \tag{5.169}$$

where e_y is the cross-track error. To obtain high lateral landing accuracy, it is important that the first cross-track controller minimises the cross-track error before touchdown. To achieve this goal, the 2% settling time of the controller needs to be less than the time it takes the aircraft to complete the glide slope, as discussed in the altitude controller section. The settling time requirement is therefore chosen as,

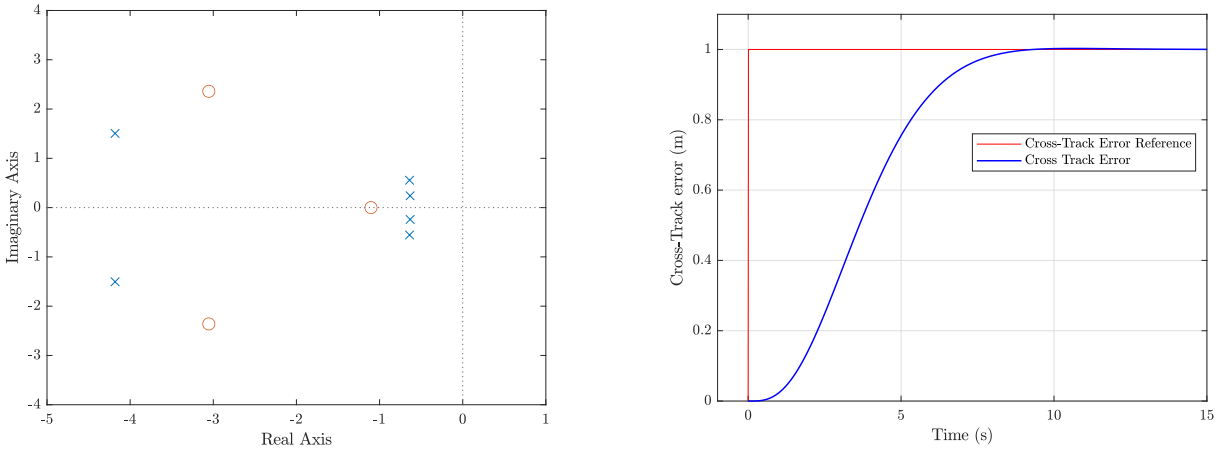
$$t_s < 13\,\text{s} \tag{5.170}$$

The controller gains are selected to meet this requirement and to also cause the dominant closed-loop pole pair to have a damping ratio (ζ_{cl}) of 0.9. This high damping ratio was chosen so that the step response has minimal overshoot. The values determined for the gains are,

$$K_{p_G} = 0.017 \tag{5.171}$$

$$K_{d_G} = 0.065 \tag{5.172}$$

The pole-zero plot and step response of the first cross-track controller with these gains are shown in Figure 5.22. The dominant poles, shown in Figure 5.22a, have a damping



(a) Pole-Zero Plot **(b)** Step Response

Figure 5.22: First cross-track controller pole-zero plot and step response

ratio of 0.935 which is within the requirement. The step response has minimal overshoot and a 2% settling time of 7.89 seconds which is well within the requirements. A limited integrator was considered to be added to the first cross-track controller to deal with roll angle bias. However, this integrator significantly slowed down the response time, which resulted in the settling time requirement not being met. The integrator also introduced excessive overshoot, which is not desired. After consulting the simulation and practical

data, it was found that the effect of roll angle bias was negligible and hence the limited integrator was omitted from the controller.

First Cross-Track Controller Closed-Loop Model

The first cross-track controller needs to be augmented into the closed-loop model of the roll angle controller, from Equation 5.160, so that the crab angle controller can be design on it. First the cross-track error and cross-track error rate states are augmented into the roll angle state vector ($\boldsymbol{x}_\phi$). The control law of the first cross-track controller is then augmented in to derive the closed-loop model, which is given in a compact state space form as,

$$\dot{\boldsymbol{x}}_y = \boldsymbol{A}_y \boldsymbol{x}_y + \boldsymbol{B}_y \boldsymbol{u}_y \tag{5.173}$$

with,

$$y = \boldsymbol{C}_y \boldsymbol{x}_y \tag{5.174}$$

where,

$$\boldsymbol{C}_y = \begin{bmatrix} \boldsymbol{0}_{1\times 7} & 1 \end{bmatrix} \tag{5.175}$$

The closed-loop transfer function that relates the cross-track reference to the cross-track error is given as,

$$\frac{y(s)}{y_{ref}(s)} = \boldsymbol{C}_y (s\boldsymbol{I} - \boldsymbol{A}_y)^{-1} \boldsymbol{B}_{y_{ref}} \tag{5.176}$$

where,

$$\boldsymbol{B}_{y_{ref}} = \boldsymbol{B}_y \begin{bmatrix} 1 & 0 \end{bmatrix}^\intercal \tag{5.177}$$

The complete derivation of the closed-loop system for the first cross-track controller was performed by De Bruin [16].

5.2.2.5 Crab Angle Controller

The crab angle controller controls the aircraft's crab angle by commanding the LSA controller based on references it receives from the guidance control system. The crab angle is defined as the yaw angle between the X-axis in the body frame and the heading angle of the ground track. The crab angle controller is used to execute the de-crab manoeuvre during landing so that the aircraft's landing gear is aligned with the moving platform on touchdown. This controller is only active for the landing phase of flight and for the other phases it is disabled. On the practical vehicle, the crab angle is obtained by subtracting the aircraft's heading from the predefined ground track heading. The aircraft's heading is obtained from the EKF, which uses magnetometer and three-axis accelerometer measurements. Figure 5.23 outlines the crab angle controller architecture.

In the figure, ψ_{crab} is the crab angle, $\psi_{\text{crab}_{ref}}$ is the crab angle reference, and $B_{w_{ref}}$

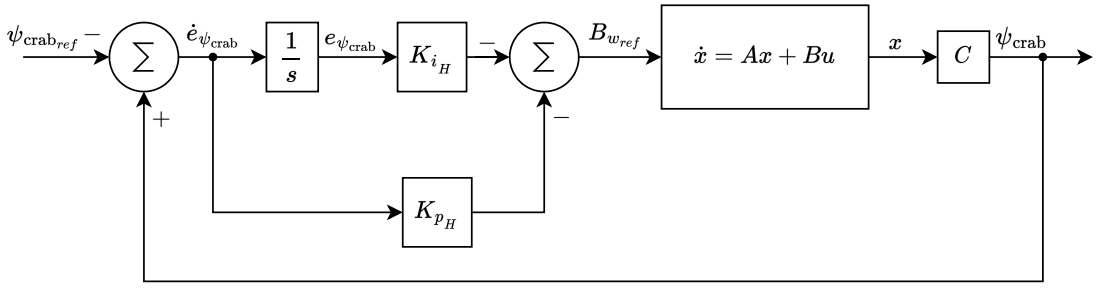


Figure 5.23: Block diagram of the crab angle controller.

is the LSA reference. The crab angle controller is designed on the full lateral linear model augmented with all the inner lateral controllers (LSA, roll rate, roll angle, and first cross-track). The first cross-track controller closed-loop model, from Equation 5.173, encapsulates these dynamics and is therefore used in forming the crab angle controller plant. De Bruin [16] proved that for the controller design (linear simulation), it can be assumed that the crab angle (ψ_{crab}) is approximately equal to the sideslip angle (β) for small deviations from trim. This is mathematically expressed as,

$$\psi_{\text{crab}} = \beta \tag{5.178}$$

This assumption allows the crab angle to be easily extracted from the $\boldsymbol{x}_y$ state vector, which means the transfer function used as the plant for the crab angle controller can be easily derived. This transfer function, which relates the LSA reference to the sideslip angle, and hence the crab angle, is given as,

$$\frac{\psi_{\text{crab}}(s)}{B_{w_{ref}}(s)} = \frac{\beta(s)}{B_{w_{ref}}(s)} = \boldsymbol{C}_\beta(s\boldsymbol{I} - \boldsymbol{A}_y)^{-1}\boldsymbol{B}_{B_{w_{ref}}} \tag{5.179}$$

where,

$$\boldsymbol{C}_\beta = \begin{bmatrix} 1 & \boldsymbol{0}_{1\times 7} \end{bmatrix} \quad ; \quad \boldsymbol{B}_{B_{w_{ref}}} = \boldsymbol{B}_y \begin{bmatrix} 0 & 1 \end{bmatrix}^\intercal \tag{5.180}$$

For the non-linear simulation and practical vehicle, the crab angle is obtained by subtracting the aircraft's heading from the ground track's heading. This crab angle formulation was used when the non-linear simulation and practical results were obtained, which are shown in later chapters.

The crab angle controller uses a PI architecture so that the integrator can compensate for the errors produced due to assuming that the crab and sideslip angles are equal. The control law of the crab angle controller with respect to figure 5.23 is given as,

$$B_{w_{ref}} = -K_{p_H}\dot{e}_{\psi_{\text{crab}}} - K_{i_H}e_{\psi_{\text{crab}}} \tag{5.181}$$

with,

$$\dot{e}_{\psi_{\text{crab}}} = \psi_{\text{crab}} - \psi_{\text{crab}_{ref}} \tag{5.182}$$

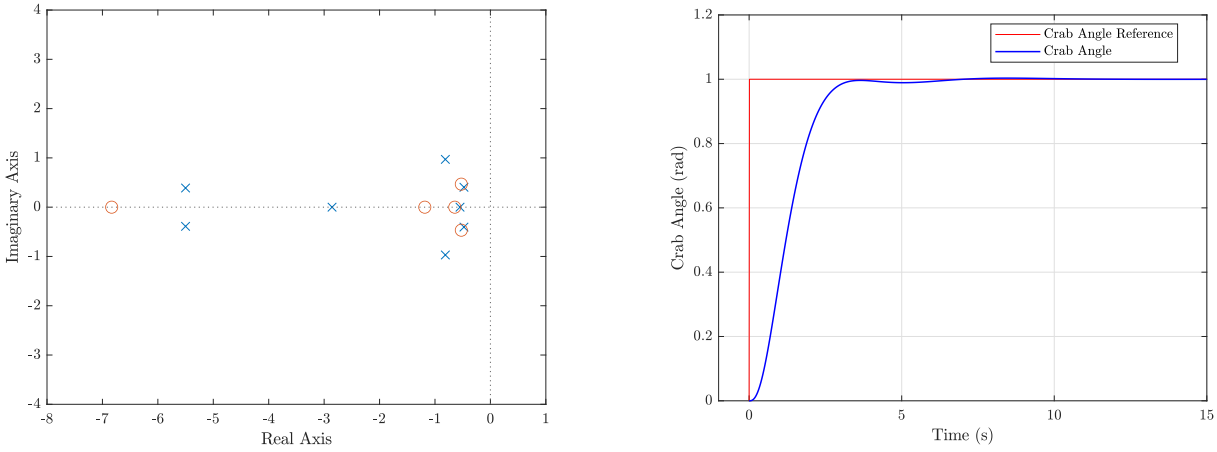
where $\dot{e}_{\psi_{\text{crab}}}$ is the crab angle error and $e_{\psi_{\text{crab}}}$ is the time intergral of the crab angle error.

A side effect of performing the de-crab manoeuvre with the crab angle controller is that the cross-track error of the aircraft increases. This is because the force produced by the lateral component of the wind still acts on the aircraft as the aircraft aligns with the platform. The first cross-track controller would eventually compensate for this, however, due to its slower response time it would not be able to do this before touchdown. It is therefore required that the crab angle controller have a fast response to minimise the cross-track error produced by de-crabbing. In light of this, the requirement for the crab angle controller is selected to be that the step response should have a rise time of less than 3 seconds, which is quite fast, but is achievable. The gains which allow the controller to meet these requirement were determined as,

$$K_{i_H} = -2.25 \tag{5.183}$$

$$K_{p_H} = -1.9 \tag{5.184}$$

The pole-zero plot and step response for the controller with these gains are shown in Figure 5.24.



(a) Pole-Zero Plot **(b)** Step Response

Figure 5.24: Crab angle controller pole-zero plot and step response

The step response has a rise time of 2.27 seconds, which is within the requirement for the controller. There is a slight oscillation in the step response at steady state, which can be attributed to the dominant closed-loop pole pair not having a high damping ratio. This oscillation is still acceptable as it does not amplify the crab angle that the crab angle controller is trying to minimise.

Crab Angle Controller Closed-Loop Model

The crab angle controller integrator state is augmented into Equation 5.173 and then the crab angle control law is substituted in to give the closed-loop model of the crab angle

controller. This closed-loop model is represented in a compact state space form as,

$$\dot{\boldsymbol{x}}_{\psi_{\text{crab}}} = \boldsymbol{A}_{\psi_{\text{crab}}}\boldsymbol{x}_{\psi_{\text{crab}}} + \boldsymbol{B}_{\psi_{\text{crab}}}\boldsymbol{u}_{\psi_{\text{crab}}} \tag{5.185}$$

with,

$$\psi = \boldsymbol{C}_{\psi_{\text{crab}}}\boldsymbol{x}_{\psi_{\text{crab}}} \tag{5.186}$$

where,

$$\boldsymbol{C}_{\psi_{\text{crab}}} = \begin{bmatrix} 1 & \boldsymbol{0}_{1\times 8} \end{bmatrix} \tag{5.187}$$

The closed-loop transfer function that relates the crab angle reference to the crab angle is given as,

$$\frac{\psi_{\text{crab}}(s)}{\psi_{\text{crab}_{ref}}(s)} = \boldsymbol{C}_{\psi_{\text{crab}}}(s\boldsymbol{I} - \boldsymbol{A}_{\psi_{\text{crab}}})^{-1}\boldsymbol{B}_{\psi_{\text{crab}_{ref}}} \tag{5.188}$$

where,

$$\boldsymbol{B}_{\psi_{\text{crab}_{ref}}} = \boldsymbol{B}_{\psi_{\text{crab}}} \begin{bmatrix} 0 & 1 \end{bmatrix}^{\intercal} \tag{5.189}$$

The complete derivation of the closed-loop system for the crab angle controller was performed by De Bruin [16].

5.2.2.6 Transition Multiplexer

As mentioned in the first cross-track controller design section, it is desired to use the first cross-track controller when the aircraft is close to the ground track, and the second cross-track controller when the aircraft is far from the ground track. There exists an optimal cross-track error point to switch from the second cross-track controller to the first, and this point was determined by Le Roux [14] as,

$$|y_e| = \frac{K_{d_G}}{K_{p_G}}\overline{V}_T \tag{5.190}$$

where $\overline{V}_T$ is the trim airspeed, and K_{d_G} and K_{p_G} are the first cross-track controller gains. It is desired to have a gradual transition from one cross-track controller to the other, to ensure that the roll angle command does not abruptly change, which could cause unexpected behaviour by the aircraft. Le Roux [14] proposed using sinusoidal weighting to achieve this gradual transition and this is illustrated in Figure 5.25.

In the plot, b_u represents the upper bound, b_l represents the lower bound, $|y_e|$ is the magnitude of the cross-track error, and r_{HG2} represents the weighting multiplier of the heading and second cross-track controller. The upper bound was chosen by Le Roux [14]

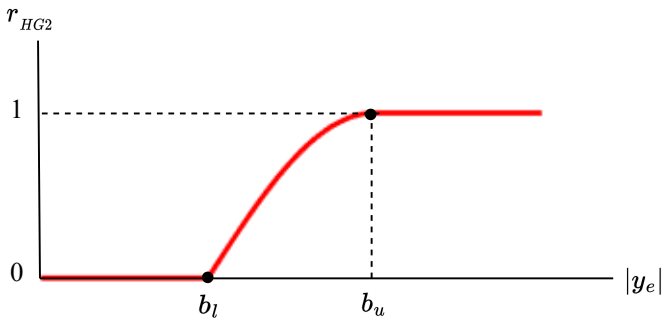


Figure 5.25: Graph showing the weighting multiplier of the heading cross-track controller based on the cross-track error magnitude.

as the optimal switching point mentioned earlier therefore,

$$b_u = \frac{K_{d_G}}{K_{p_G}} \overline{V}_T \tag{5.191}$$

The lower bound is chosen to be half the upper bound therefore,

$$b_l = \frac{b_u}{2} \tag{5.192}$$

There are three regions that can be identified from this graph, and their significance are:

- $0 \leq |y_e| < b_l$ - In this inner region, only the first cross-track controller is active, as the cross-track error is within the operating range of this controller.
- $b_l \leq |y_e| < b_u$ - In this transition region, both controllers are active and their commands are scaled using weighting multipliers. The magnitude of these multipliers are based on the aircraft's cross-track error.
- $b_u \leq |y_e|$ - In this outer region, only the heading and second cross-track controllers are active, as the cross-track error is outside the operating range of the first cross-track controller.

The magnitude of the heading and second cross-track controller weighting multiplier is calculated as,

$$r_{HG2} = sin\left(\frac{\pi}{2}\frac{|y_e| - b_l}{b_u - b_l}\right) \tag{5.193}$$

The first cross-track controller's weighting multiplier can then be calculated as,

$$r_{G1} = 1 - r_{HG2} \tag{5.194}$$

For the inner region, the output of the transition multiplexer is simply the roll angle command from the first cross-track controller. For the outer region, it is the roll angle command from the heading controller. Finally for the transition region, it is the sum of

these two roll angle commands with the weighting multipliers applied to them, which is represented as,

$$\phi_{ref} = r_{_{HG2}}\, \phi_{ref_{HG2}} + r_{_{G1}}\, \phi_{ref_{G1}} \tag{5.195}$$

where $\phi_{ref_{HG2}}$ and $\phi_{ref_{G1}}$ are the roll angle commands from the heading and first cross-track controller respectively. The roll angle command is sent to the roll angle controller from the transition multiplexer.

5.2.2.7 Heading Controller

The heading controller controls the aircraft's heading angle by commanding the roll angle controller based on references it receives from the second cross-track controller. The heading angle refers to the angle between the aircraft's nose and true north. This differs from the crab angle controller whose crab angle refers to the angle between the aircraft's nose and the ground track. The heading angle is limited to $\pm 180°$. On the practical vehicle, the heading angle is obtained from the EKF which uses magnetometer and accelerometer measurements. Figure 5.26 shows that the heading controller uses a proportional architecture.

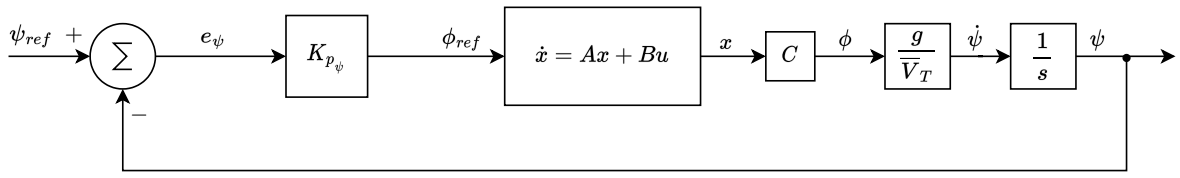


Figure 5.26: Block diagram of the heading controller.

In the figure, ψ is the heading angle, ψ_{ref} is the heading angle reference, and ϕ_{ref} is the roll angle reference. An important characteristic that the heading controller should have is to be able to provide the roll angle command that will cause the aircraft to follow the shortest path to the reference heading. This is not possible with the architecture in Figure 5.26, as the heading error cannot find the shortest path. This point can be illustrated using Figure 5.27 where the reference heading is $150°$ and the current heading is $-120°$. The heading error would be $e_\psi = 150 - (-120) = 270°$ and since the gain K_{p_ψ} is positive, the roll angle reference will also be positive indicating a right turn. This is not the shortest path as a left turn with a $-90°$ heading error would achieve the same the result.

A more sophisticated method is therefore required to calculate the heading error. This method consists of first finding the magnitude of the angle by performing the dot product between vectors formed from the heading angle and heading angle reference. The sign of the angle is then calculated based on the cross and dot products of these vectors. The

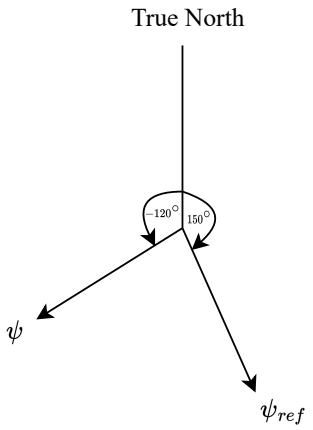


Figure 5.27: Diagram showing an example of a heading state and reference value.

equations used to implement this method are,

$$u_{ref} = [\cos(\psi_{ref}) \, , \, \sin(\psi_{ref}) \, , \, 0] \quad ; \quad u = [\cos(\psi) \, , \, \sin(\psi) \, , \, 0] \tag{5.196}$$

$$u_D = [0 \, , \, 0 \, , \, 1] \tag{5.197}$$

$$x = (u \times u_{ref}) \cdot u_D \tag{5.198}$$

$$\text{sign} = (x > 0) - (x < 0) \tag{5.199}$$

$$e_\psi = \text{sign} \ \arccos(u \cdot u_{ref}) \tag{5.200}$$

where $\times$ is the cross product, $\cdot$ is the dot product, and e_ψ is the heading angle error.

The heading controller is designed on the full lateral model that is augmented with the LSA, roll rate, and roll angle controllers. The heading angle plant can therefore be formed using the roll angle closed-loop model. In the first cross-track controller section it was mentioned that the lateral acceleration for an aircraft that is turning at a constant roll angle is given as,

$$a_{_L} = g \ \tan(\phi) \tag{5.201}$$

Using small angle approximation, and acknowledging that the lateral acceleration is related to heading angle rate, this equation can be written as,

$$a_{_L} = g\phi = \overline{V}_T \dot{\psi} \tag{5.202}$$

Rearranging results in,

$$\dot{\psi} = \frac{g}{\overline{V}_T} \phi \tag{5.203}$$

The heading can therefore be easily obtained from the roll angle which is already included in the roll angle state vector $\boldsymbol{x}_\phi$. The plant of the heading controller is therefore the transfer function that relates the roll angle reference to the heading angle which is given

as,

$$\frac{\psi(s)}{\phi_{ref}(s)} = \frac{1}{s}\boldsymbol{C}_{\dot{\psi}}(s\boldsymbol{I} - \boldsymbol{A}_{\phi})^{-1}\boldsymbol{B}_{\phi_{ref}} \tag{5.204}$$

where,

$$\boldsymbol{C}_{\dot{\psi}} = \begin{bmatrix}\boldsymbol{0}_{1\times 3} & \frac{g}{V_T} & \boldsymbol{0}_{1\times 2}\end{bmatrix} \tag{5.205}$$

The heading control law with respect to Figure 5.26 is,

$$\phi_{ref} = K_{p_{\psi}} e_{\psi} \tag{5.206}$$

with,

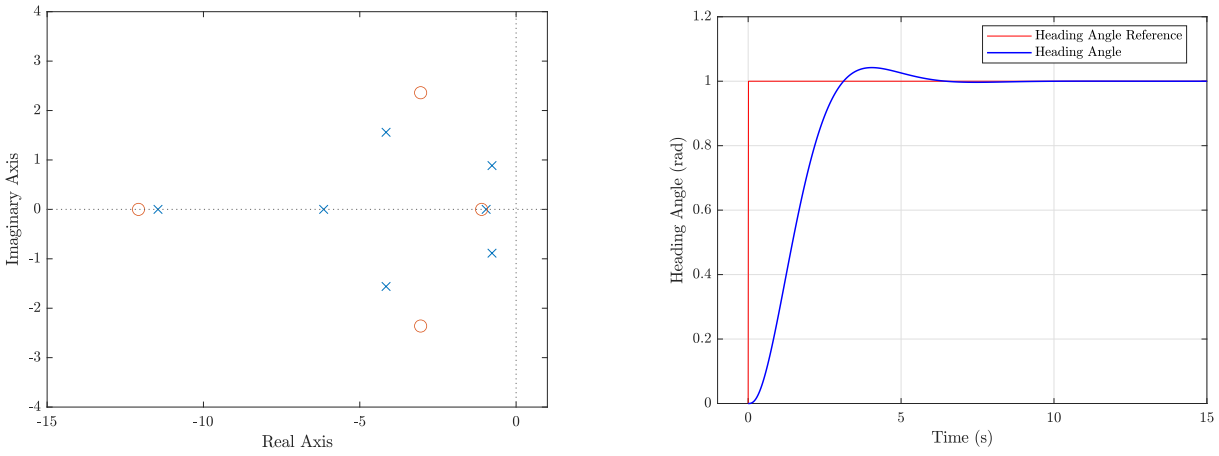
$$e_{\psi} = \psi_{ref} - \psi \tag{5.207}$$

where e_{ψ} is the heading angle error.

The requirements for this controller are that the step response should have a rise time of less than 3 seconds, an overshoot of less than 20%, and a 2% settling time of less than 10 seconds. These requirements were adapted from Le Roux [14] but were made more stringent, so that improved performance compared to Le Roux could be obtained. The heading controller gain that was determined to meet these requirements is,

$$K_{p_{\psi}} = 1.25 \tag{5.208}$$

The pole-zero plot and step response of the heading controller using this gain are shown in Figure 5.28.



(a) Pole-Zero Plot **(b)** Step Response

Figure 5.28: Heading controller pole-zero plot and step response

The step response has a rise time of 2.54 seconds, an overshoot of 4.2%, and a 2% settling time of 5.24 seconds, which are all within the requirements for the controller. There is a slight undershoot after the overshoot which is caused by the dominant pole pair not having a high damping ratio. This undershoot is still within the 2% steady-state limit

and is therefore acceptable.

Heading Controller Closed-Loop Model

The heading controller closed-loop model is derived by first augmenting the heading state into the closed-loop model of the roll angle controller, from Equation 5.160. The control law of the heading controller is then substituted in and the resulting model is written in a compact state space form as,

$$\dot{\boldsymbol{x}}_\psi = \boldsymbol{A}_\psi \boldsymbol{x}_\psi + \boldsymbol{B}_\psi \boldsymbol{u}_\psi \tag{5.209}$$

with,

$$\psi = \boldsymbol{C}_\psi \boldsymbol{x}_\psi \tag{5.210}$$

where,

$$\boldsymbol{C}_\psi = \begin{bmatrix} \boldsymbol{0}_{1\times 6} & 1 \end{bmatrix} \tag{5.211}$$

The closed-loop transfer function that relates the heading angle reference to the heading angle is given as,

$$\frac{\psi(s)}{\psi_{ref}(s)} = \boldsymbol{C}_\psi (s\boldsymbol{I} - \boldsymbol{A}_\psi)^{-1} \boldsymbol{B}_{\psi_{ref}} \tag{5.212}$$

where,

$$\boldsymbol{B}_{\psi_{ref}} = \boldsymbol{B}_\psi \begin{bmatrix} 1 & 0 \end{bmatrix}^\intercal \tag{5.213}$$

The complete derivation of the closed-loop system for the heading controller is performed by Le Roux [14].

5.2.2.8 Second Cross-Track Controller

The second cross-track controller controls the cross-track error by commanding the heading controller based on references it receives from the guidance control system. The second cross-track controller command $\Delta\psi_{ref}$ is added to the heading of the ground track ψ_{track} as shown in Figure 5.29. The second cross-track controller command is also limited to $\pm 45°$ to ensure that the aircraft minimises the cross-track error while also following the ground track. On the physical aircraft, the cross-track is obtained from the guidance algorithm as mentioned in the first cross-track controller section. Figure 5.29 shows the controller architecture of the second cross-track controller. In the figure, y is the cross-track error,

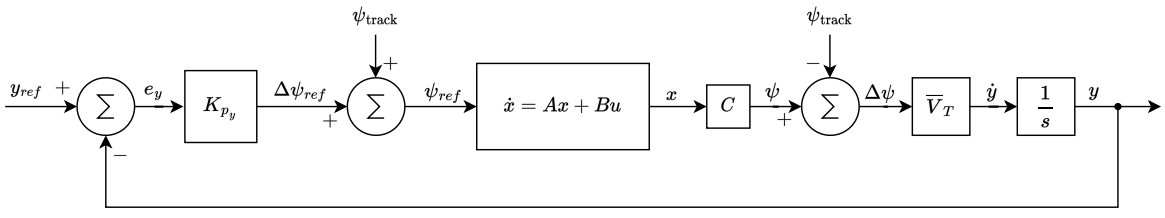


Figure 5.29: Block diagram of the second cross-track controller.

y_{ref} is the cross-track error reference, and $\Delta\psi_{ref}$ is the heading angle reference relative to the ground track heading ψ_{track}. The plant for the second cross-track controller is formed using the closed-loop model of the heading controller. The cross-track error rate is the projection of the aircraft's ground speed in the direction of the cross-track error which represented as,

$$\dot{y} = \overline{V}_T \sin(\psi - \psi_{\text{track}}) \tag{5.214}$$

Using small angle approximation results in,

$$\dot{y} = \overline{V}_T (\psi - \psi_{\text{track}}) \tag{5.215}$$

The heading controller requires a heading reference with respect to true north and, since the output of the second cross-track controller is the relative heading, the ground track heading needs to be added to it before sending it to the heading controller. This is represented as,

$$\psi_{ref} = \Delta\psi_{ref} + \psi_{\text{track}} \tag{5.216}$$

The cross-track error state can now be augmented into the heading closed-loop model, resulting in

$$\begin{bmatrix} \dot{\boldsymbol{x}}_\psi \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_\psi & 0 \\ \boldsymbol{C}_{\dot{y}} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_\psi \\ y \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_\psi \\ \boldsymbol{0}_{1\times 2} \end{bmatrix} \boldsymbol{u}_\psi + \begin{bmatrix} \boldsymbol{0}_{7\times 1} \\ -\overline{V}_T \end{bmatrix} \psi_{\text{track}} \tag{5.217}$$

where,

$$\boldsymbol{C}_{\dot{y}} = \begin{bmatrix} \boldsymbol{0}_{1\times 6} & \overline{V}_T \end{bmatrix} \tag{5.218}$$

The transfer function that relates the relative heading angle reference to the cross-track error represents the second cross-track controller plant and it is given as,

$$\frac{y(s)}{\Delta\psi_{ref}(s)} = \frac{1}{s} \boldsymbol{C}_{\dot{y}} (s\boldsymbol{I} - \boldsymbol{A}_\psi)^{-1} \boldsymbol{B}_{\psi_{ref}} \tag{5.219}$$

The control law of the second cross-track controller with respect to Figure 5.29 is,

$$\Delta\psi_{ref} = K_{p_y} e_y \tag{5.220}$$

with,

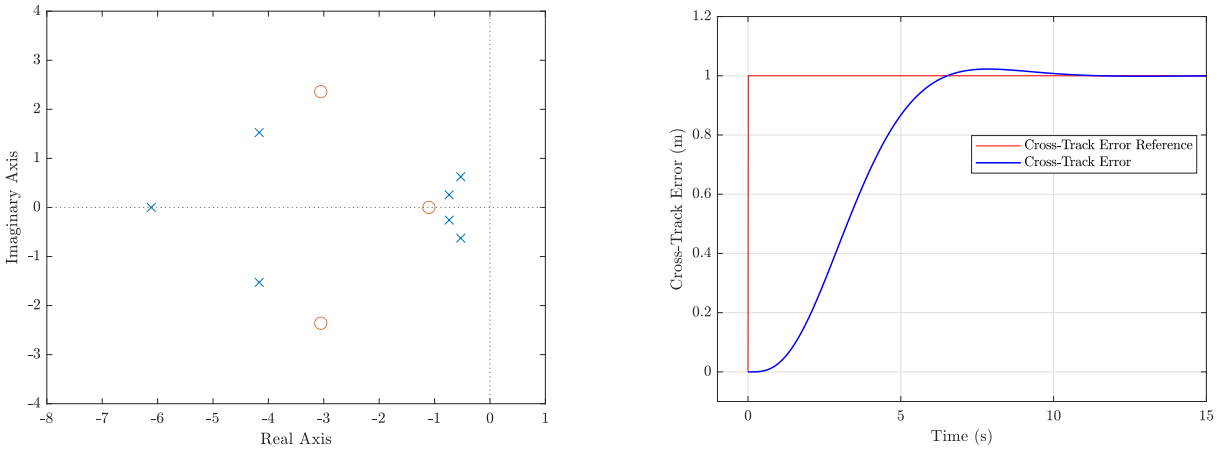
$$e_y = y_{ref} - y \tag{5.221}$$

where e_y is the cross-track error. The step response for the second cross-track controller is required to have a rise time of less than 6 seconds, an overshoot of less than 20%, and ideally a 2% settling time of less than 13 seconds. The 2% settling time requirement is the same as the first cross-track controller and the remaining requirements are adapted from Le Roux [14]. The 2% settling time requirement is not as important to fulfil, as the first

cross-track controller will control the aircraft when it is close to the ground track, which is steady state for the second cross-track controller. The second cross-track controller gain determined to meet these requirements is,

$$K_{p_y} = 0.017 \tag{5.222}$$

The pole-zero plot and step response of the second cross-track controller using this gain are shown in Figure 5.30.



(a) Pole-Zero Plot **(b)** Step Response

Figure 5.30: Second cross-track controller pole-zero plot and step response

The step response has a rise time of 5.25 seconds, an overshoot of 2.3%, and a 2% settling time of 8.54 seconds, which are all within the requirements for the controller. The 2% settling time requirement was met even though it was not as important. There is a slight undershoot after the overshoot due to the low damping ratio of the dominant closed-loop poles. Comparing this step response to the first cross-track controller step response from Figure 5.22b, it can be seen that the second cross-track controller has a faster rise time while the first cross-track controller has a faster 2% settling time. The second cross-track controller also has a larger overshoot which, while being within the requirements, is not desired. This is due to it being desired to have a centimeter level cross-track error on landing which requires very low steady-state error. The first cross-track controller's response is ideal for this, and that is why it is chosen to perform the cross-track control when the aircraft is close to the ground track.

Second Cross-Track Controller Closed-Loop Model

The second cross-track controller closed-loop model is derived by first augmenting the cross-track error state into the closed-loop model of the heading controller, from equation 5.209. The control law of the second cross-track controller is then substituted in and the

resulting model is written in a compact state space form as,

$$\dot{\boldsymbol{x}}_{y2} = \boldsymbol{A}_{y2}\boldsymbol{x}_{y2} + \boldsymbol{B}_{y2}\boldsymbol{u}_{y2} \tag{5.223}$$

with,

$$y = \boldsymbol{C}_{y2}\boldsymbol{x}_{y2} \tag{5.224}$$

where,

$$\boldsymbol{C}_{y2} = \begin{bmatrix} \boldsymbol{0}_{1\times 7} & 1 \end{bmatrix} \tag{5.225}$$

The closed-loop transfer function that relates the cross-track reference to the cross-track error is given as,

$$\frac{y(s)}{y_{ref}(s)} = \boldsymbol{C}_{y2}(s\boldsymbol{I} - \boldsymbol{A}_{y2})^{-1}\boldsymbol{B}_{y2_{ref}} \tag{5.226}$$

where,

$$\boldsymbol{B}_{y2_{ref}} = \boldsymbol{B}_{y2}\begin{bmatrix} 1 & 0 \end{bmatrix}^{\mathsf{T}} \tag{5.227}$$

The complete derivation of the closed-loop system for the second cross-track controller was performed by Le Roux [14].

5.3. Model Predictive Control Design

This section will first present some general theory regarding MPC to gain a better understanding of the design approach. Then, the MPC design for the fixed-wing UAV will be presented.

5.3.1 MPC Theory

Model Predictive Control (MPC) is a range of control methods that uses a model of a process to obtain the control action at each time step by minimising a cost function [63]. The MPC uses the model to predict the process's behaviour into the future over a horizon known as the prediction horizon (n_y). The amount of control actions that the MPC can apply during the prediction horizon, is known as the control horizon (n_u). A simplified overview of the MPC's structure is shown in Figure 5.31. The MPC consists of the plant/process model, optimiser, constraints, and cost function. These components will discussed in the following subsections. In Figure 5.31, k is the current time instant and i is the factor of the sample time T_s. The MPC executes every sample time T_s at which it updates its values and recalculates the ideal control actions. The MPC's main goal is to produce the control actions that will bring the predicted model output $y_m(k+i)$ as close to the reference $r(k+i)$ as possible [63].

MPC is also known as receding horizon control as the prediction horizon moves with

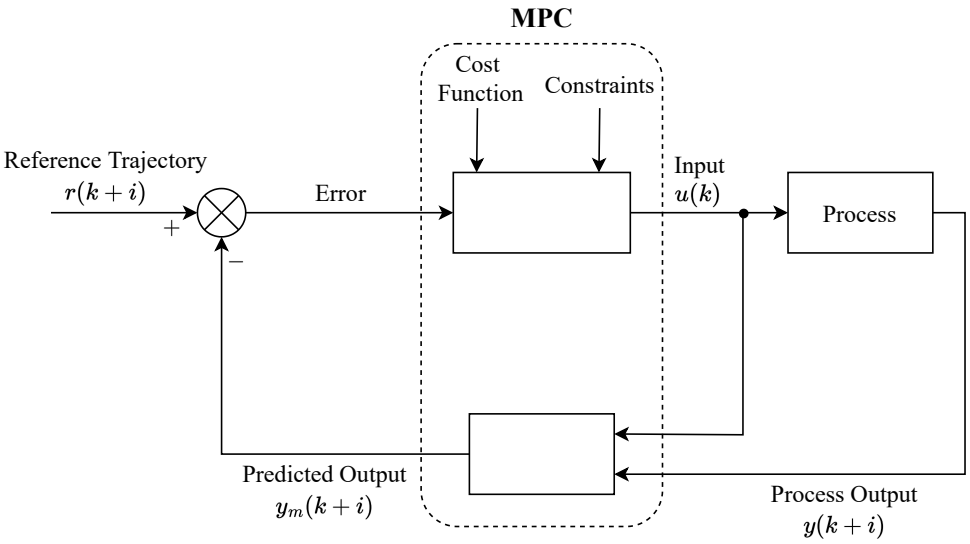


Figure 5.31: MPC structural diagram

the time instants [64]. This movement is due to the predicted outputs and control actions being calculated every time instant. Figure 5.32 illustrates the concept of the receding horizon for the MPC.

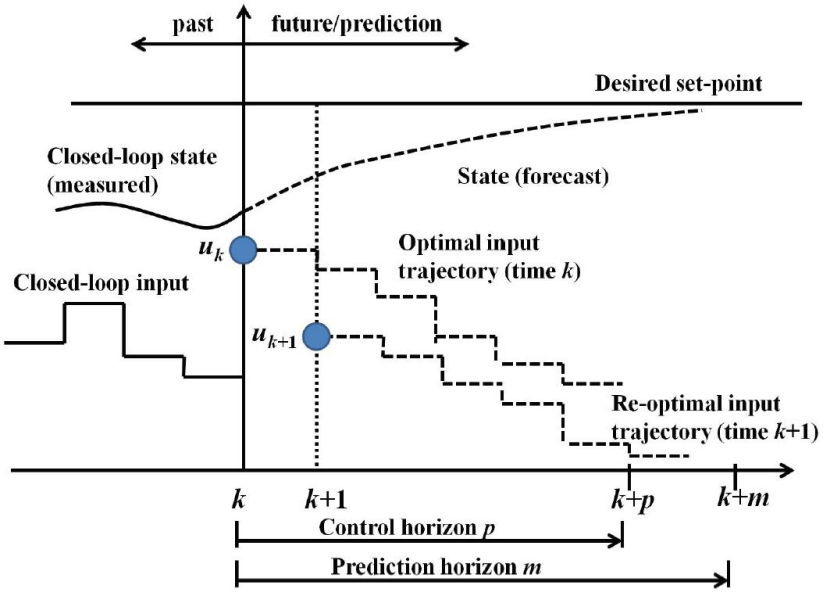


Figure 5.32: Illustration of the MPC receding horizon reproduced from [65].

Using Figure 5.32's notation, for time instant k, the MPC calculates the predicted outputs over the prediction horizon m, based on the control actions calculated over the control horizon p. Only the first control action is applied to the process and the rest are discarded. At the next time instant $k+1$, the MPC will repeat the optimisation process and calculate a new set of control actions. This is computationally inefficient and requires a significant amount of resources which is a reason why MPCs are frequently used for slower processes [66]. The control horizon is subjected to the constraint $p \leq m$ and is

often considerably smaller than the prediction horizon to reduce the MPC's computational complexity.

The implementation of the MPC depends on the type of MPC used, which in this research project is a linear MPC as the plant is a linear model. The MPC architecture used for this research project is based on the architecture used by Amadi [38], who himself based his architecture on the architecture used by Wang [62]. Wang proposed a general MPC architecture which could be applied to many different applications. Amadi applied Wang's MPC architecture to control the angular rates of a multi-rotor UAV (quadcopter) while running on a Pixhawk flight controller. Amadi used the standard PX4 PID controllers to control the other states of the UAV. The Pixhawk has low processing power as it contains an STM32 microcontroller, and therefore it is quite a challenging task to run an MPC on it. The Pixhawk also ran PX4 autopilot software with all the components required to operate the UAV. Amadi had to disable a substantial amount of components in the PX4 software to get the MPC to function correctly. Once he did, the MPC performed exceptionally well with a fast response.

For this research project, it was planned to run the MPC on a Jetson Nano, which is more powerful than the Pixhawk but not as powerful as a desktop PC. Therefore, the MPC architecture had to be efficient, hence why Amadi's MPC architecture was chosen. This research project applied Amadi's MPC architecture to control the altitude and airspeed of a fixed-wing UAV.

The remainder of this section will summarise Amadi's MPC architecture. The components of the MPC that was modified for a fixed-wing UAV will be discussed in Section 5.3.2.

5.3.1.1 State Matrix Augmentation

Consider the discrete state space model below for a general process that the MPC is trying to control at time step k,

$$\boldsymbol{x}_M(k+1) = \boldsymbol{A}_M \boldsymbol{x}_M(k) + \boldsymbol{B}_M \boldsymbol{u}_M(k) \tag{5.228}$$

$$\boldsymbol{y}_M(k) = \boldsymbol{C}_M \boldsymbol{x}_M(k) \tag{5.229}$$

It is desired to have zero steady-state error for the MPC's step response, therefore an integrator needs to be augmented into the MPC's model. Amadi [38] used Wang's [62] augmentation method, whose process will be described. The MPC model can be considered for the difference between time steps which is given as,

$$\boldsymbol{x}_M(k+1) - \boldsymbol{x}_M(k) = \boldsymbol{A}_M(\boldsymbol{x}_M(k) - \boldsymbol{x}_M(k-1)) + \boldsymbol{B}_M(\boldsymbol{u}_M(k) - \boldsymbol{u}_M(k-1)) \tag{5.230}$$

$$\boldsymbol{y}_M(k+1) - \boldsymbol{y}_M(k) = \boldsymbol{C}_M(\boldsymbol{x}_M(k+1) - \boldsymbol{x}_M(k)) \tag{5.231}$$

Letting,

$$\Delta \boldsymbol{x}_{M}(k+1) = \boldsymbol{x}_{M}(k+1) - \boldsymbol{x}_{M}(k) \tag{5.232}$$

$$\Delta \boldsymbol{x}_{M}(k) = \boldsymbol{x}_{M}(k) - \boldsymbol{x}_{M}(k-1) \tag{5.233}$$

$$\Delta \boldsymbol{u}_{M}(k) = \boldsymbol{u}_{M}(k) - \boldsymbol{u}_{M}(k-1) \tag{5.234}$$

and substituting the equations into the MPC difference model and simplifying, results in,

$$\Delta \boldsymbol{x}_{M}(k+1) = \boldsymbol{A}_{M}\Delta \boldsymbol{x}_{M}(k) + \boldsymbol{B}_{M}\Delta \boldsymbol{u}_{M}(k) \tag{5.235}$$

$$\boldsymbol{y}_{M}(k+1) = \boldsymbol{C}_{M}\boldsymbol{A}_{M}\Delta \boldsymbol{x}_{M}(k) + \boldsymbol{C}_{M}\boldsymbol{B}_{M}\Delta \boldsymbol{u}_{M}(k) + \boldsymbol{y}_{M}(k) \tag{5.236}$$

The input of the model becomes $\boldsymbol{u}_{M}(k)$. A new state vector is formed by combining $\Delta \boldsymbol{x}_{M}(k)$ and $\boldsymbol{y}_{M}(k)$ to give,

$$\boldsymbol{x}_{AUG}(k) = \begin{bmatrix} \Delta \boldsymbol{x}_{M}(k)^{T} & \boldsymbol{y}_{M}(k) \end{bmatrix}^{T} \tag{5.237}$$

Combining equations 5.235, 5.236, and 5.237 produces the augmented state space model as,

$$\begin{bmatrix} \Delta \boldsymbol{x}_{M}(k+1) \\ \boldsymbol{y}_{M}(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{M} & \boldsymbol{0}_{n\times q} \\ \boldsymbol{C}_{M}\boldsymbol{A}_{M} & \boldsymbol{I}_{q\times q} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x}_{M}(k) \\ \boldsymbol{y}_{M}(k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_{M} \\ \boldsymbol{C}_{M}\boldsymbol{B}_{M} \end{bmatrix} \Delta \boldsymbol{u}_{M}(k) \tag{5.238}$$

$$\boldsymbol{y}_{M}(k) = \begin{bmatrix} \boldsymbol{0}_{q\times n} & \boldsymbol{I}_{q\times q} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x}_{M}(k) \\ \boldsymbol{y}_{M}(k) \end{bmatrix} \tag{5.239}$$

where q is the number of outputs in the model, n is the number of states in the model, and $\boldsymbol{I}_{q\times q}$ is an identity matrix with dimensions $q \times q$. The augmented state space model can be expressed in a compact form as,

$$\boldsymbol{x}_{AUG}(k+1) = \boldsymbol{A}_{AUG}\boldsymbol{x}_{AUG}(k) + \boldsymbol{B}_{AUG}\Delta \boldsymbol{u}_{AUG}(k) \tag{5.240}$$

$$\boldsymbol{y}_{AUG}(k) = \boldsymbol{C}_{AUG}\boldsymbol{x}_{AUG}(k) \tag{5.241}$$

where,

$$\boldsymbol{y}_{AUG}(k) = \boldsymbol{y}_{M}(k) \;;\; \Delta \boldsymbol{u}_{AUG}(k) = \Delta \boldsymbol{u}_{M}(k) \tag{5.242}$$

5.3.1.2 Output Predictions

The predicted plant output must be calculated to design the MPC with the future control signal used as the adjustable variable(s). Assuming that k is the current time step, the

future control trajectory is given as,

$$\Delta \boldsymbol{U} = \left[\Delta \boldsymbol{u}(k),\, \Delta \boldsymbol{u}(k+1),\, \Delta \boldsymbol{u}(k+2),\, \cdots,\, \Delta \boldsymbol{u}(k+n_u-1)\right]^T \tag{5.243}$$

Amadi [38] showed that the output predictions can be expressed as,

$$\begin{bmatrix} \boldsymbol{y}(k+1) \\ \boldsymbol{y}(k+2) \\ \boldsymbol{y}(k+3) \\ \vdots \\ \boldsymbol{y}(k+n_y) \end{bmatrix} = \begin{bmatrix} \boldsymbol{CA} \\ \boldsymbol{CA}^2 \\ \boldsymbol{CA}^3 \\ \vdots \\ \boldsymbol{CA}^{n_y} \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} \boldsymbol{CB} & 0 & 0 & \cdots & 0 \\ \boldsymbol{CAB} & \boldsymbol{CB} & 0 & \cdots & 0 \\ \boldsymbol{CA}^2\boldsymbol{B} & \boldsymbol{CAB} & \boldsymbol{CB} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{CA}^{n_y-1}\boldsymbol{B} & \boldsymbol{CA}^{n_y-2}\boldsymbol{B} & \boldsymbol{CA}^{n_y-3}\boldsymbol{B} & \cdots & \boldsymbol{CA}^{n_y-n_u}\boldsymbol{B} \end{bmatrix} \Delta \boldsymbol{U} \tag{5.244}$$

where $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are matrices from the augmented model in equations 5.240 and 5.241. The output predictions can be expressed in a more compact form as,

$$\boldsymbol{Y} = \boldsymbol{P}\boldsymbol{x}(k) + \boldsymbol{H}\Delta \boldsymbol{U} \tag{5.245}$$

where,

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{CA} \\ \boldsymbol{CA}^2 \\ \boldsymbol{CA}^3 \\ \vdots \\ \boldsymbol{CA}^{n_y} \end{bmatrix} \;;\; \boldsymbol{H} = \begin{bmatrix} \boldsymbol{CB} & 0 & 0 & \cdots & 0 \\ \boldsymbol{CAB} & \boldsymbol{CB} & 0 & \cdots & 0 \\ \boldsymbol{CA}^2\boldsymbol{B} & \boldsymbol{CAB} & \boldsymbol{CB} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{CA}^{n_y-1}\boldsymbol{B} & \boldsymbol{CA}^{n_y-2}\boldsymbol{B} & \boldsymbol{CA}^{n_y-3}\boldsymbol{B} & \cdots & \boldsymbol{CA}^{n_y-n_u}\boldsymbol{B} \end{bmatrix} \tag{5.246}$$

5.3.1.3 Constraints

The MPC allows constraints to be applied to the input, output, and states of the MPC model. This provides the MPC with the knowledge of any limitations in the process which it will account for when generating the control actions. For this research project, constraints will only be applied to the input (control) variables $\boldsymbol{u}(k)$ and their rate of change $\Delta \boldsymbol{u}(k)$. The constraints on the input variables are represented as,

$$\boldsymbol{u}_{min} \leq \boldsymbol{u}(k) \leq \boldsymbol{u}_{max} \tag{5.247}$$

where $\boldsymbol{u}(k)$ is the input vector containing the input variables at time instance k, $\boldsymbol{u}_{min}$ is the minimum limit for the input variables, and $\boldsymbol{u}_{max}$ is the maximum limit for the input variables. Correspondingly, the constraints for the rate of change of the input variables is represented as,

$$\Delta \boldsymbol{u}_{min} \leq \Delta \boldsymbol{u}(k) \leq \Delta \boldsymbol{u}_{max} \tag{5.248}$$

Amadi [38] showed that the rate of change of input constraints can be expressed in matrix form as,

$$\begin{bmatrix} -\boldsymbol{I} \\ \boldsymbol{I} \end{bmatrix} \Delta \boldsymbol{U} \leq \begin{bmatrix} -\Delta \boldsymbol{U}^{min} \\ \Delta \boldsymbol{U}^{max} \end{bmatrix} \tag{5.249}$$

where $\Delta \boldsymbol{U}$, $\Delta \boldsymbol{U}^{min}$, and $\Delta \boldsymbol{U}^{max}$ are the input variables and corresponding limits over the control horizon. The $\Delta \boldsymbol{U}$ matrix is defined in equation 5.243.

Amadi shows that the input variables can be written to incorporate their rate of change, as follows

$$\begin{bmatrix} \boldsymbol{u}(k) \\ \boldsymbol{u}(k+1) \\ \vdots \\ \boldsymbol{u}(k+n_u-1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{I} \\ \vdots \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{u}(k-1) + \begin{bmatrix} \boldsymbol{I} & 0 & 0 & \cdots & 0 \\ \boldsymbol{I} & \boldsymbol{I} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{u}(k) \\ \Delta \boldsymbol{u}(k+1) \\ \vdots \\ \Delta \boldsymbol{u}(k+n_u-1) \end{bmatrix} \tag{5.250}$$

The input constraints can therefore be written in terms of the rate of change of the input variables as,

$$\begin{aligned} -(\boldsymbol{C}_1 \boldsymbol{u}(k-1) + \boldsymbol{C}_2 \Delta \boldsymbol{U}) &\leq -\boldsymbol{U}^{min} \\ (\boldsymbol{C}_1 \boldsymbol{u}(k-1) + \boldsymbol{C}_2 \Delta \boldsymbol{U}) &\leq \boldsymbol{U}^{max} \end{aligned} \tag{5.251}$$

where,

$$\boldsymbol{C}_1 = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{I} \\ \vdots \\ \boldsymbol{I} \end{bmatrix}, \ \boldsymbol{C}_2 = \begin{bmatrix} \boldsymbol{I} & 0 & 0 & \cdots & 0 \\ \boldsymbol{I} & \boldsymbol{I} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{I} \end{bmatrix} \tag{5.252}$$

The constraints for the input variables and their rate of change are combined as,

$$\begin{bmatrix} \boldsymbol{C}_{\Delta u} \\ \boldsymbol{C}_u \end{bmatrix} \Delta \boldsymbol{U} \leq \begin{bmatrix} \boldsymbol{d}_{\Delta u} \\ \boldsymbol{d}_u \end{bmatrix} \tag{5.253}$$

where,

$$\boldsymbol{C}_u = \begin{bmatrix} -\boldsymbol{C}_2 \\ \boldsymbol{C}_2 \end{bmatrix}; \boldsymbol{d}_u = \begin{bmatrix} -\boldsymbol{U}^{min} + \boldsymbol{C}_1 \boldsymbol{u}(k-1) \\ \boldsymbol{U}^{max} - \boldsymbol{C}_1 \boldsymbol{u}(k-1) \end{bmatrix}; \boldsymbol{C}_{\Delta u} = \begin{bmatrix} -\boldsymbol{I} \\ \boldsymbol{I} \end{bmatrix}; \boldsymbol{d}_{\Delta u} = \begin{bmatrix} -\Delta \boldsymbol{U}^{min} \\ \Delta \boldsymbol{U}^{max} \end{bmatrix} \tag{5.254}$$

The constraints in Equation 5.253 can be compactly expressed as,

$$\boldsymbol{CC} \Delta \boldsymbol{U} \leq \boldsymbol{d} \tag{5.255}$$

where,

$$\boldsymbol{CC} = \begin{bmatrix} \boldsymbol{C}_{\Delta u} \\ \boldsymbol{C}_u \end{bmatrix}; \boldsymbol{d} = \begin{bmatrix} \boldsymbol{d}_{\Delta u} \\ \boldsymbol{d}_u \end{bmatrix} \tag{5.256}$$

5.3.1.4 Cost Function

The cost function is used to balance the reduction of the error between the reference values and predicted outputs with the control actions (input variables) used to achieve the outputs. At time instant k, only one set of reference values $\boldsymbol{r}(k)$ is known and this poses a problem as the error needs to be calculated for the entire prediction horizon n_y. A common solution is to assume that the reference remains the same for the entire prediction horizon, which is represented by the matrix,

$$\boldsymbol{R}_s = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{I} \\ \vdots \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{r}(k) \tag{5.257}$$

This matrix is then used to calculate the error in the cost function. For this research project, assuming a constant reference would be true for most cases. However, when the aircraft captures the glide slope, the altitude reference is a ramp response which changes continuously. The future altitude reference for the glide slope can still be easily calculated as the glide slope angle is constant. This will be shown in section 5.3.2.4.

The cost function used by Amadi [38] is derived from Wang [62] and is given below,

$$J = (\boldsymbol{R}_s - \boldsymbol{Y})^T(\boldsymbol{R}_s - \boldsymbol{Y}) + \Delta\boldsymbol{U}^T\boldsymbol{W}\Delta\boldsymbol{U} \tag{5.258}$$

The first term in the cost function minimises the error between the reference values and the predicted output, while the second term minimises the control actions $\Delta\boldsymbol{U}$. $\boldsymbol{W}$ is a diagonal weighting matrix used to tune the MPC response. High weight values in the matrix prioritises minimising the control action magnitudes $\Delta\boldsymbol{U}$ at the expense of a slower response. Low weight values in the matrix prioritises minimising the error between the reference and predicted outputs at the expense of large control action magnitudes ($\Delta\boldsymbol{U}$).

The predicted output from Equation 5.245 is substitute into the cost function which results in,

$$J = (\boldsymbol{R}_s - \boldsymbol{P}\boldsymbol{x}(k))^T(\boldsymbol{R}_s - \boldsymbol{P}\boldsymbol{x}(k)) - 2\Delta\boldsymbol{U}^T\boldsymbol{H}^T(\boldsymbol{R}_s - \boldsymbol{P}\boldsymbol{x}(k)) + \Delta\boldsymbol{U}^T(\boldsymbol{H}^T\boldsymbol{H} + \boldsymbol{W})\Delta\boldsymbol{U} \tag{5.259}$$

5.3.1.5 Optimiser

The optimiser minimises the cost function in Equation 5.259 to obtain the control actions $\Delta\boldsymbol{U}$ to apply to system, while adhering to the constraints. The optimisation first consists

of taking the first derivative of the cost function which results in,

$$J = \frac{1}{2}\Delta \boldsymbol{U}^T \boldsymbol{E} \Delta \boldsymbol{U} + \Delta \boldsymbol{U}^T \boldsymbol{F} \tag{5.260}$$

with the constraints $\boldsymbol{CC}\Delta \boldsymbol{U} \leq \boldsymbol{d}$. $\boldsymbol{E}$ and $\boldsymbol{F}$ are matrices used by the optimiser and have the following expressions,

$$\boldsymbol{E} = 2(\boldsymbol{H}^T \boldsymbol{H} + \boldsymbol{W}) \; ; \; \boldsymbol{F} = -2\boldsymbol{H}^T(\boldsymbol{R}_s - \boldsymbol{P}\boldsymbol{x}(k)) \tag{5.261}$$

The minimisation of the cost function is a quadratic programming problem, which can be solved using many different methods. However, Amadi chose Wang's primal-dual method. This method consists of systematically eliminating constraints. Wang [62] derives the dual problem equation as,

$$\min_{\lambda \geq 0}(\frac{1}{2}\lambda^T \boldsymbol{T}\lambda + \lambda^T \boldsymbol{K} + \frac{1}{2}\boldsymbol{d}^T \boldsymbol{E}^{-1}\boldsymbol{d}) \tag{5.262}$$

where λ is a vector of Lagrange multipliers while the $\boldsymbol{T}$ and $\boldsymbol{K}$ matrices are,

$$\boldsymbol{T} = \boldsymbol{CC}\boldsymbol{E}^{-1}\boldsymbol{CC}^T \tag{5.263}$$

$$\boldsymbol{K} = \boldsymbol{d} + \boldsymbol{CC}\boldsymbol{E}^{-1}\boldsymbol{F} \tag{5.264}$$

The dual programming problem is solved using an algorithm called the Hildreth's quadratic programming procedure. This procedure is the optimiser and its implementation can found in Wang's textbook [62] or Amadi's thesis [38]. The output of the optimiser is the control actions over the control horizon. However, only the first control action is applied to the process, and the rest are discarded.

5.3.2 Fixed-Wing UAV MPC Design

The MPC controller replaces the classical altitude and airspeed controllers for the landing phase of flight to improve the landing accuracy performance. The MPC controls the altitude and airspeed states simultaneously by producing the thrust command from trim and climb rate reference respectively, based on references it receives from the guidance control system. The MPC architecture is based on the architecture used by Amadi [38]. However, it was modified to control the altitude and airspeed states. Figure 5.33 outlines the MPC operation. The MPC uses a model of the aircraft to the predict aircraft's future behaviour. The optimiser uses the cost function, constraints, and the aircraft model to calculate the optimal control actions (climb rate reference and thrust command from trim) based on the altitude and airspeed references it receives. Hildreth's quadratic programming method is used as the optimiser for the MPC.

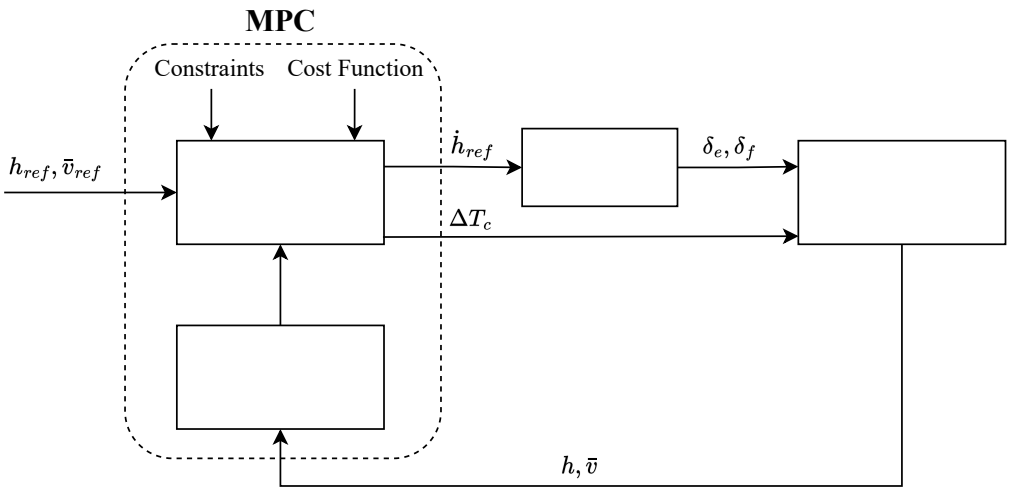


Figure 5.33: Block diagram showing the overview of the MPC controller.

5.3.2.1 MPC Parameters

The MPC has three selectable parameters that influence its performance, namely the sample time T_s, prediction horizon n_y and control horizon n_u. These parameters must be chosen to obtain desirable MPC performance while not being too computationally expensive. The parameters were initially going to be chosen based on rules of thumb obtained from the Mathworks white paper [67]. However, this project's MPC plant behaviour did not match the behaviour expected by the paper. Therefore the rules of thumb were adjusted to suit this project's MPC application. For the sample time, the rule of thumb is that there should be 10 to 20 samples within the rise time, which is expressed mathematically as,

$$\frac{T_r}{20} \leq T_s \leq \frac{T_r}{10} \tag{5.265}$$

where T_r is the rise time of the open-loop step response. This project's MPC open-loop system produces a ramp response for a step reference and is therefore unstable. As an open-loop step response cannot be obtained, it is instead decided to use the desired rise time of the MPC closed-loop step response in the equation. The desired rise time is 2 seconds for both the altitude and airspeed step responses, with 20 samples within the rise time. The sample time is therefore calculated as,

$$T_s = \frac{2}{20} = 0.1 \text{ seconds} \tag{5.266}$$

The rule of thumb for the prediction horizon is that the prediction time should be greater than or equal to the settling time of the open-loop step response, which is expressed as,

$$n_y T_s \geq T_{\text{settling}} \tag{5.267}$$

As the open-loop step response was unattainable, the desired settling time for the altitude and airspeed step responses was used instead. The desired settling time was 2.5 seconds. The prediction horizon was therefore calculated to be,

$$n_y = \frac{2.5}{0.1} = 25 \text{ samples} \tag{5.268}$$

The prediction horizon should be limited to between 20 to 30 samples. The calculated prediction horizon is 25 samples, which is within these bounds. The control horizon's rule of thumb is that it should be between 10% to 20% of the prediction horizon, which is mathematically expressed as,

$$0.1n_y \leq n_u \leq 0.2n_y \tag{5.269}$$

The limit is chosen as 20% of the prediction horizon. Therefore,

$$n_u = 0.2 \times 25 = 5 \text{ samples} \tag{5.270}$$

The control horizon should be more than 2 samples, which means that the calculated value of 5 samples satisfies the rule of thumb. The sample time determines the period at which the MPC runs. The prediction and control horizons influence the MPC's matrices sizes, which impact its computational complexity.

5.3.2.2 MPC Plant Model

The plant used by the MPC consists of the full longitudinal model augmented with only the NSADLC and climb rate controller. The thrust lag characteristics are also reaugmented to improve the model accuracy. The thrust lag model from equation 4.55 is restated below,

$$\dot{T} = -\frac{K_T}{\tau_e}T + \frac{K_{T_c}}{\tau_e}T_c \tag{5.271}$$

It was assumed that the thrust commanded was the thrust produced for the classical airspeed controller as it had an integrator to compensate for any discrepancies. The MPC does not have such an integrator therefore the thrust model must more accurately represent the real thrust dynamics for the MPC to work correctly on the practical vehicle. The method used to obtain the thrust constants is discussed in appendix B.5.1. Using this method, the thrust magnitude constant K_T and thrust command constant K_{T_c} was set as,

$$\begin{aligned} K_T &= 1 \\ K_{T_c} &= 2.57975 \end{aligned} \tag{5.272}$$

The thrust lag model with these constants was augmented into the linear longitudinal model from equation 4.107 to give,

$$\begin{bmatrix} \dot{\bar{v}} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \\ \Delta\dot{T} \end{bmatrix} = \begin{bmatrix} \frac{\partial \dot{U}}{\partial U} & \overline{V}_T \frac{\partial \dot{U}}{\partial W} & \frac{\partial \dot{U}}{\partial Q} & \frac{\partial \dot{U}}{\partial \Theta} & \frac{\partial \dot{U}}{\partial T} \\ \frac{1}{\overline{V}_T}\frac{\partial \dot{W}}{\partial U} & \frac{\partial \dot{W}}{\partial W} & \frac{1}{\overline{V}_T}\frac{\partial \dot{W}}{\partial Q} & \frac{1}{\overline{V}_T}\frac{\partial \dot{W}}{\partial \Theta} & \frac{1}{\overline{V}_T}\frac{\partial \dot{W}}{\partial T} \\ \frac{\partial \dot{Q}}{\partial U} & \overline{V}_T \frac{\partial \dot{Q}}{\partial W} & \frac{\partial \dot{Q}}{\partial Q} & \frac{\partial \dot{Q}}{\partial \Theta} & \frac{\partial \dot{Q}}{\partial T} \\ \frac{\partial \dot{\Theta}}{\partial U} & \overline{V}_T \frac{\partial \dot{\Theta}}{\partial W} & \frac{\partial \dot{\Theta}}{\partial Q} & \frac{\partial \dot{\Theta}}{\partial \Theta} & \frac{\partial \dot{\Theta}}{\partial T} \\ 0 & 0 & 0 & 0 & -\frac{K_T}{\tau_e} \end{bmatrix} \begin{bmatrix} \bar{v} \\ \alpha \\ q \\ \theta \\ \Delta T \end{bmatrix} + \begin{bmatrix} \frac{\partial \dot{U}}{\partial \delta_E} & \frac{\partial \dot{U}}{\partial \delta_F} & 0 \\ \frac{1}{\overline{V}_T}\frac{\partial \dot{W}}{\partial \delta_E} & \frac{1}{\overline{V}_T}\frac{\partial \dot{W}}{\partial \delta_F} & 0 \\ \frac{\partial \dot{Q}}{\partial \delta_E} & \frac{\partial \dot{Q}}{\partial \delta_F} & 0 \\ \frac{\partial \dot{\Theta}}{\partial \delta_E} & \frac{\partial \dot{\Theta}}{\partial \delta_F} & 0 \\ 0 & 0 & \frac{K_{T_c}}{\tau_e} \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_f \\ \Delta T_c \end{bmatrix} \tag{5.273}$$

which can be written in a compact state space form as,

$$\dot{\boldsymbol{x}}_{Long3} = \boldsymbol{A}_{Long3}\boldsymbol{x}_{Long3} + \boldsymbol{B}_{Long3}\boldsymbol{u}_{Long3} \tag{5.274}$$

The NSADLC controller was then augmented into this model using the same gains and method as equations 5.58 to 5.66 to give,

$$\dot{\boldsymbol{x}}_{Hyb2} = \boldsymbol{A}_{Hyb2}\boldsymbol{x}_{Hyb2} + \boldsymbol{B}_{Hyb2}\boldsymbol{u}_{Hyb2} \tag{5.275}$$

Thereafter the climb rate controller was augmented into equation 5.275 using the gains and method from equations 5.84 and 5.85 to give,

$$\dot{\boldsymbol{x}}_{\dot{h}2} = \boldsymbol{A}_{\dot{h}2}\boldsymbol{x}_{\dot{h}2} + \boldsymbol{B}_{\dot{h}2}\boldsymbol{u}_{\dot{h}2} \tag{5.276}$$

The ”2” subscript is used to distinguish these models from the classical controller counterparts as these models do not include the airspeed controller. The altitude state was then augmented into this model so it can be used by the MPC resulting in,

$$\begin{bmatrix} \dot{\boldsymbol{x}}_{\dot{h}2} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{\dot{h}2} & & & & \boldsymbol{0}_{8\times1} \\ 0 & -V_T & 0 & V_T & \boldsymbol{0}_{1\times5} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{\dot{h}2} \\ h \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_{\dot{h}2} \\ \boldsymbol{0}_{1\times2} \end{bmatrix} \boldsymbol{u}_{\dot{h}2} \tag{5.277}$$

which can be written in a compact state space form as,

$$\dot{\boldsymbol{x}}_{MPC} = \boldsymbol{A}_{MPC}\boldsymbol{x}_{MPC} + \boldsymbol{B}_{MPC}\boldsymbol{u}_{MPC} \tag{5.278}$$

with,

$$\begin{bmatrix} h \\ \bar{v} \end{bmatrix} = \boldsymbol{C}_{MPC}\boldsymbol{x}_{MPC} \tag{5.279}$$

where,

$$\boldsymbol{C}_{MPC} = \begin{bmatrix} \boldsymbol{0}_{1\times8} & 1 \\ 1 & \boldsymbol{0}_{1\times8} \end{bmatrix} \tag{5.280}$$

This MPC plant model was then discretised using the sample time T_s so that the MPC can use the model for every time iteration. The MPC model also undergoes the augmentation process described in Section 5.3.1.1 so that the MPC optimiser matrices can be calculated.

5.3.2.3 Constraints

There are three constraints that can be applied to the MPC, namely the input, output and state constraints. The input constraint limit the control actions ($\dot{h}_{ref}$ and ΔT_c) produced by the MPC, while the output constraint limit the output variables regulated by the MPC (h, $\bar{v}$), and the state constraints limit the state variables in the model ($\boldsymbol{x}_{_{MPC}}$) used by the MPC. The output constraints are not applied to the MPC as the altitude and airspeed can vary to any value. The state constraints are also not applied, as other controllers control these states and therefore the MPC should not limit those controllers. Only the input constraints are applied to the climb rate reference and thrust command from trim, as they are limited by the physical capabilities of the aircraft. The input constraints that are applied are specified as,

$$-2\,\text{m/s} \leq \dot{h}_{ref} \leq 2\,\text{m/s} \tag{5.281}$$

$$-26.5513\,\text{N} \leq \Delta T_c \leq 13.4487\,\text{N} \tag{5.282}$$

$$-1.2 \leq \Delta\dot{h}_{ref} \leq 1.2 \tag{5.283}$$

$$-12 \leq \Delta\Delta T_c \leq 12 \tag{5.284}$$

where the $\Delta\dot{h}_{ref}$ and $\Delta\Delta T_c$ are the rates of the climb rate reference and thrust command from trim respectively. The climb rate reference limit is selected to prevent excessive strain on the aircraft while still allowing the aircraft to follow the glide slope. The thrust command limit is set to constrain the motor thrust between 0 N to 40 N, as this is the thrust range the motor can physically produce, as highlighted in Appendix A.2. As the MPC thrust command is added to the trim value of $T_T = 26.5513$ N (Equation 4.83), the MPC thrust command range therefore corresponds to the physical motor thrust range. These climb rate reference and thrust command limits also apply to the classical altitude and airspeed controllers. However, they are not considered during the design phase, but only for the non-linear simulation and practical phases. The input rate limits ($\Delta\dot{h}_{ref}$ and $\Delta\Delta T_c$) are selected to be 60% of the amplitude of the input limits (Equations 5.281 and 5.282). The constraint values are used to form the $\boldsymbol{CC}$ and $\boldsymbol{d}$ matrices from Equation 5.256 to be used by the optimiser.

5.3.2.4 Cost Function and Optimiser

The cost function used for the MPC of this research project is the same as Amadi's [38] from Equation 5.259. During the normal phases of flight, it is safe to assume that the

altitude and airspeed references are constant for the entire prediction horizon. However as previously mentioned, during the glide slope capture phase, the altitude reference (h_{ref}) follows a ramp response. This requires the reference matrix $\boldsymbol{R}_s$ used in the cost function to account for the ramp for future references in the prediction horizon. Let the altitude reference at time instant k be,

$$r_1(k) = h_{ref} \tag{5.285}$$

As the glide slope angle is constant, the future prediction can be calculated using the equation,

$$r_1(k+i) = r_1(k) + i\tan(-\gamma)T_s V_g \qquad i \in \{0, 1, \cdots, n_y - 1\} \tag{5.286}$$

where V_g is the aircraft's ground speed. The altitude references are added to the reference matrix $\boldsymbol{R}_s$.

The optimiser's goal is to find the input signals (climb rate reference and thrust command from trim) that minimise the cost function. The optimiser consists of the Hildreth's quadratic programming procedure used by Amadi. The $\boldsymbol{E}$ and $\boldsymbol{F}$ matrices from Equation 5.261 are calculated to be used by Hildreth's algorithm.

5.3.2.5 MPC Tuning

The MPC is tuned by adjusting the weights in the weight matrix $\boldsymbol{W}$. The $\boldsymbol{W}$ matrix is a square diagonal matrix that contains the weights for both input variables (climb rate reference and thrust command from trim) for each time step over the control horizon. The $\boldsymbol{W}$ matrix therefore has a size of $2n_u \times 2n_u$. The weights of the input variables are kept the same for all the time steps in the control horizon and only differ from one another. The weight matrix has the form,

$$\begin{bmatrix}
w_{\dot{h}_{ref}1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & w_{\Delta T_c 1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & w_{\dot{h}_{ref}2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & w_{\Delta T_c 2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & w_{\dot{h}_{ref}3} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & w_{\Delta T_c 3} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & w_{\dot{h}_{ref}4} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{\Delta T_c 4} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{\dot{h}_{ref}5} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{\Delta T_c 5}
\end{bmatrix} \tag{5.287}$$

where $w_{\dot{h}_{ref}i}$ is the climb rate reference weight, $w_{\Delta T_c i}$ is the thrust command weight, and $i \in 1, ..., n_u$ is the time step. The $w_{\dot{h}_{ref}i}$ weights are set to the same value and the $w_{\Delta T_c i}$ weights are also set to the same value. When the weights are set to low values, the state

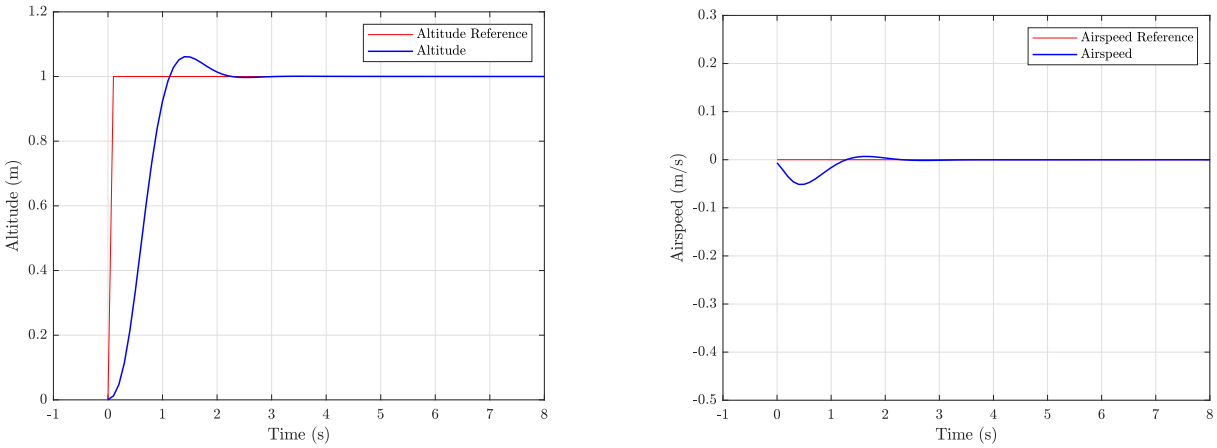
errors are quickly minimised at the expense of large input variable commands. When the weights are set to high values, the input variable commands are minimised which produces a slower response. It is therefore important to choose the correct weights to obtain desirable performance while not producing aggressive input variable commands.

Initially, the MPC was tuned with weights that produced step responses with significantly better performance than the classical altitude and airspeed controllers in simulation. However, on the practical vehicle, the MPC was too aggressive as its step responses had excessive oscillation. This was due to the MPC being so reliant on the plant model. Therefore, any inaccuracies in the model would have a significant impact on the MPC performance. These inaccuracies occur in the real world due to uncertainties. The MPC was therefore retuned to be less aggressive with the weights set to the following,

$$w_{\dot{h}_{ref}i} = 7.5e-1 \tag{5.288}$$

$$w_{\Delta T_c i} = 7.5e-2 \tag{5.289}$$

The MPC altitude and airspeed step responses with these weights are shown in Figure 5.34 and Figure 5.35 respectively. It should be noted that when obtaining the step response for one of the states (for example, altitude), the reference for the other state (airspeed) was kept at a constant value to not affect the results. This same procedure was followed to obtain the MPC's step responses in the non-linear simulations and on the practical vehicle.



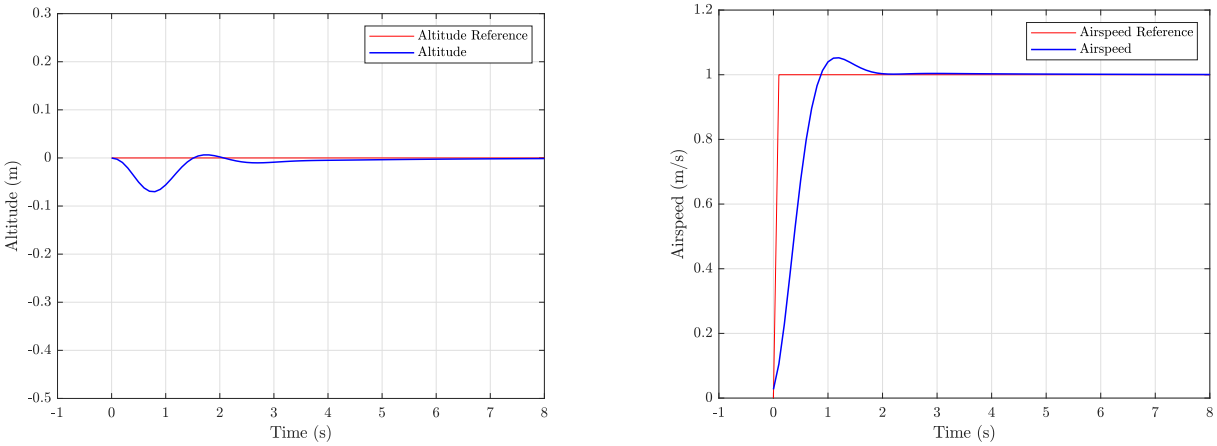
(a) Altitude Response for Altitude Step

(b) Airspeed Response for Altitude Step

Figure 5.34: MPC controller altitude and airspeed responses for an altitude step

The MPC altitude response for the altitude step, shown in Figure 5.34a, has a rise time of 0.99 seconds, an overshoot of 6.2%, zero steady-state error, and a 2% settling time of 1.92 seconds. These characteristics are within the requirements specified in the classical altitude controller design section. The rise time and settling time of the MPC are much better then the classical altitude controller at the expense of increased overshoot. This increased overshoot is acceptable as it is within the 20% overshoot limit. The airspeed

response for the altitude step, shown in Figure 5.34b, is minimally affected by the altitude step, as the MPC was able to maintain the airspeed close to its reference while responding to the altitude step.



(a) Altitude Response for Airspeed Step **(b)** Airspeed Response for Airspeed Step

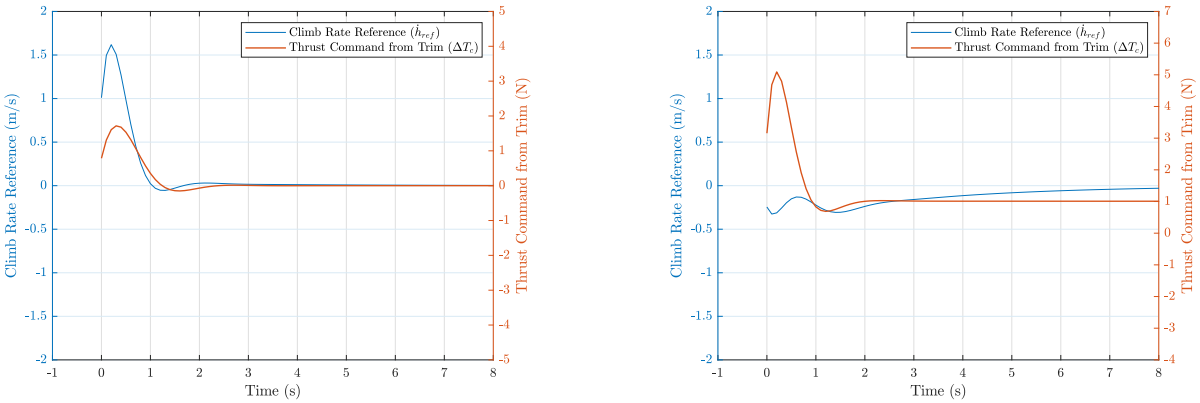
Figure 5.35: MPC controller altitude and airspeed responses for airspeed step

The MPC airspeed response for the airspeed step, shown in Figure 5.35b, has a rise time of 0.7 seconds, an overshoot of 5.3%, zero steady-state error, and a 2% settling time of 1.6 seconds. These values are within the requirements specified in the classical airspeed controller design section. The MPC rise time is faster than the classical airspeed controller at the expense of increased overshoot. However, the 2% settling time is the same for both controllers. The altitude response for the airspeed step, shown in Figure 5.35a, is minimally affected by the airspeed step, as the MPC was able to maintain the altitude close to its reference while responding to the airspeed step.

The reason the MPC has faster response times compared to the corresponding classical controllers is because the MPC is able to use both states to minimise the error. This behaviour can be observed for the altitude response for the airspeed step, shown in Figure 5.35a, where the MPC slightly decreases the altitude to increase the airspeed, which assists in reducing the airspeed error. When the airspeed error becomes small, the MPC will then increase the altitude back to its reference. Similar behaviour occurs with large altitude steps where the airspeed will change its value to minimise the altitude error. The 1 m altitude step in Figure 5.34 is too small to show this effect, however, this behaviour does occur for larger steps (more than 10 m).

These results show that, at least for the linear model, the MPC has better performance then the classical altitude and airspeed controller.

The climb rate reference ($\dot{h}_{ref}$) and thrust command from trim (ΔT_c) commanded by the MPC for the altitude and airspeed steps are shown in Figure 5.36. The MPC's commands are well within the limits applied to them. Therefore, the MPC should be physically realisable. The MPC commands are far from their limits, which indicates



(a) MPC Commands for Altitude Step **(b)** MPC Commands for Airspeed Step

Figure 5.36: MPC controller commands for altitude and airspeed steps

that the MPC could be tuned further to obtain improved performance. As previously mentioned, an MPC was designed with superior performance to the current MPC, however, it caused excessive oscillation on the physical UAV. While the current MPC might not fully utilise its potential, it is a safer tune to use on the physical UAV and should reduce oscillations.

5.3.2.6 MPC Limited integrator

A limited integrator is added in parallel with the MPC as shown in Figure 5.37.

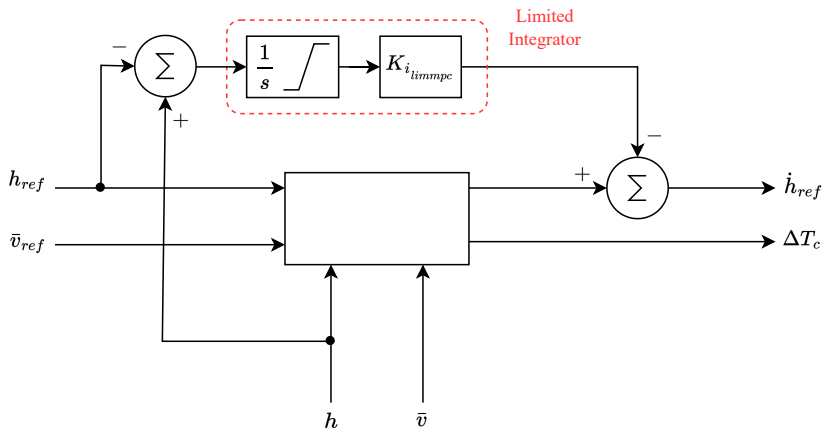


Figure 5.37: Block diagram showing the limited integrator in parallel with the MPC controller.

The integrator is used to assist the MPC in altitude control to enable the aircraft to follow a ramp reference, which is the glide slope during the landing phase. This is similar to the limited integrator used on the classical altitude controller. Without the limited integrator, the MPC cannot follow a ramp reference as it has a steady-state error. This occurs because when the aircraft starts following the downwards ramp, its airspeed begins to increase and this causes the MPC to want to increase its altitude, which it cannot do due to its altitude reference, resulting in conflicting behaviour. The MPC eventually

becomes satisfied with slight-steady state errors in both airspeed and altitude, as these states reach an equilibrium. The limited integrator eliminates the altitude steady-state error by providing an additional climb rate command which is combined with the MPC climb rate command output. This does result in an increase in the airspeed steady state error. However, this is deemed acceptable as the wind will cause the airspeed to vary in any case. The airspeed is not required to be at a specific value for landing, as long as it is above the stall speed. This is due to the ground speed being used instead to configure the aircraft for landing. The gain used for the limited integrator is,

$$K_{i_{limmpc}} = 0.65 \tag{5.290}$$

where this value was obtained through trail and error using non-linear simulations. The limit for the integrator is set to ± 2 m/s. The limited integrator is not active when obtaining the MPC altitude and airspeed step responses. It is only activated during the landing tests.

5.4. Summary

The flight control system combined classical control with model predictive control, and its design was presented in this chapter. The classical controllers' design was split into the longitudinal and lateral groups and utilised successive loop closure methods. The MPC was added to replace the classical airspeed and altitude controllers to obtain a more precise landing. All the components of the MPC were considered in this chapter, and the results of the MPC showed that it had superior performance compared to its classical controller counterparts. Now that the flight control system (FCS) has been designed to control the local states of the aircraft, the next chapter will focus on the design of the guidance control system, which provides references to the FCS to allow the aircraft to follow a trajectory and land on the moving platform.

Chapter 6

Guidance Control System Development

The Guidance Control System (GCS), which is developed in this chapter, is used to guide the aircraft to follow a trajectory around the airfield and land on a moving platform by providing references and states to the FCS. The GCS consists of the guidance algorithm, waypoint scheduler, landing position predictor, and state machine. These individual components are presented in the following sections.

6.1. Guidance Algorithm

The trajectory that the aircraft follows around the airfield is defined by a set of predefined waypoints and straight line segments between these waypoints. The waypoints consists of North and East coordinates measured in meters and are specified in the inertial frame. A straight line segment between a set of waypoints is called the ground track. Figure 6.1 shows the ground track formed between a source and destination waypoint.

The cross-track controllers, discussed in the sections 5.2.2.4 and 5.2.2.8, were designed to minimise the cross-track error to ensure that the aircraft can stay on the ground track. These controllers therefore require the cross-track error value to perform their function. The cross-track error is defined as the perpendicular distance from the aircraft to the ground track [AA833 course notes]. To easily obtain the cross-track error, a guidance frame, also known as the guidance axis system, is defined which relates the aircraft's position to the ground-track as shown in Figure 6.1.

The guidance frame's origin is placed at the source waypoint with the X-axis aligned with the ground track and pointing towards the destination waypoint. The guidance frame is related to the inertial frame by the ground track heading ψ_{track} which is calculated using,

$$\tan(\psi_{\text{track}}) = \frac{E_{\text{dest}} - E_{\text{src}}}{N_{\text{dest}} - N_{\text{src}}} \tag{6.1}$$

where $(E_{\text{dest}}, N_{\text{dest}})$ are the destination waypoint coordinates and $(E_{\text{src}}, N_{\text{src}})$ are the source waypoint coordinates.

A rotation matrix is used to convert aircraft's position to the guidance frame so that

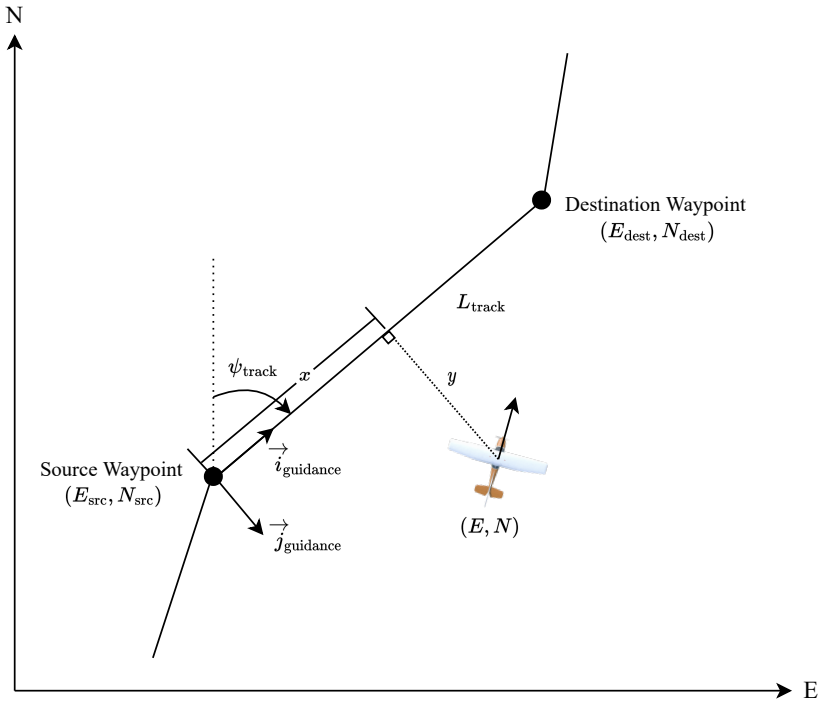


Figure 6.1: Block diagram showing the guidance frame [adapted from AA833 course notes].

the cross-track error can be obtained. This calculation is performed using the equation,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos(\psi_{\text{track}}) & \sin(\psi_{\text{track}}) \\ -\sin(\psi_{\text{track}}) & \cos(\psi_{\text{track}}) \end{bmatrix} \begin{bmatrix} N - N_{\text{src}} \\ E - E_{\text{src}} \end{bmatrix} \tag{6.2}$$

where x is the in-track distance and y is the cross-track error. The in-track distance is defined as the distance of the aircraft's projection onto the ground track from the source waypoint [AA833 course notes]. The track length L_{track} is calculated using,

$$L_{\text{track}} = \sqrt{(N_{\text{dest}} - N_{\text{src}})^2 + (E_{\text{dest}} - E_{\text{src}})^2} \tag{6.3}$$

The in-track distance and the track length are used by the waypoint scheduler to switch waypoints. The first cross-track controller also requires the cross-track error rate ($\dot{y}$) to function and this is calculated using,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos(\psi_{\text{track}}) & \sin(\psi_{\text{track}}) \\ -\sin(\psi_{\text{track}}) & \cos(\psi_{\text{track}}) \end{bmatrix} \begin{bmatrix} \dot{N} \\ \dot{E} \end{bmatrix} \tag{6.4}$$

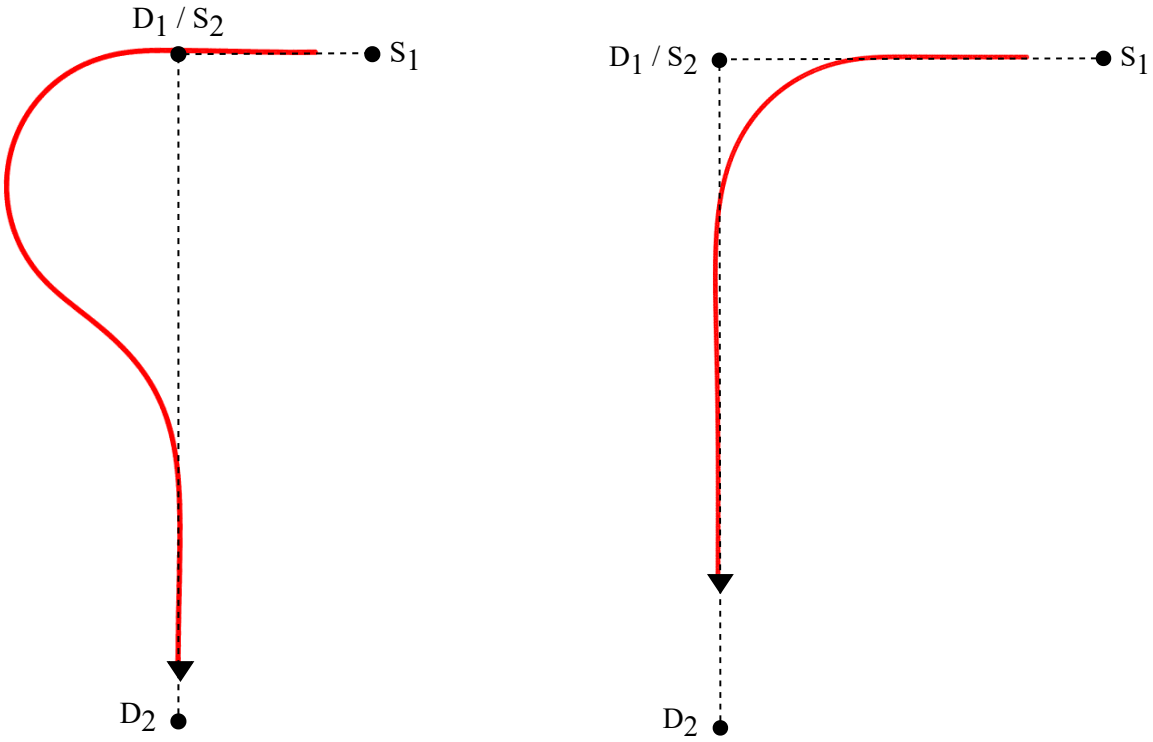
where $\dot{N}$ and $\dot{E}$ are the aircraft ground velocity components in the inertial frame. On the practical vehicle, the aircraft position and velocity in the inertial frame, which is used by the guidance algorithm, is obtained from the EKF which uses DGPS measurements.

6.2. Waypoint Scheduler

The waypoint scheduler's main function is to determine which waypoints are currently being tracked by selecting the source and destination waypoints which are sent to the guidance algorithm. The guidance algorithm then works out the corresponding states of the aircraft with respect to the ground track which is formed from these waypoints, as highlighted in the previous section.

The waypoint scheduler selects the next set of waypoints in the waypoint list if the aircraft has reach the end of the current ground track. It does this by comparing the aircraft's in-track distance x to the track length L_{track}. Initially, it was decided to switch the waypoints when the in-track distance was greater than the the track length. However, this resulted in the aircraft overshooting the next ground track, increasing the cross-track error as shown in Figure 6.2a. The cross-track controllers would have to first deal with the overshoot before they can begin to minimise the cross-track error. This would take a significant amount of the in-track distance of the new ground track segment and hence time which the cross-track controllers, unfortunately, do not have. This would result in the cross-track error not being small enough in time for the landing. A solution to this problem, as suggested by Le Roux [14], is to perform early switching when the aircraft is close to the destination waypoint, by looking ahead and checking when the aircraft's in-track distance is within a specified range from the target waypoint. This allows the cross-track controllers to immediately minimise the cross-track error for the next ground track segment, and focus on improving the lateral accuracy for landing, as shown in Figure 6.2b. The specific distance to the destination waypoint at which early switching should be performed was obtained by considering the aircraft's maximum roll angle and forward speed. A distance of 75m from the destination waypoint was chosen as the distance at which the aircraft must switch to the next waypoint. This distance was determined through trail and error in simulation. The waypoint scheduler performs three additional secondary functions to assist the other components in the GCS, namely:

- Resetting the current waypoint to the first waypoint in the waypoint list when the abort command is called by the state machine.
- When the aircraft is not in the land sub-mode, the final waypoint is reset to the first waypoint when its ground track is completed, so that the aircraft can circuit continuously around the airfield.
- When the aircraft is in the land sub-mode and the aircraft is in line with the runway, the waypoint scheduler sends a command to the state machine letting it know that the aircraft can land.



(a) Standard waypoint switching method where the waypoints are switched after reaching the destination waypoint.

(b) Look-ahead waypoint switching method where the waypoints are switched a certain distance before reaching the destination waypoint.

Figure 6.2: A comparison between the standard and look-ahead waypoint switching methods.

6.3. Landing Position Predictor

The landing position predictor calculates the predicted touchdown point between the fixed-wing UAV and the moving platform. The prediction algorithm is a simple projected touchdown point technique proposed by Le Roux [14] which uses the aircraft and moving platform positions and velocities. Consider an aircraft and moving platform moving at different speeds in the same direction, as shown in Figure 6.3. In the figure, $\overline{V}_a$ is the horizontal component of the aircraft ground velocity, $\overline{V}_p$ is the moving platform horizontal velocity, d_a is the distance from the aircraft to the touchdown point, d_p is the distance from the moving platform to the touchdown point, d_{ap} is the instantaneous relative distance between the aircraft and the platform, h_{vp} is the virtual platform altitude, and γ is the glide slope angle. The aircraft is assumed to be behind the moving platform and is moving faster then it, therefore $\overline{V_a} > \overline{V_p}$, which means that the aircraft will eventually catch up to the platform. It can be assumed that the aircraft is on the glide slope for the majority of the time when the prediction algorithm is used, as the moving platform only moves when the aircraft is on final approach.

For the aircraft and moving platform to meet at the same point for touch down, they both need to cover a distance d_a and d_p respectively within a time Δt. Therefore, the time

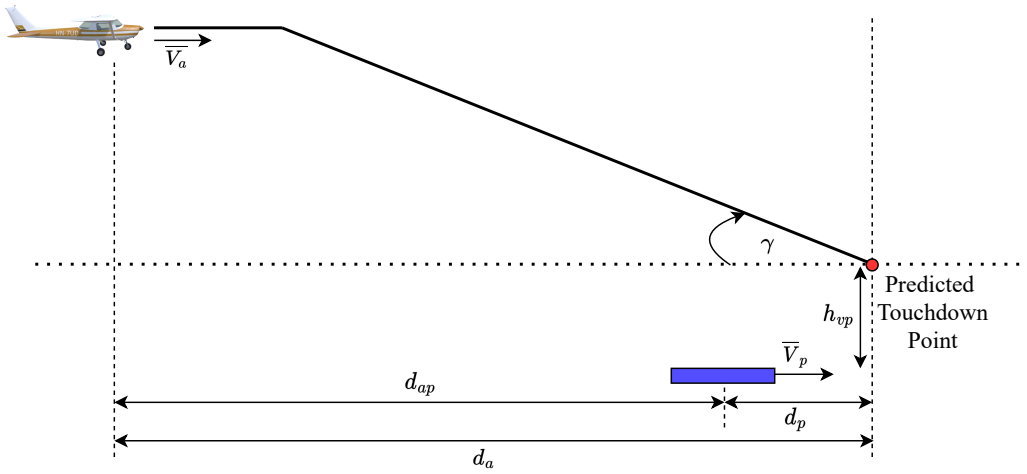


Figure 6.3: Diagram showing the projection of the touchdown point between the aircraft and moving platform.

it takes the aircraft to reach the moving platform is calculated as,

$$\Delta t = \frac{d_{ap}}{\overline{V}_a - \overline{V}_p} \tag{6.5}$$

where Δt is the time it takes for the aircraft to reach the moving platform, d_{ap} is the instantaneous relative distance between the aircraft and the platform, and $(\overline{V}_a - \overline{V}_p)$ is the relative speed between the aircraft and the platform. The distance between the moving platform and the predicted touchdown point is calculated as,

$$d_p = \overline{V}_p \Delta t \tag{6.6}$$

This distance defines the touchdown point with respect to the moving platform's position.

On the practical UAV, the relative position and velocity between the UAV and moving platform are obtained from the DGPS measurements. Originally, the aircraft's horizontal velocity component was used to obtain the time to touchdown, however, this resulted in unusual behaviour. Consider an aircraft descending on the glide slope and having a slight increase in velocity. The time it takes to reach the moving platform decreases and hence the touchdown point moves closer. The glide slope therefore changes position and the aircraft needs to descend to capture the new glide slope. This descent further increases the aircraft's velocity repeating the behaviour. This cyclic behaviour can be broken by setting a constant velocity that the aircraft should maintain on the glide slope, and this velocity is selected to be the trim speed. Substituting this set velocity into Equation 6.5 results in,

$$\Delta t = \frac{d_{ap}}{\overline{V}_T \cos(\gamma) - \overline{V}_p} \tag{6.7}$$

The $\cos(\gamma)$ term is added to obtain the horizontal component of the desired aircraft

velocity. Note that setting the aircraft velocity does not eliminate the touchdown point shifting however it does reduce it. The touchdown position can then be obtained by using either the aircraft's or moving platform's current position and velocity with the time value calculated. The moving platform is chosen to be used as its velocity is unlikely to change as it approaches the touchdown point.

Equation 6.6 defined the touchdown point with respect to the moving platform. However, it is desired to define the touchdown point in the inertial frame so that it can be used by other components in the GCS. The moving platform's position is already defined in the inertial frame. Therefore, its position can be added to the distance between the platform and the touchdown point (d_p) to obtain the touchdown point in the inertial frame. This is mathematically expressed as,

$$d_{td} = d_p + d_{p_0} \tag{6.8}$$

where d_{p_0} is the current inertial position of the platform when the prediction is made.

The prediction algorithm that has been described only operates in one dimension. However, the touchdown point required is defined in the inertial frame with three-position coordinates (N, E, D). The touchdown point's altitude (h_{td}) is set to the sum of the moving platform altitude (h_{mp}) and the virtual platform altitude (h_{vp}). The touchdown point's down coordinate D is the negative of the touchdown point's altitude ($D = -h_{td}$). The moving platform's altitude is generally constant and only changes slightly due to environmental factors such as dips in the runway or sensor noise. This means that the touchdown point's altitude, and hence the down coordinate (D), is not required to be predicted and can be updated with the DGPS measurements. The moving platform's horizontal coordinates change continuously as it moves and therefore these two coordinates need to be predicted for the touchdown point. Originally, the prediction algorithm ran in both the N and D directions of the inertial axis system which worked well as long as the platform moved in exactly a straight line. Unfortunately, the RC car used as the moving platform is very agile which means that its heading can change rapidly. This poses an unusual problem where the lateral position of the touchdown point with respect to the runway can vary wildly as is demonstrated in Figure 6.4.

Consider the RC car on the runway moving slightly to the right of the centreline (position A) as shown in Figure 6.4. If the car maintains this heading then the touchdown point is predicted to be on the right side of the centreline at location 1. This is not desired therefore the driver of the RC car will correct it by commanding a left steering angle to change its heading, putting the car in position B. The touchdown point now changes to the left side of the centreline (location 2). If the driver lets the car overshoot the centreline, then he would have to correct it again putting the touchdown point back to location 1. This oscillatory behaviour is not ideal as the UAV would not be able to change its heading fast enough resulting, in an inaccurate landing. The RC car can also not be guaranteed to

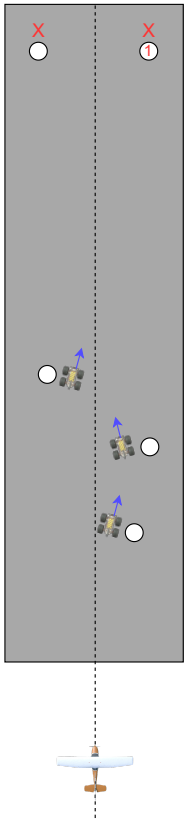


Figure 6.4: Diagram showing the predicted touchdown point changing based on the moving platform heading. [RC car model adapted from [68]]

follow the centreline perfectly due to loose joints on the car and the runway not being completely level.

A solution to this oscillatory behaviour is to only perform the touchdown point prediction in the direction of the runway centreline. This is possible as both the moving platform and UAV generally move in the direction of the runway centreline. Using this method will cause the touchdown point to only fall on the centreline. However, the offset of the moving platform's position from the centreline still needs to be dealt with. This issue is resolved with the cross-track controllers by giving them a reference that equals the offset of the moving platform from the centreline.

The disadvantage of using this method is that it forces the aircraft to land at a certain heading, which in this case is the runway heading. This is not a problem for this project, as the aircraft is required to land on the platform in the direction of the runway. For a scenario where the moving platform maintains a new heading during the landing, this method would not work and therefore either the original prediction in both the X and Y axes of the inertial reference frame needs to be used, or the runway heading angle needs to be changed to the new moving platform heading.

To implement this method, the UAV and moving platform positions and velocities must be defined with respect to the runway centreline. This can be achieved by using the runway frame whose X-axis points in the direction of the centreline of the runway, as

shown in Figure 6.5.

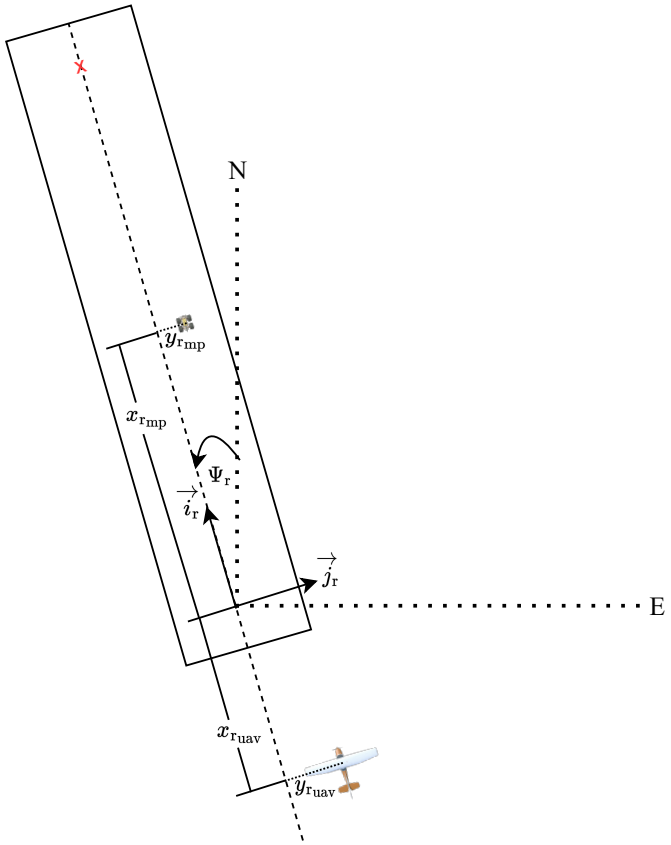


Figure 6.5: Diagram showing the runway frame.

The runway frame is related to the inertial frame by an angle Ψ_{r}, with their origins occurring at the same point. The runway heading Ψ_{r} is the angle between the runway and true north, and this angle is fixed as the runway does not change its position in real life. The runway heading was found to be,

$$\Psi_{\mathrm{r}} = -16.123^{\circ} \tag{6.9}$$

A rotation matrix with the Ψ_{r} angle can be used to transform the aircraft and moving platform positions and velocities from the inertial frame to the runway frame. This is mathematically expressed as,

$$\begin{aligned} \mathbf{V}_{\mathrm{r}} &= \mathbf{R}_{\Psi_{\mathrm{r}}} \mathbf{V}_{\mathrm{i}} \\ \begin{bmatrix} x_{\mathrm{r}} \\ y_{\mathrm{r}} \end{bmatrix} &= \begin{bmatrix} \cos\Psi_{\mathrm{r}} & \sin\Psi_{\mathrm{r}} \\ -\sin\Psi_{\mathrm{r}} & \cos\Psi_{\mathrm{r}} \end{bmatrix} \begin{bmatrix} x_{\mathrm{i}} \\ y_{\mathrm{i}} \end{bmatrix} \end{aligned} \tag{6.10}$$

where $\mathbf{V}_{\mathrm{r}}$ and $\mathbf{V}_{\mathrm{i}}$ are coordinate vectors in the runway and inertial frames respectively. Only the x component of the aircraft and moving platform positions and velocities in the runway frame is used with the prediction algorithm which was defined from Equations 6.6 to 6.8. The y component of the moving platform's position in the runway frame is

sent to the state machine. The state machine then sets this value as the reference for the cross-track controllers during the descent phase for landing.

The touchdown point is considered a waypoint and therefore it must be converted from the runway frame to the inertial frame, so that it can be used by the guidance algorithm. This is done by using the inverse of the rotation matrix $\mathbf{R}_{\Psi_r}$ which is its transpose. The conversion is mathematically represented as,

$$\begin{aligned} \mathbf{P}_{\mathrm{TD_i}} &= (\mathbf{R}_{\Psi_r})^{-1}\,\mathbf{P}_{\mathrm{TD_r}} \\ \mathbf{P}_{\mathrm{TD_i}} &= (\mathbf{R}_{\Psi_r})^{\intercal}\,\mathbf{P}_{\mathrm{TD_r}} \end{aligned} \tag{6.11}$$

where $\mathbf{P}_{\mathrm{TD_i}}$ and $\mathbf{P}_{\mathrm{TD_r}}$ are the touchdown points in the inertial and runway frames, respectively.

6.4. State Machine

The state machine provides the references for the FCS outer controllers to follow. These references are the airspeed ($\bar{v}_{ref}$), altitude (h_{ref}), cross-track error (y_{ref}) and crab angle ($\psi_{\mathrm{crab}_{ref}}$) references, which all change depending on the phase of flight. One of two distinct state machines are used to control the aircraft depending on the type of landing being performed. One state machine is used if the aircraft is doing a stationary runway landing, while the other is used for the moving platform landing. Before the state machines are introduced, the stabilisation of the aircraft for landing will first be discussed. Thereafter both state machines will be described. Finally, the section will be concluded by discussing the additional minor function performed by the state machine.

6.4.1 Aircraft Landing Stabilisation

It is important that the aircraft be stabilised for landing before it touches down, to ensure that the aircraft is in a safe position to accurately land and come to a complete stop without damaging the aircraft. The aircraft is stabilised if its variables are within predefined limits which are set depending on the landing performed. First the limits for the runway landing will be discussed and thereafter the moving platform landing limits are introduced.

6.4.1.1 Stationary Runway Landing Limits

The aircraft variables which are checked for stabilisation during a runway landing are the airspeed ($\overline{V}$), sink rate ($\dot{D}$), crab angle (ψ_{crab}), pitch angle (Θ), roll angle (Φ), cross-track error (y) and altitude error (h_{err}).

The aircraft is expected to land at an airspeed of 16 m/s for the runway landing. The airspeed can deviate slightly from this value due to the wind speed changing as the aircraft

lands. It is expected that the aircraft should be able to keep the airspeed within ±1 m/s from the airspeed reference, otherwise the aircraft may depart from the glide slope. The ideal sink rate of the aircraft on the glide slope is calculated using,

$$\dot{D} = V_{\text{ground}} \sin \gamma \tag{6.12}$$

where $\dot{D}$ is the sink rate, V_{ground} is the ground speed, and γ is the glide slope angle. Assuming that there is no wind, and therefore the ground speed equals the ideal airspeed of 16 m/s, the ideal sink rate is calculated to be 1.12 m/s. The wind has a significant effect on the aircraft's sink rate, because if it is assumed that the aircraft's airspeed is sufficiently controlled, then the wind speed directly influences the ground speed and hence the sink rate. The maximum acceptable wind speed in which the aircraft is expected to fly is 3.1 m/s. If this is a tail wind, then the sink rate will be at its maximum. The ground speed of the aircraft at this wind speed will be 19.1 m/s therefore the maximum allowable sink rate is 1.33 m/s.

For a standard runway landing, a flare manoeuvre is performed to reduce the sink rate just before touchdown, to minimise the touchdown force acting on the aircraft which could cause damage. It was decided not to perform the flare manoeuvre for the landing as it would cause the aircraft to overshoot its intended touchdown point, decreasing the landing accuracy. The maximum sink rate therefore had to be tested to ensure that damage is not caused to the aircraft. This was done by performing a drop test from a certain height, so that the aircraft reaches the maximum sink rate just before impact. The height for the drop test was calculated using the following equation used by Le Roux [14],

$$h_0 = \frac{\bar{v}^2}{2g} \tag{6.13}$$

where h_0 is the height from which the aircraft is dropped and $\bar{v}$ is the speed of the aircraft just before impact. Using the maximum sink rate of 1.33 m/s, the aircraft should be dropped from a height of 0.0906 m or 9.06 cm. The aircraft was dropped from this height and the landing gear was able to absorb the shock without damaging the aircraft.

The maximum crab angle that the aircraft is allowed to have for a landing was set to 10°, as a higher angle would be too difficult for the rudder reduce to zero, and it would struggle to de-crab the aircraft. The maximum pitch and roll angle limits were determined by De Bruin [16] using the airframe geometry. Since the same airframe is used for this research project, the limits would be the same. The maximum allowable pitch and roll angle for the airframe are 9° and 23°, respectively. To ensure that the aircraft does not experience a wingstrike or tailstrike, more stringent pitch and roll angle limits were selected with their values being set to 6° and 8° respectively. The aircraft is expected to land within a 3 m by 3 m square and therefore the maximum allowable cross-track error is 1.5

m for the aircraft to have an accurate landing. The in-track error also has a maximum limit of 1.5 m however, its accuracy depends on the aircraft's ability to track the glide slope. This means that the altitude tracking error must be limited to a maximum value of,

$$h_{err} = 1.5 \tan \gamma = 0.105 \text{ m} \approx 0.1 \text{ m} \tag{6.14}$$

A summary of all the limits that the aircraft must satisfy for a stationary runway landing is shown in Table 6.1.

Table 6.1: Stationary Runway Landing Limits

Aircraft Variable	Landing Limit
Airspeed	$15 \text{ m/s} < \overline{V} < 17 \text{ m/s}$
Sink Rate	$\dot{D} < 1.33 \text{ m/s}$
Crab Angle	$\lvert\psi_{\text{crab}}\rvert < 10^\circ$
Pitch Angle	$\Theta < 6^\circ$
Roll Angle	$\lvert\Phi\rvert < 8^\circ$
Cross-track Error	$\lvert y\rvert < 1.5 \text{ m}$
Altitude Error	$\lvert h_{err}\rvert < 0.1 \text{ m}$

6.4.1.2 Moving Platform Landing Limits

The aircraft variables checked for stabilisation during a moving platform landing are the same as those for the runway landing. However, the cross-track error of the aircraft is measured with respect to the moving platform instead of the runway centreline ($y - y_{r_{mp}}$). An additional variable is also checked for stabilisation, namely the moving platform's cross-track error from the centreline ($y_{r_{mp}}$).

The aircraft's airspeed for a moving platform landing is 18 m/s and, similar to the runway landing, it is expected that the aircraft keeps the airspeed within ± 1 m/s of the airspeed reference. Using equation 6.12 with the aircraft's airspeed of 18 m/s, the ideal sink rate is calculated to be 1.26 m/s. The 3.1 m/s tail wind is also considered when obtaining the maximum sink rate of the aircraft which is calculated to be 1.47 m/s. As the aircraft tracks an agile moving platform, the sink rate limit was increased to 1.80 m/s which is still safe as the aircraft does not actually impact the platform. This also means the drop test would not need to be performed.

The crab and pitch angle limits were kept the same as for the runway landing. The roll angle limit was increased as the aircraft had to track the moving platform in the lateral direction as well, requiring the aircraft to bank. The roll angle limit was set to 15° which is still less then the maximum allowable limit previously mentioned. The aircraft will not touchdown in any case, so the possibility of a wingstrike is minute.

The cross-track error limit of the aircraft with respect to the moving platform was set to 1.5 m as this is the maximum acceptable cross-track error on touchdown. The cross-track error limit of the moving platform with respect to the centreline was set to 3 m to ensure that the aircraft does not follow the platform in case it veers off the runway. The altitude tracking error limit was increased to 0.3 m which is required because the touchdown point and hence the glide slope changes continuously. The touchdown point stabilises as the aircraft approaches it therefore the altitude error should reduce. Table 6.2 shows a summary of all the limits used to determine if the aircraft is sufficiently stabilised for a moving platform landing.

Table 6.2: Moving Platform Landing Limits

Aircraft Variable	Landing Limit
Airspeed	$17 \text{ m/s} < \overline{V} < 19 \text{ m/s}$
Sink Rate	$\dot{D} < 1.8 \text{ m/s}$
Crab Angle	$\lvert\psi_{\text{crab}}\rvert < 10°$
Pitch Angle	$\Theta < 6°$
Roll Angle	$\lvert\Phi\rvert < 15°$
Aircraft Cross-track Error from Moving Platform	$\lvert y - y_{r_{mp}}\rvert < 1.5 \text{ m}$
Moving Platform Cross-track Error	$\lvert y_{r_{mp}}\rvert < 3 \text{ m}$
Altitude Error	$\lvert h_{err}\rvert < 0.3 \text{ m}$

6.4.2 Stationary Runway Landing State Machine

The stationary runway state machine contains 6 distinct states (0-5) that must be followed to perform a runway landing, as shown in Figures 6.6 and 6.7. When the state machine is first activated, by either entering land mode or after the moving platform landing has been completed, it starts in the Waypoint Navigation state (state 0) and completes the circuit around the airfield. When the aircraft is on final approach, the waypoint scheduler alerts the state machine and this allows the state machine to transition to the Final Approach state (state 1). In this state the aircraft is commanded to slow down to its landing speed of 16 m/s.

When the aircraft reaches a distance d_g from the touchdown point (shown in Figure 6.7), and the altitude error h_{err} is less than a meter, the state machine will transition to the Glideslope Tracking state (state 2). The state machine causes the aircraft to track the glide slope by providing the appropriate altitude reference to the FCS. When the aircraft is a distance d_t from the touchdown point, the state machine checks if the aircraft is stabilised by determining if the aircraft variables from Table 6.1 are within their limits.

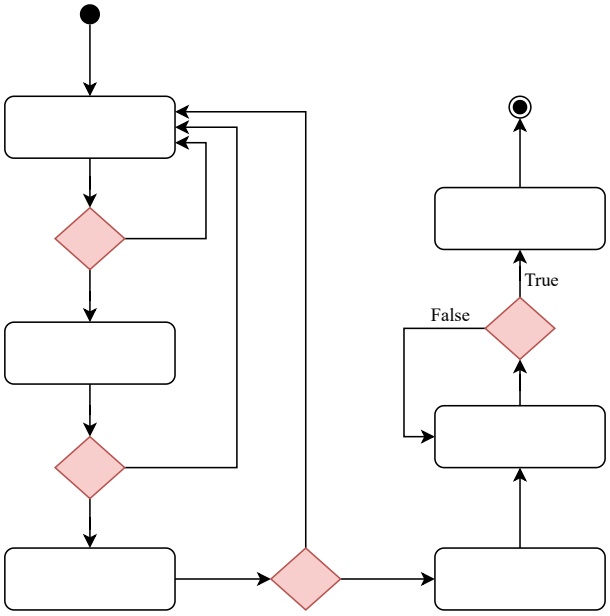


Figure 6.6: Diagram showing the states for the stationary runway landing. The decisions made by the state machine are shown in Table 6.3.

Table 6.3: Stationary runway landing state machine decisions

Decision 1 (D1)
Final approach waypoint reached ? (waypoint END == true ?)
Decision 2 (D2)
Altitude tracking error is within acceptable limits ? ($\lvert h_{err} \rvert < 1$ m ?)
Decision 3 (D3)
Aircraft state variables are within the acceptable envelope for a runway landing (Defined in Table 6.1) ?
Decision 4 (D4)
Touchdown acceleration spike detected ? ($\ddot{D} < -20$ m/s^2 ?)
Aircraft altitude is below 0.5 meters ? ($h < 0.5$ m ?)

If the aircraft is not sufficiently stabilised, then the state machine will abort the landing attempt and cause the aircraft to return to the Waypoint Navigation state so it can go around and try the landing again. If the aircraft is sufficiently stabilised, then the state machine transitions to the Stabilised state (state 3). From this state onward the aircraft can no longer abort and has to commit to the landing unless land mode is disengaged or the human pilot takes over. This was chosen because it is not desirable for the aircraft to perform any drastic manoeuvres when close to the ground, as it could be catastrophic and lead to a crash. If it is unsafe to land when close to the ground, then the human pilot will take over and manually go around.

When the aircraft is a distance d_d from the touchdown point, then it starts to perform the de-crab manoeuvre. This manoeuvre is executed when the state machine, in the Decrab state (state 4), activates the crab angle controller and sends it a 0° angle reference.

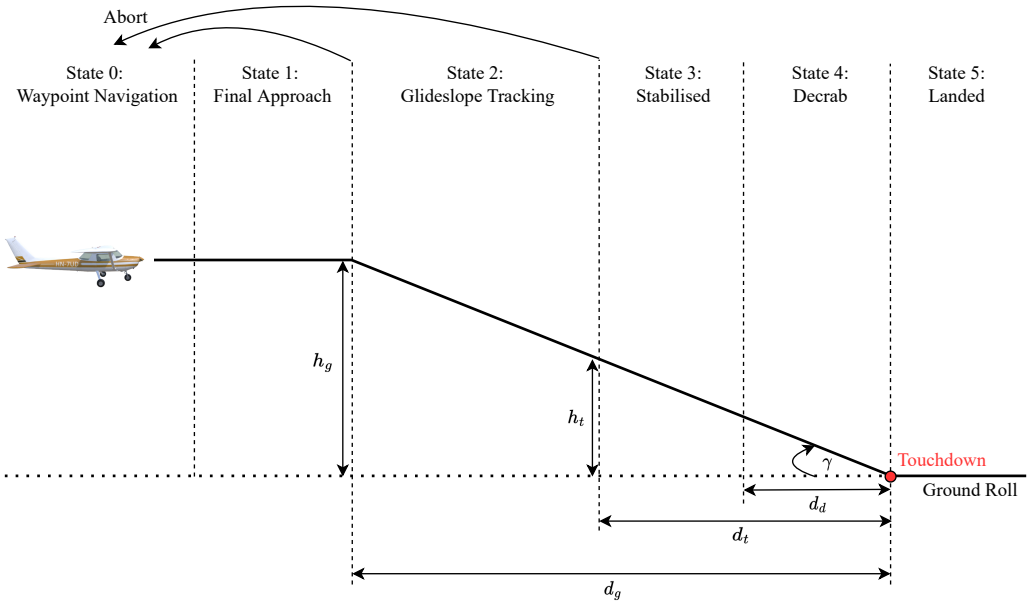


Figure 6.7: Diagram showing the state transitions for the stationary runway landing profile. The values for the symbols in the diagram are shown in Table 6.4.

Table 6.4: Stationary runway landing profile values

Symbol	Value
d_g	250 m
h_g	17.48 m
γ	4°
h_t	5 m
d_t	71.5 m
d_d	Depends on V_{ground}

The distance d_d is dynamically calculated based on the aircraft ground speed (V_{ground}) and the crab angle controller's rise time (τ_{crab}), and it is calculated using,

$$d_d = V_{\text{ground}} \tau_{\text{crab}} \tag{6.15}$$

The aircraft continues along the glide slope until it touches down on the runway. At touchdown, the aircraft experiences a large acceleration spike which can be measured to determine if the aircraft has landed. If the acceleration spike is greater than -20 m/s^2 and the aircraft's altitude is less than 0.5 m (to prevent early activation) then the state machine transitions to the Landed state (state 5). In this state the thrust command and control surface deflections are set to zero, and it is expected that the human safety pilot will immediately takeover and bring the aircraft to a standstill.

6.4.3 Moving Platform Landing State Machine

The moving platform state machine is similar to runway landing state machine with the main difference being that it has an additional state. The moving platform state machine therefore has 7 distinct states (0-6) to perform the moving platform landing, as shown in Figures 6.8 and 6.9. The moving platform state machine is activated when the aircraft is placed into moving platform mode. It starts in the Waypoint Navigation state (state 0) similar to the runway state machine. The moving platform state machine will then transition to the Final Approach state (state 1) when the aircraft is aligned with the runway and the waypoint scheduler alerts the state machine. Unlike the runway state machine, the moving platform state machine does not slow the aircraft down but instead keeps it at the trim airspeed of 18 m/s.

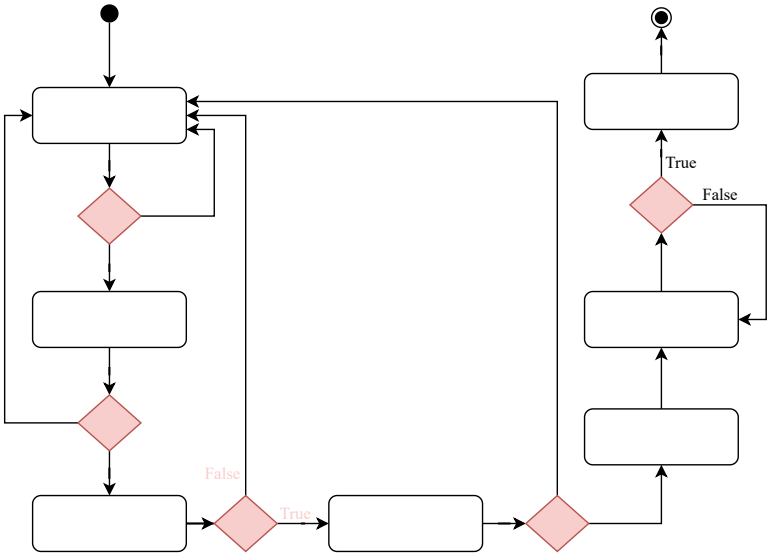


Figure 6.8: Diagram showing the states for the moving platform landing. The decisions made by the state machine are shown in Table 6.5.

When the aircraft is a distance d_g from the touchdown point (shown in Figure 6.9) and has an altitude error of less than one meter, the state machine transitions to the Glideslope tracking state (state 2) so that the aircraft can catch the glide slope.

When the aircraft is a distance d_{ct} from the touchdown point and the moving platform's cross-track error ($y_{r_{mp}}$) is within 3 m of the runway centreline, the state machine transitions to the Moving Platform Tracking state (state 3). In this state the state machine provides the cross-track controllers a reference that equals the moving platform's cross-track position from the centreline ($y_{r_{mp}}$). This is so that the aircraft can track the moving platform's lateral position. The distance d_{ct} is dynamically calculated using the aircraft's ground speed V_{ground} and both the first cross-track and crab angle controller rise times (τ_{ct} and τ_{crab}).

$$d_{ct} = V_{\text{ground}} \left(\tau_{ct} + \tau_{\text{crab}} \right) \tag{6.16}$$

Table 6.5: Moving Platform landing state machine decisions

Decision 1 (D1) Final approach waypoint reached ? (waypoint END == true ?)
Decision 2 (D2) Altitude tracking error is within acceptable limits ? ($\|h_{err}\| < 1$ m ?)
Decision 3 (D3) Moving platform's cross-track error is within acceptable limits ? ($\|y_{r_{mp}}\| < 3$ m ?)
Decision 4 (D4) Aircraft state variables are within the acceptable envelope for a moving platform landing (Defined in Table 6.2) ?
Decision 5 (D5) Aircraft altitude has reached virtual platform altitude ? ($h < h_{vp}$ m ?)

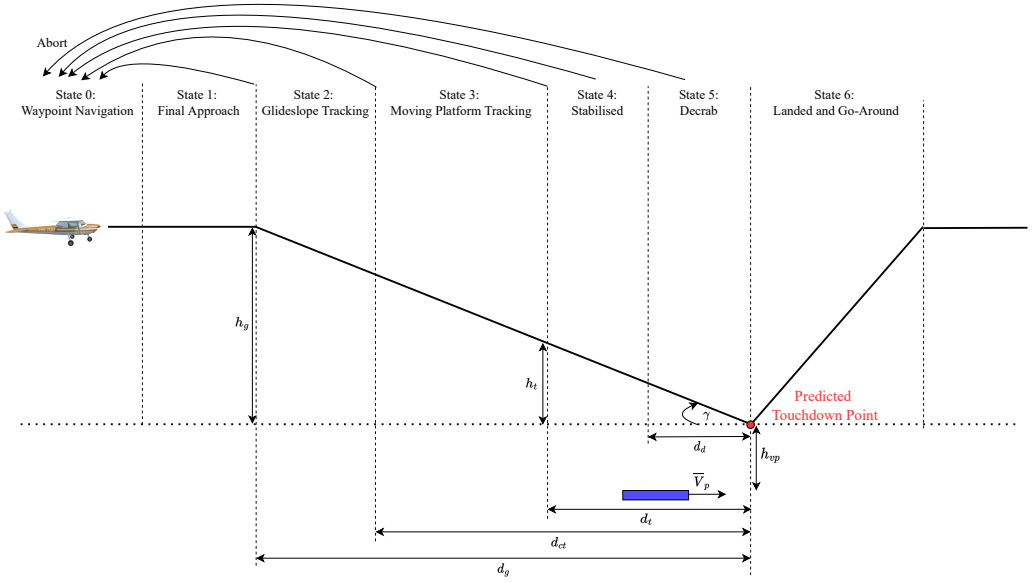


Figure 6.9: Diagram showing the state transitions for the moving platform landing profile. The values for the remaining symbols in the diagram are shown in Table 6.6.

Table 6.6: Moving platform landing profile values

Symbol	Value
d_{ct}	Depends on V_{ground}
h_{vp}	3 m
$\overline{V}_p$	3 m/s

The reason the crab angle controller rise time is included when calculating d_{ct} is so that the aircraft can be aligned with the moving platform before the aircraft de-crabs. De-crabbing causes the aircraft to increase its cross-track error which makes alignment with the platform difficult, which is the reason why these two manoeuvres are performed separately. When the aircraft is a distance d_t from the touchdown point, the state machine checks if the

aircraft is stabilised by determining if the aircraft variables from Table 6.2 are within their limits. If the aircraft is not sufficiently stabilised then, similar to the runway landing, it will abort the landing attempt and try again. If the aircraft is sufficiently stabilised then the state machine transitions to the Stabilised state (state 4). The stability check for the moving platform landing is continuously done from this state onward until touchdown, as there is no physical impact with the moving platform and the aircraft does the go-around after landing anyways. This continuous check will protect the aircraft if the moving platform veers off the runway as the aircraft will then abort the landing attempt. As with the runway landing, the aircraft de-crabs when it reaches a distance d_d from the touchdown point and the state machine transitions to the Decrab state (state 5). The aircraft is considered to have touched down when its altitude is less than the virtual platform's altitude. The moving platform state machine will then transition to the Landed and Go-Around state (state 6) and after that, it activates the stationary runway landing state machine. The aircraft is then programmed to go around and land on the runway. However, it is more likely that the human safety pilot would take over and manually land on the runway.

6.4.4 Additional Function of State Machine

The additional function of both state machines is to produce the pre-injected climb rate reference ($\dot{\bar{h}}_{ref}$) used by the classical altitude controller. This reference is calculated by using the aircraft ground speed V_{ground} and glide slope angle γ with the equation,

$$\dot{\bar{h}}_{ref} = -V_{\text{ground}} \tan\gamma \tag{6.17}$$

The state machines provide the classical altitude controller with this reference when the aircraft starts tracking the glide slope (from state 2 onward).

6.5. Summary

This chapter presented the guidance control system used to navigate the aircraft around the airfield. First, the guidance algorithm used to calculate the values needed by the FCS cross-track controllers was described. Then the waypoint scheduler used to select the current waypoints being tracked was introduced. This was followed by the description of the landing position predictor, which was used to predict the touchdown point between the aircraft and the moving platform. Finally, the state machines used during the runway and moving platform landing were discussed. Now that both the flight and guidance control systems have been developed, they can be tested in non-linear simulations to verify their performance in a more realistic environment before they are implemented on the physical system. The non-linear simulations are presented in the next chapter.

Chapter 7

Non-Linear Simulation

This chapter presents the non-linear simulation implementation and results. The non-linear simulations are performed to test the developed control systems on a non-linear model that more closely resembles the physical UAV. These simulations will identify any discrepancies in the control system design, which allows them to be corrected before being implemented on the physical UAV. This is a crucial process as the non-linear simulations are a low-risk environment compared to the practical tests where there is only one physical UAV. The non-linear simulations are first performed in Simulink and then in the PX4 Autopilot software in software-in-the-loop (SITL) mode. The Simulink non-linear model is first discussed, with the additions made to control systems for the non-linear simulations being mentioned. The software-in-the-loop implementation is then described, with the different components used in the implementation being highlighted. Finally, the non-linear simulation results for control systems are analysed.

7.1. Simulink Non-Linear Model

The controllers were designed using a linear model. However, their performance must be verified using a non-linear model, which better represents the non-linearities of the aircraft. This will also ensure that all the linearisation decisions made are validated. The controllers are first tested on a non-linear model in Simulink so that the controllers performance can be verified fairly quickly. This is important as the controllers need to be tuned to obtain ideal performance on a non-linear model while still staying within their requirements. Simulink is ideal for this as it is easy and time-efficient to test the controllers. The Simulink model does not include sensor noise or wind therefore, the results obtained are the best case for the controllers, which serves as a good baseline for the software-in-the-loop (SITL) results.

Most of the controller references are bounded to ensure that the aircraft behaves in a predictable manner and that a commanded manoeuvre will not cause the aircraft to go unstable. These reference bounds which applied are listed below:

- The NSA reference $C_{w_{ref}}$ for level fight is $-g$ m/s^2 where g is the gravitational acceleration. This due to the accelerometer experiencing a normal force when the

aircraft is flying level. The NSA reference is limited to $\pm g$ m/s^2 from this level flight reference value.

- The climb rate reference is limited to ± 2 m/s to prevent strain on the aircraft.
- The roll angle reference is limited to $\pm \frac{\pi}{6}$ rad ($\pm 30°$) to prevent the aircraft from stalling on turns.
- The LSA reference is limited to $\pm g$ m/s^2.

The control surface deflection angles are also limited to ± 1 rad ($\pm 57.3°$), as this is the maximum angle that the SITL mixer can provide. This limit is applied to the Simulink simulation as well to make its results consistent with the SITL results. The thrust command is bounded between 0 N and 40 N as this is the thrust range of the physical motor. All the controller integrators contain anti-windup to prevent them from winding up when the controller output saturates.

7.2. Software in the Loop Implementation

The Software in the Loop (SITL) simulation is performed to test the controllers in a more realistic environment and, if successful, it will provide confidence in the controller's ability to perform on the practical vehicle. The SITL implementation includes sensor noise and wind which improve the realism of the simulation. The SITL simulation is implemented by using the PX4 autopilot software to execute the flight and guidance control systems, Gazebo to simulate the aircraft physics, and ROS to run the MPC algorithm, as shown in Figure 7.1. For SITL, the PX4 autopilot software, ROS node, and Gazebo simulator are all run on the same computer which uses a Linux operating system.

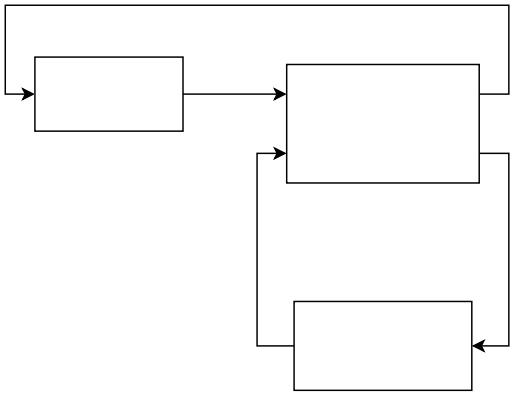


Figure 7.1: Diagram showing an overview of the SITL implementation.

The classical controllers of the flight control system are implemented in the PX4 software. The PX4 software therefore outputs the control surface deflections and thrust command produced by the lower level controllers, which are sent to the Gazebo simulator

using MAVLink. Gazebo applies these commands on the aircraft model and then outputs the simulated sensor values derived from the aircraft physics. The PX4 software also sends the MPC, which is located in the ROS node, its airspeed and altitude references together with the aircraft airspeed and altitude states. The MPC in turn provides the PX4 software with the climb rate reference and thrust command from trim.

7.2.1 Robot Operating System

The Robot Operating System (ROS) is a set of software libraries that uses ROS nodes to perform actions and uses ROS topics to transmit data between these nodes [69]. The MPC algorithm is implemented in a ROS node and it uses MAVROS with MAVLink to communicate with PX4. MAVLink is a messaging protocol used to allow communication between isolated hardware or software components on UAVs [50]. In the SITL implementation, MAVLink connects both the ROS node and Gazebo to the PX4 software. MAVROS is a communication driver that converts MAVLink messages to ROS messages, and vice versa [51].

The details of the MPC implementation on a ROS node in simulation and on practical hardware can be found in Appendix B.5.2.

7.2.2 PX4 Autopilot Software

PX4's architecture is similar to ROS where it uses modules to execute certain functions and these modules communicate with each other using topics. The functions they perform include flight control, state estimation, and communication. A simplified overview of the PX4 architecture used for the SITL implementation is shown in Figure 7.2.

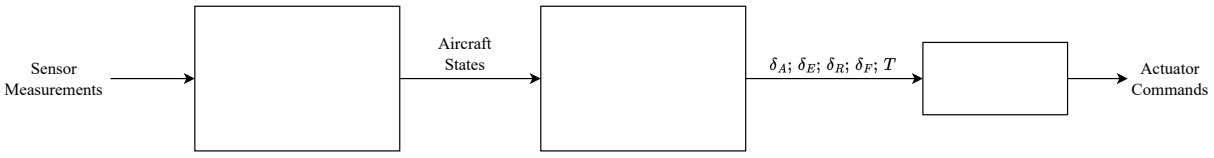


Figure 7.2: Diagram showing a simplified overview of the PX4 architecture.

The sensors measurements provided to the PX4 software are updated at different rates and are often too slow to be used by the controllers. This issue is resolved by using a state estimator which estimates the states of the aircraft at a higher rate and uses the sensor measurements for corrections. PX4 already contains an estimator which is an Extended Kalman Filter (EKF) that has been successfully used in many UAV projects. This EKF is therefore chosen to be the state estimator for this research project. The EKF operates in delayed time and it uses a complementary filter to propagate the state estimates to real time [70].

The flight and guidance control systems from Chapters 5 and 6 were implemented from scratch in a custom module using the PX4 software module template. This module is

programmed in C++ and its notation are derived from other PX4 software modules. This custom module can be built for both the SITL simulation and for the Pixhawk 4 hardware. The only differences between the two builds are minor changes used to accommodate the practical UAV. The flight and guidance control systems run every 20 ms in the PX4 software, as this is the update time of the aircraft states from the EKF.

The mixer is used to convert the control surface deflections and thrust command to actuator commands [71]. This is a crucial conversion for the practical UAV as the servos for the control surfaces and the electronic speed controller (ESC) for the motor require a PWM signal with a duty cycle between a certain range. The practical mixer is discussed in more detail in Appendix B.3. The SITL mixer does not have a conversion (scaling and offset) for the control surface deflections as there is no physical component to accommodate for. The mixer only accepts a normalised input between -1 to 1 and since the deflection angles are usually less than ± 1 rad ($\pm 57.3°$), they are not normalised. This means that the deflection angles commanded by the controllers are sent unchanged from the PX4 software to the Gazebo simulator. The deflection angles are therefore limited to the normalised range, making their limits ± 1 rad ($\pm 57.3°$). This deflection angle range is much larger than the commands provided by the lower-level controllers, and therefore the controllers' performance should not be impacted. The SITL mixer performs a conversion on the thrust command as the thrust has a range from 0 N to 40 N. The thrust command is normalised between 0 and 1 to send it through the mixer. The original thrust command is recovered in the Gazebo simulator to apply it to the aircraft.

7.2.3 Gazebo Simulator

Gazebo simulates the fixed-wing UAV and the moving platform as models in a three-dimensional world. Figure 7.3 shows the fixed-wing UAV in the Gazebo world.

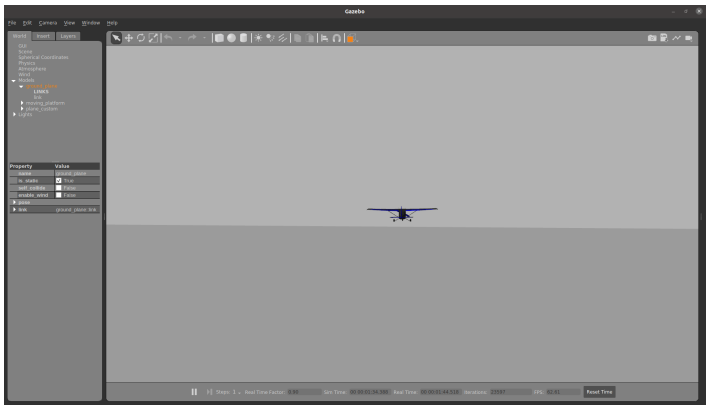


Figure 7.3: Simulated fixed-wing UAV flying in Gazebo simulation.

Gazebo also uses a similar structure to ROS in which it uses plugins to perform actions on the models while using topics to transmit data between the plugins. Figure 7.4 shows a simplified overview of the fixed-wing UAV simulation in Gazebo.

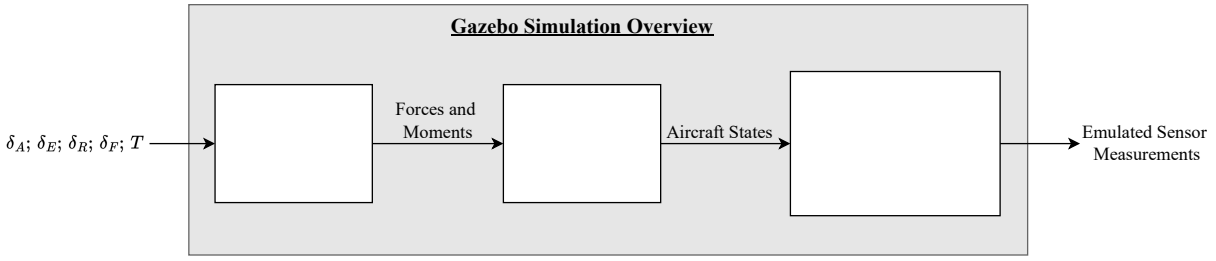


Figure 7.4: Diagram showing a simplified overview of the Gazebo simulation.

A model for the fixed-wing UAV is already available by default. However, the model's aerodynamic and thrust plugins cannot be used as they do not model the UAV as a rigid body. Custom aerodynamic and thrust plugins were therefore created which implement the force and moment models from sections 4.3.1 and 4.3.2, respectively. The aerodynamic plugin includes the different types of wind modelled in section 4.4. The turbulence filters from Table 4.2 are discretised before they are added. The control surface deflections output by the PX4 software are used in the aerodynamic model equations to calculate the aerodynamic forces and moments acting on the aircraft. The thrust command is first converted back to its unnormalised value before it is used in the thrust model equation to calculate the thrust force acting on the aircraft.

Gazebo produces emulated sensor measurements based on the fixed-wing UAV model states, so that they can be used by the PX4 EKF. The emulated sensor measurements are produced by individual plugins which represent the sensors. The sensors that are simulated are the GPS, barometer, magnetometer, IMU and airspeed sensor.

The moving platform model and plugin were created from scratch. The physics of the platform is not that important, as it will be represented by an RC car that is manually controlled. Only the position and velocity of the platform are of interest, therefore the plugin uses the standard Gazebo functions to move the platform at a constant velocity in a straight line.

7.3. Non-Linear Simulation Results

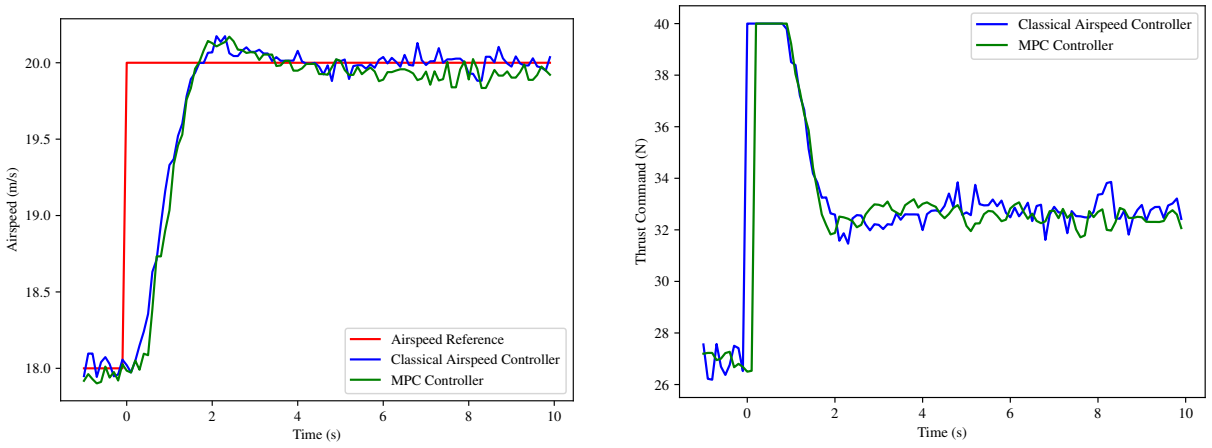
The non-linear simulation environment has now been introduced. Therefore, this section focuses on analysing the results of the non-linear simulation. First, the step responses of the individual controllers are simulated and analysed. Next, the waypoint navigation, stationary runway landing, and moving platform landing are simulated and discussed. Finally, the runway and moving platform landing simulations with wind are performed to show the distribution and evaluate the landing accuracy of the touchdown points when affected by wind. Only the SITL results will be analysed, as the Simulink results are very similar to SITL.

7.3.1 Controller Step Responses

The controllers in the FCS were designed using a linear model. However, now their performance must be tested on the non-linear simulation model to verify that the design decisions made were sound. The inner-loop controllers (NSADLC, LSA and roll rate) are not tested on the non-linear aircraft model as they cause the aircraft to depart from trim flight. These controllers will be indirectly tested through the outer-loop controllers. The heading and second cross-track controllers are also not tested as they are not used for the landing procedure but only to bring the aircraft close to the ground track. No wind effect is active when testing the individual controllers in the SITL simulation. The step response magnitude for the controllers are set to values that have a significant impact on the aircraft, while also not causing the controllers to saturate.

7.3.1.1 Airspeed Controller

The airspeed controllers are tested with the aircraft being in level flight holding its current altitude. Figure 7.5 shows the airspeed step response and the corresponding thrust command for both the classical airspeed and MPC controllers. Note that the airspeed step is applied with respect to the trim airspeed of the aircraft and the controller thrust is added to the trim thrust, which is exerted on the aircraft.



(a) Airspeed Step Response **(b)** Thrust Command

Figure 7.5: Airspeed step response and corresponding thrust command for the classical airspeed and MPC controllers.

The airspeed state in SITL is only updated every 100 ms as the state is the filtered output of the airspeed sensor measurements. The EKF is not involved in obtaining the airspeed state; therefore, the update rate is slow. The airspeed response contains visible high-frequency variations caused by sensor noise. The sensor noise is present in the airspeed state provided to the airspeed controllers. Nonetheless, both the classical airspeed and MPC controllers can adequately control the airspeed even with the slow update rate and sensor noise.

The simulated closed-loop step response of the classical airspeed controller exhibits a rise time of 1.42 seconds, an overshoot of 9%, and zero steady-state error. These metrics are within the requirements for the controller specified in the classical airspeed controller design section. The requirements specified for the controller are a rise time of less than 3 seconds, an overshoot of less than 20% and zeros steady-state error. The nonlinear airspeed response is more representative, and less ideal than the linear airspeed response, which can be attributed to the simplified model (Equation 5.2) used during the design process. The model decoupled the airspeed state from the other longitudinal states under the assumption that the airspeed is not affected by the other states. This is not entirely true as the airspeed is affected by the NSA and hence the NSADLC controller.

The simulated closed-loop airspeed step response for the MPC controller has a rise time of 1.46 seconds, an overshoot of 8.5%, and a 0.075 m/s nominal steady-state error. The rise time and overshoot are within the MPC controller requirements however, the steady-state error is not, as it should have been zero. The MPC controller requirements are the same as the classical airspeed controller requirements that were just mentioned. The steady-state error in airspeed is due to the MPC trying to maintain the current altitude reference by sacrificing some airspeed control. As the aircraft airspeed increases, the lift force exerted on the aircraft also increases, which affects the altitude, and this needs to be dealt with by the MPC. The MPC linear airspeed response has better performance then the non-linear response which is due to the MPC being very model-dependent. Applying the MPC to a different model from the one it is designed on degrades its performance, which is expected. The K_{T_c} thrust scalar was introduced to compensate for inaccuracies in the MPC thrust model when compared to the practical data. By introducing this scalar, the practical performance improved at the expense of worse simulation performance. The MPC was first designed without the scalar and it had less steady-state error however, its practical performance was lacking, therefore the scalar was introduced.

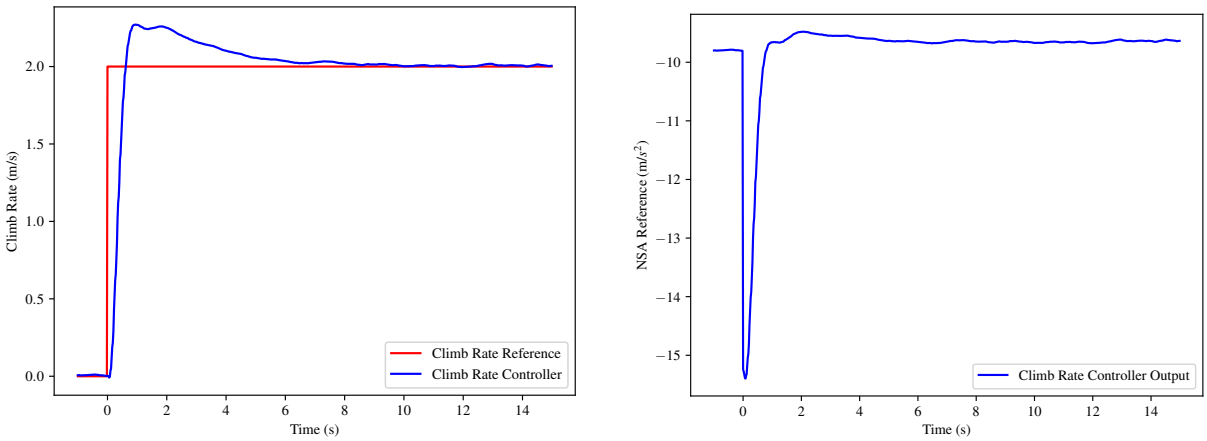
Comparing the classical airspeed and MPC step responses it can be seen that they are similar to a point after which the MPC's response exhibits the steady-state error. Therefore the classical airspeed controller performs better than the MPC controller in SITL, which contrasts the linear step response results. This is due to the MPC's model dependence which degrades its SITL performance more than the classical airspeed controller. It is very difficult for the controllers to keep the airspeed exactly at the reference due to the airspeed depending on wind which randomly changes. It is more important for the controllers to keep the airspeed above the stall speed and to ensure that the airspeed does not effect the other controllers. This is especially true for the MPC as its altitude control needs to be highly accurate to maintain the glide slope on landing, even at the expense of airspeed control performance. Nonetheless, both the controllers are capable of reasonably controlling the airspeed.

The associated thrust command for both controllers, shown in Figure 7.5b, are within

the physical limits of the motor, therefore these controllers should be physically realisable.

7.3.1.2 Climb Rate Controller

The climb rate controller is tested with the aircraft flying at a constant heading and airspeed. Figure 7.6 shows the simulated climb rate step response and the associated NSA reference commanded produced by the climb rate controller.



(a) Climb Rate Step Response **(b)** NSA Command

Figure 7.6: Climb rate step response and corresponding NSA command for the climb rate controller.

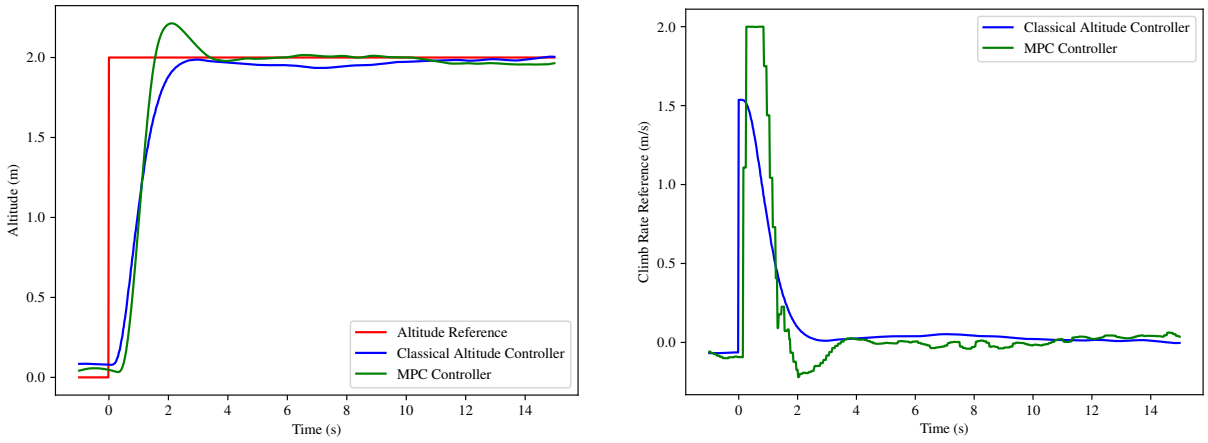
The non-linear climb rate response has less sensor noise than the non-linear airspeed response, which is due to the climb rate state being provided by the EKF that reduces noise. The climb rate is the negative of the down inertial velocity component, which the EKF generates by using the IMU with corrections from the GPS measurements. The non-linear climb rate step response has a rise time of 0.54 seconds, a 13.65% overshoot, and zero steady-state error, which are within the requirements for the climb rate controller. The climb rate controller requirements are a rise of less than 3 seconds, an overshoot of less than 20% and zero-steady state error. The step response also has a 2% setting time of 5.85 seconds. The rise time and 2% settling time of the non-linear climb rate step response are faster then the linear response at the expense of a slightly higher overshoot. These non-linear performance metrics are acceptable. However, they are expected to influence the outer-loop altitude controller.

The associated commanded NSA reference in Figure 7.6b is within the $\pm g$ limit from the level flight setpoint ($-g$ m/s^2), making the controller realisable.

7.3.1.3 Altitude Controller

The altitude controllers are tested with the aircraft flying at a constant heading and airspeed. The altitude steps are applied from a captured altitude point which is set when the autopilot is engaged. This altitude point is aimed to be set close 50 m above the

runway. Figure 7.7a shows the the altitude controllers' non-linear step response, with the aircraft altitude being measured with respect to the captured altitude point. Figure 7.7b shows the associated climb rate reference commanded by the altitude controllers to produce the step response. The limited integrators for the classical altitude and MPC controllers are not active for these step responses to have a fair comparison between the non-linear and linear responses.



(a) Altitude Step Response **(b)** Climb Rate Command

Figure 7.7: Altitude step response and corresponding climb rate command for the classical altitude and MPC controllers.

The classical controller altitude step response has a rise time of 1.82 seconds, no overshoot, and zero steady-state error which are within the requirements for the classical controller. The classical altitude controller requirements are a rise time of less than 6 seconds, less than 20% overshoot, zero steady-state error, and a 2% settling time of less than 13 seconds. The classical altitude step response has an undershoot, which drastically increases its 2% settling time to 9.17 seconds when compared to the linear step response. This settling time, however, is still within the controller requirement and is therefore acceptable. The MPC step response has a rise time of 1.40 seconds, a 10.7% overshoot, and zero steady-state error which are all within the requirements for the MPC. The MPC altitude requirements are the same as the classical altitude controller requirements mentioned above. The 2% settling time for the MPC altitude response is 3.13 seconds, which is slower than the linear MPC altitude response, due to the MPC being model-dependant. However, it is still acceptable, as it is within the requirement for the MPC. The MPC does slightly deviate from the reference towards the end of the response, however, it is still within the 2% limit, and it does return to the reference.

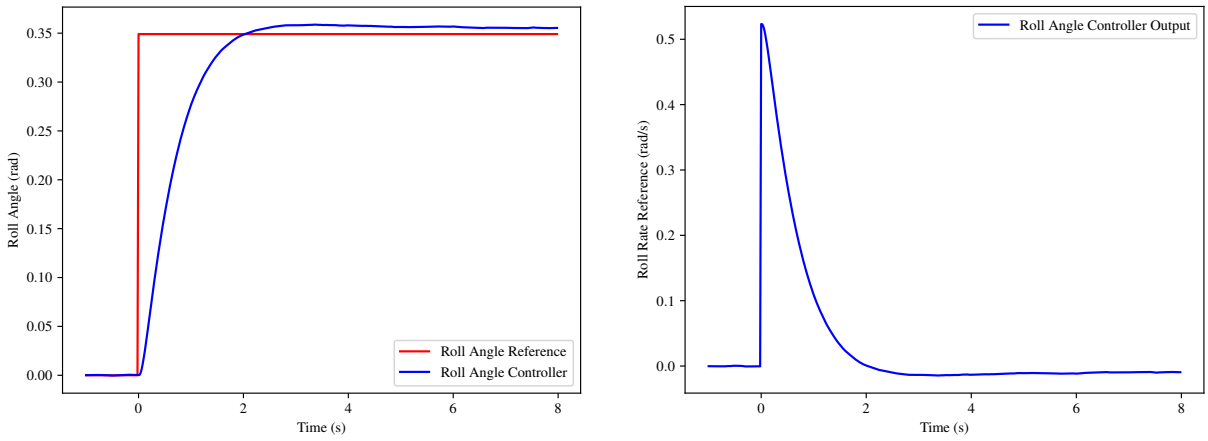
Comparing the classical altitude controller and MPC responses, it can be seen that the MPC performs better, as it has a faster rise and settling time at the expense of higher overshoot. This faster rise and settling time is due to the MPC being able to use the airspeed to assist its altitude response and to also command the maximum climb rate reference (shown in Figure 7.7b) to increase the altitude as fast as possible. The MPC

overshoot is an acceptable trade-off in obtaining the fast settling time required by the aircraft to quickly track the glide slope and achieve an accurate landing.

The climb rate reference produced by both controllers in Figure 7.7b are within the limits for the reference. The MPC is able to utilise the climb rate reference to its limit as the limit is specified when the MPC is designed.

7.3.1.4 Roll Angle Controller

The roll angle controller is tested with the aircraft flying at a constant altitude and airspeed. Figure 7.8 shows the roll angle controller step response and the associated roll rate reference commanded by the controller. The roll angle step magnitude is set to 20° (0.349 rad).



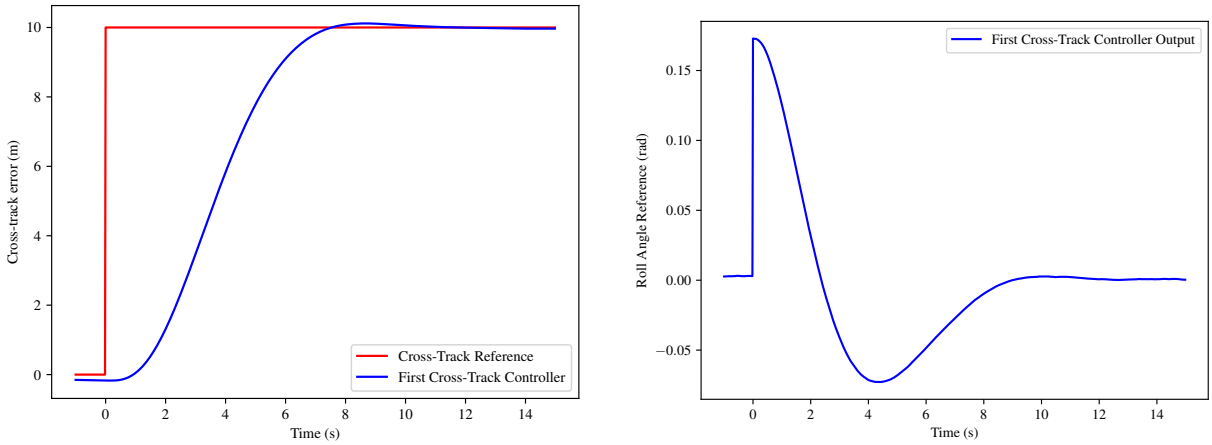
(a) Roll Angle Step Response **(b)** Roll Rate Command

Figure 7.8: Roll angle step response and corresponding roll rate command for the roll angle controller.

The non-linear roll angle step response has a 1% overshoot and a 2% settling time of 2 seconds, which are within the requirements of the controller. The roll angle controller requirements are minimal overshoot and a 2% settling time of less than 3 seconds. These step response metrics match the linear roll angle step response values, which shows that there is minimal difference in the roll angle characteristics between the linear and non-linear model. Similar to the linear response, the non-linear roll angle response has a steady-state error which is due to the roll angle controller not having an integrator. This error will be dealt with by the outer cross-track controller. The roll rate reference commanded by the roll angle controller is not limited. However, the maximum aileron deflection produced by the roll rate controller to achieve the commanded reference is -2.58°, which is physically attainable.

7.3.1.5 First Cross-Track Controller

The cross-track controller is tested with the aircraft being at a constant altitude and airspeed. Figure 7.9 shows the first cross-track controller step response and the associated roll angle command commanded by the controller.



(a) Cross-Track Step Response **(b)** Roll Angle Command

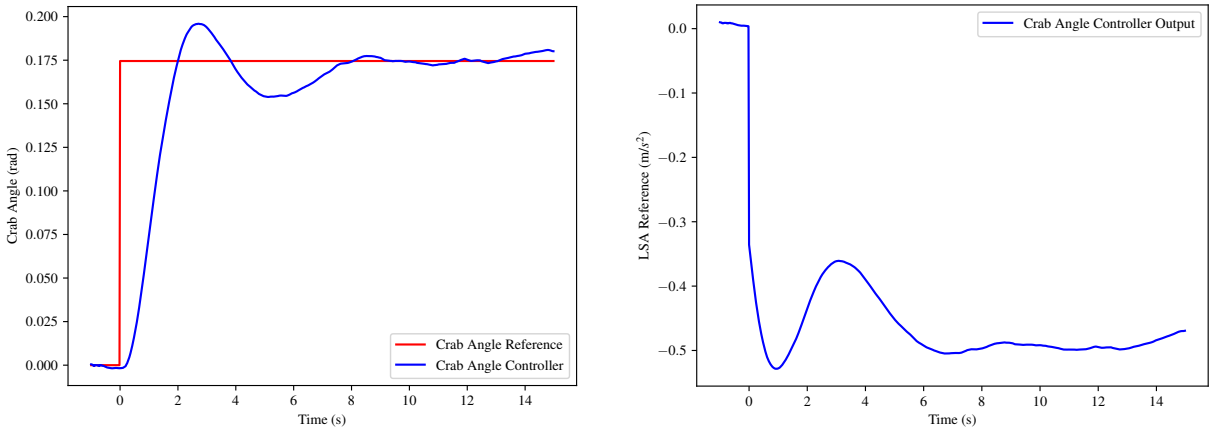
Figure 7.9: Cross-Track step response and corresponding roll angle command for the first cross-track controller.

The non-linear cross-track controller response has a 2% settling time of 6.98 seconds which is within the requirements of the controller, and is slightly better then the linear response. The first cross-track controller requirement is a 2% settling time of less than 13 seconds. The non-linear response does not have a steady-state error, which shows that the first cross-track controller compensates for the steady-state error of the roll angle controller. The roll angle reference commanded by the first cross-track controller is within the $\pm 30°$ ($\pm\frac{\pi}{6}$ rad) bound for the reference.

7.3.1.6 Crab Angle Controller

The crab angle controller is tested with the aircraft flying at a constant airspeed and altitude. The crab angle for the non-linear simulation and the practical vehicle is the difference between the aircraft heading and the ground track heading. Figure 7.10 shows the crab angle controller step response and the associated LSA command produced by the controller.

The non-linear crab angle response has a rise time of 1.77 seconds which is within the controller requirement and is faster than the linear response. The crab angle controller requirement is a rise time of less than 3 seconds. The non-linear response has much higher overshoot and undershoot compared to the linear response. This is due to the lateral dynamics used by the cross-track controller having a larger impact on the directional dynamics used by the crab angle controller than expected. The cross-track controller will therefore have a greater influence on the crab angle controller's response. Since the



(a) Crab Angle Step Response **(b)** LSA Command

Figure 7.10: Crab angle step response and corresponding LSA command for the crab angle controller.

crab angle controller is only used to perform the de-crab manoeuvre, it is more important for the aircraft to approximately align itself with the runway as fast as possible than to have an accurate alignment. This is achieved by the controller with a fast rise time, and therefore its performance is acceptable. The LSA reference commanded by the crab angle controller is within the $\pm g$ limit, and is therefore practically realisable.

7.3.2 Stationary Runway and Moving Platform Landing

Now that the performance of the individual controllers in the FCS has been analysed, the behaviour of the complete control system in simulation can be tested. First the aircraft's waypoint navigation behaviour will be tested. Then the stationary runway landing performance will be tested. Finally, the moving platform landing performance will be tested. No wind effect will be active for this section's SITL simulation so that the control system's pure landing performance, unaffected by external disturbances, can be assessed. Wind will be added in Section 7.3.3, where its effect on the landing performance will be tested. Sensor noise, however, is present in the SITL simulation. For landing scenarios, the performance of both the classical airspeed and altitude controllers, as well as the MPC, will be evaluated. The classical airspeed and altitude controllers will be used as a baseline for the MPC, as previously mentioned. The baseline will provide a gauge on the MPC's performance to determine if there is any improvement when using the MPC.

7.3.2.1 Waypoint Navigation

By default, the control system navigates the aircraft around the airfield until a controller test or land command is given, which will change the system's behaviour. Figure 7.11 shows the aircraft's simulated flight path around the airfield as it follows the waypoints in SITL simulation.

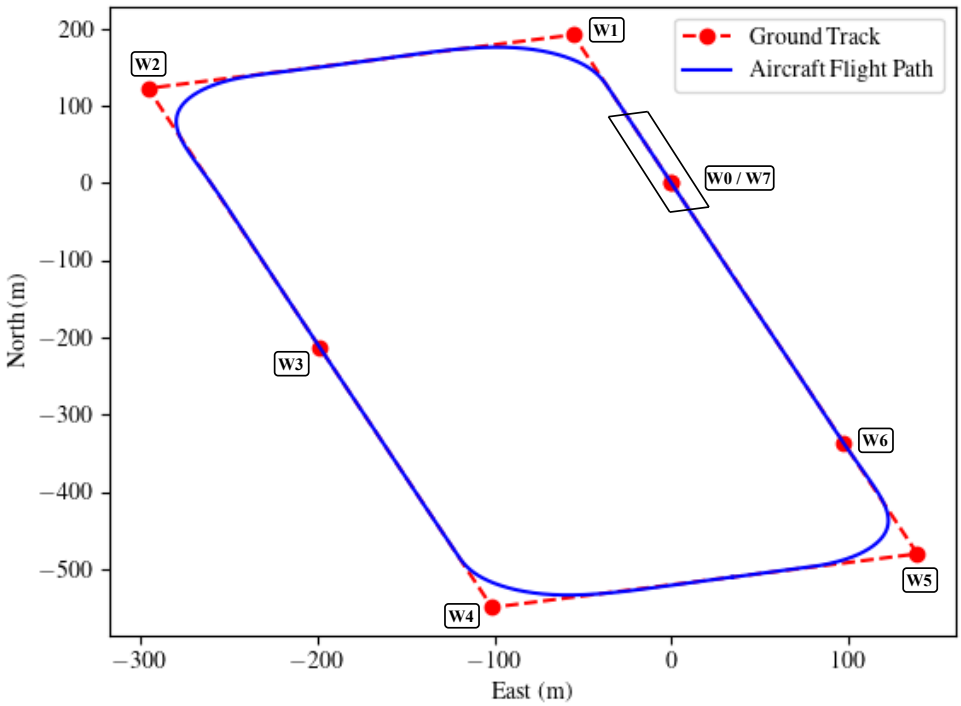


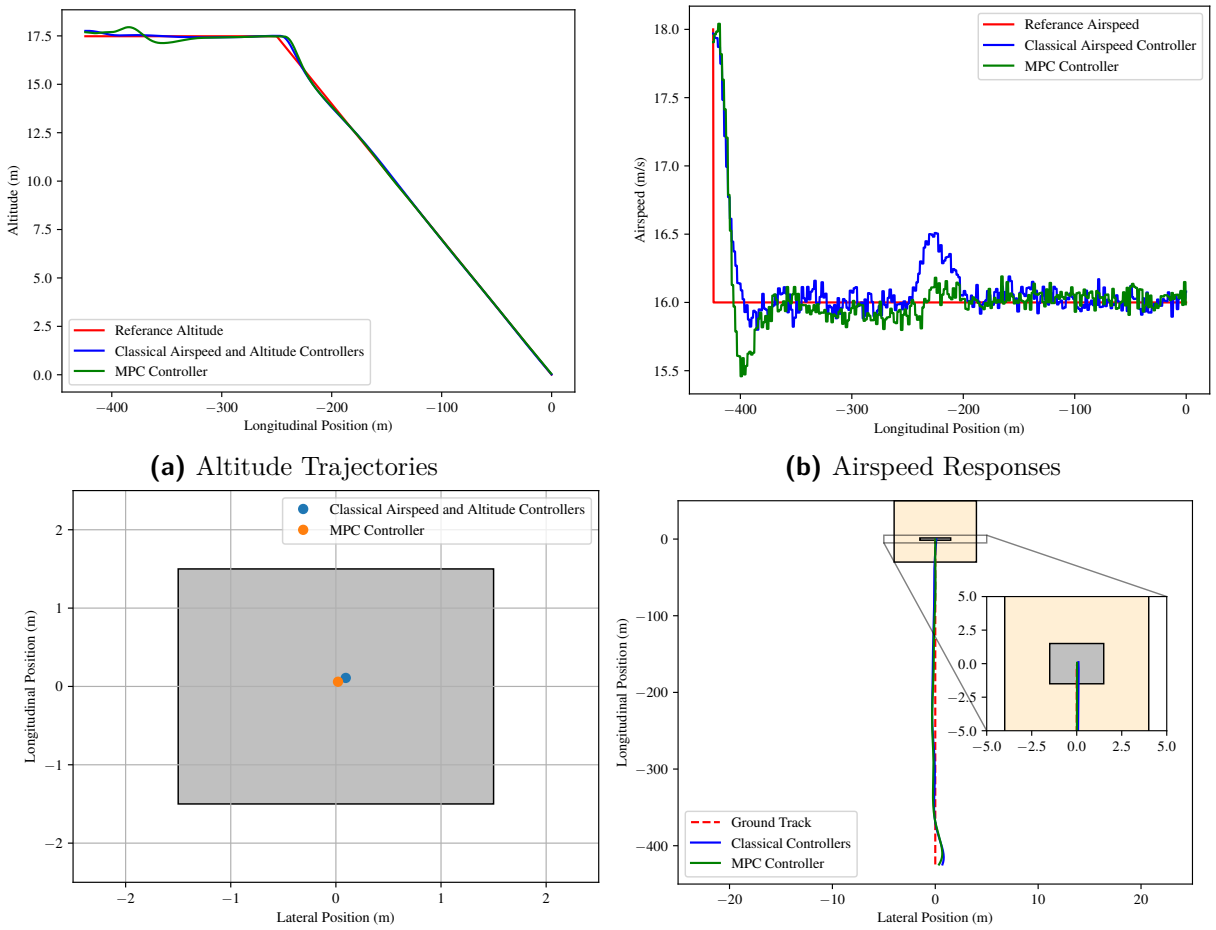
Figure 7.11: Diagram showing the ground track reference and the aircraft flight path in the SITL simulation. The waypoints forming the ground track are labeled and the runway diagram is added at waypoint 0 / 7.

The waypoint locations are chosen based on the space available at the physical airfield. The runway heading at the airfield was determined to be $\psi_r = -16.123°$. Therefore, the waypoints that form the circuit are adjusted to ensure that the circuit's final leg is aligned with the runway. Waypoint 0/7 represents the location where the aircraft is armed on the runway, and waypoint 3 is added to ensure the aircraft follows the correct groundtrack when the autopilot is engaged. Waypoint 6 is the location where waypoint navigation ends if the aircraft is commanded to land. Figure 7.11 shows that the aircraft is able to accurately follow the ground track. The early waypoint switching method allows the aircraft to be inline with the runway on final approach and this should enable the aircraft to have a high lateral landing accuracy.

7.3.2.2 Runway Landing

Figure 7.12 shows the aircraft's performance for a runway landing in SITL simulation, using either the classical altitude and airspeed controllers or the MPC controller.

The longitudinal position in the figures is the in-track position of the aircraft along the runway with respect to the intended touchdown point. The lateral position in the figures is the cross-track position of the aircraft perpendicular to the runway with respect to the intended touchdown point. In Figure 7.12d, the cream rectangle represents the runway and the grey rectangle represents the bounding box on the runway for the runway landing. All the plots, besides Figure 7.12c, start when waypoint navigation ends for the aircraft,



(a) Altitude Trajectories

(b) Airspeed Responses

(c) Touchdown Locations

(d) Horizontal Position Trajectories

Figure 7.12: The aircraft trajectories, airspeed and touchdown locations for a runway landing in SITL simulation, using either the classical altitude and airspeed controllers or the MPC controller.

and the plots end when the aircraft touches down on the runway.

For the runway landing, the aircraft is slowed down to 16 m/s when it is on final approach. This is shown in Figure 7.12b where the airspeed reference is reduced to 16 m/s from the trim speed of 18 m/s. Both the classical airspeed controller and the MPC are able to slow the aircraft down. The MPC uses the altitude to reduce the airspeed, which is manifested as a small oscillation at the beginning of the MPC altitude trajectory shown in Figure 7.12a. The MPC recovers from the oscillation before it reaches the glide slope and therefore its glide slope tracking performance is not influenced. The aircraft starts to descend to capture the glide slope at a longitudinal position of -250 m from the intended touchdown point. As the aircraft descends, its airspeed increases however both the classical airspeed controller and the MPC are able to return the airspeed back to the reference. The MPC has better control over the airspeed compared to the classical airspeed controller, as the MPC's airspeed peak is lower than the classical airspeed controller's peak when the descent begins. This is due to the MPC being able to balance the airspeed and altitude errors as the aircraft descends. The classical airspeed controller can only

respond to the airspeed disturbances that are caused by the aircraft descending with the classical altitude controller.

As shown in Figure 7.12a, both the classical altitude controller and the MPC can track the glide slope accurately, however, the MPC performs better as it has a slightly lower altitude error. Consequently as shown in Figure 7.12c, the MPC has better landing accuracy with an in-track error of 6 cm and a cross-track error of 2 cm while the classical controllers achieve an in-track error of 11 cm and a cross-track error of 9 cm. The in-track error is defined as the longitudinal distance between the aircraft and the desired touchdown point. The landing accuracy for both controllers can vary by a few centimetres from the reported values due to the noise of the sensors and the EKF. As the virtual platform has a size of 3 m by 3 m, the maximum in-track and cross-track errors that the aircraft may have is ± 1.5 m. Both the controllers have landing accuracies that are well within these limits, and therefore their performance is acceptable.

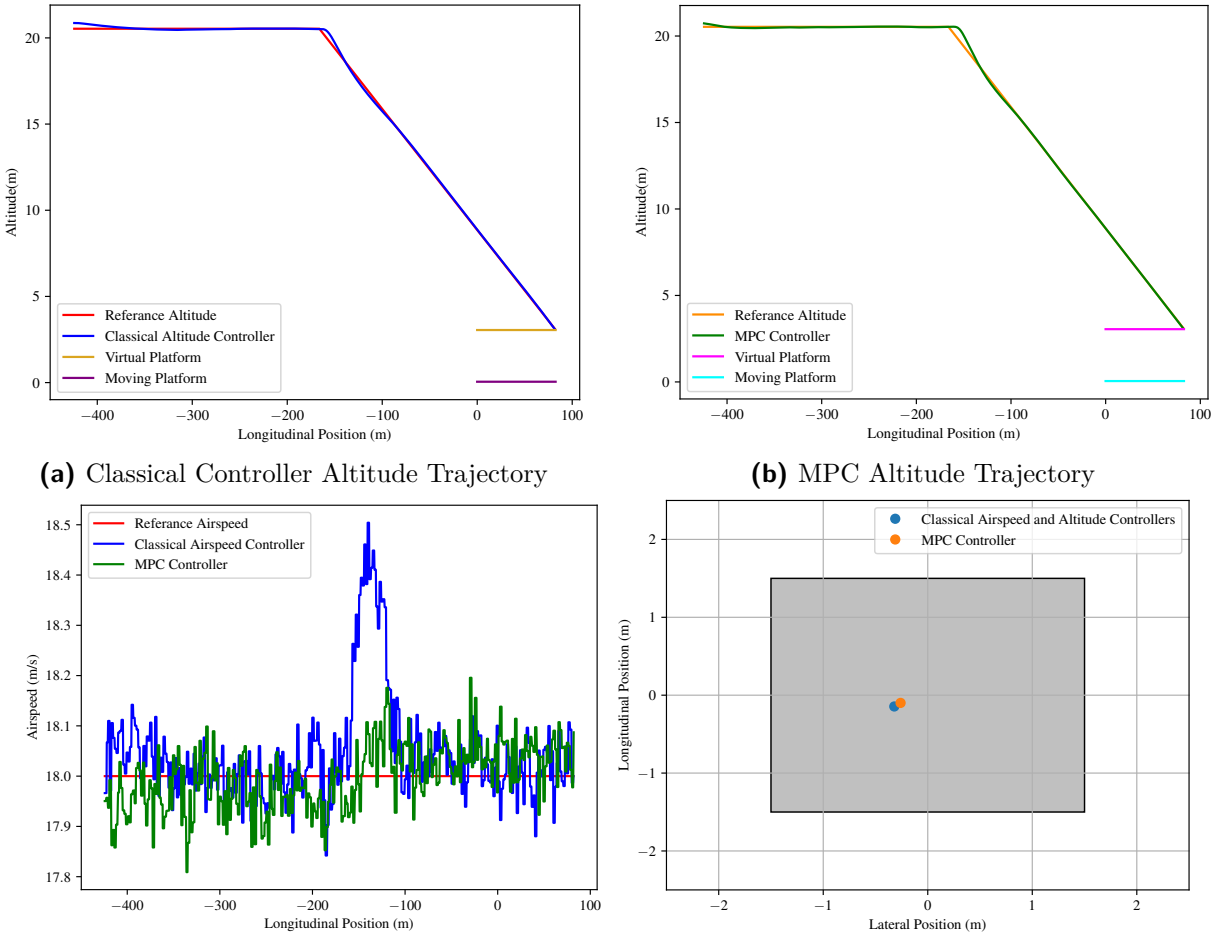
Figure 7.12d shows the lateral tracking performance of the aircraft for the runway landing. The same cross-track controller is used for the classical controllers and MPC landing. This results in their trajectories having similar lateral tracking performance. For both trajectories, the cross-track controller can reduce the cross-track error to a centimetre level before it reaches the intended touchdown point. This indicates that, at least in ideal conditions, the cross-track controller is sufficient for lateral tracking.

7.3.2.3 Moving Platform Landing

Now that it has been verified that both controllers can land the aircraft on a runway, their moving platform landing performance can be tested. Figure 7.13 shows the aircraft's performance for a moving platform landing in SITL simulation, using either the classical altitude and airspeed controllers or the MPC controller. The longitudinal position in the figures, besides Figure 7.13d, is the in-track position of the aircraft along the runway with respect to the runway frame's origin. The longitudinal and lateral positions in Figure 7.13d are the in-track and cross-track distances from the centre of the virtual platform.

As described in section 3.3.3, the virtual and moving platforms share the same longitudinal and lateral positions, but the virtual platform is 3 m higher than the moving platform. The virtual platform is the target for the UAV to land on for the moving platform landing scenario. Both the virtual and the moving platforms move at a speed of 3 m/s at a constant heading.

The aircraft is kept at the trim airspeed therefore Figure 7.13b shows that the MPC does not exhibit an oscillation in altitude before tracking the glide slope, as was the case for the runway landing. The aircraft begins to descend on the glide slope at a longitudinal position of around -160 m. The aircraft's airspeed begins to increase and, as shown in Figure 7.13c, the classical airspeed controller and MPC can keep the airspeed close to the reference. Once again, the MPC can control the airspeed better than the classical airspeed



(a) Classical Controller Altitude Trajectory

(b) MPC Altitude Trajectory

(c) Airspeed Responses

(d) Touchdown Locations

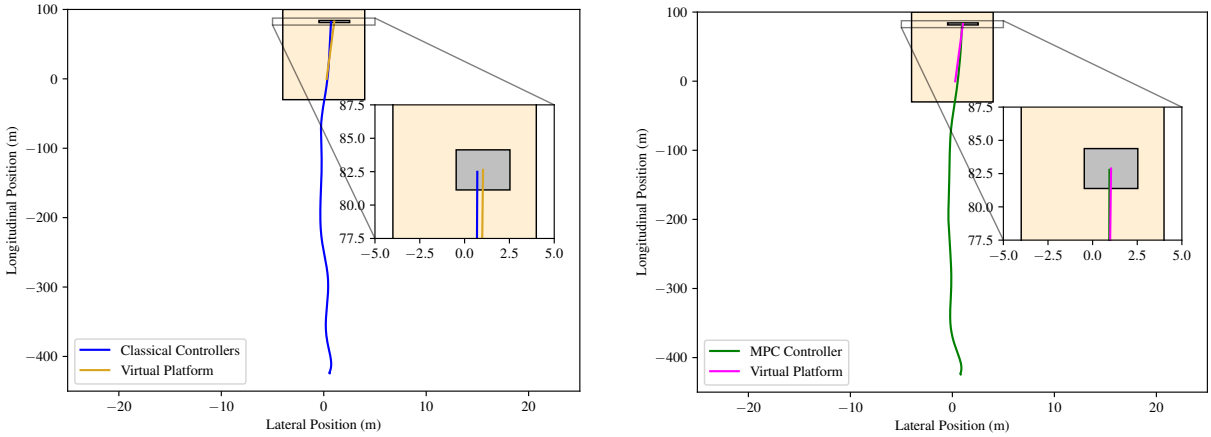
Figure 7.13: The aircraft altitude trajectories, airspeed and touchdown locations for a moving platform landing in SITL simulation, using either the classical altitude and airspeed controllers or the MPC controller.

controller, as the MPC's airspeed peak is lower than the classical airspeed controller's peak.

Figure 7.13a and Figure 7.13b show that both the classical altitude controller and the MPC can track the glide slope fairly well. However, just as with the runway landing, the MPC performs better due to its response having less altitude error when following the glide slope reference. The MPC's landing accuracy is therefore better with an in-track error of 10 cm and a cross-track error of 26 cm, while the classical controllers' accuracy has an in-track error of 15 cm and a cross-track error of 32 cm. These touchdown points are shown in Figure 7.13d. Both the MPC and classical controllers' in-track errors are slightly larger than their stationary runway landing errors. This is due to the predicted touchdown point varying continuously as the aircraft and platform states change, and this consequently causes the glide slope altitude references to also vary. The aircraft also has a higher ground speed for the moving platform landing, as it is not slowed down, which gives it less time to capture the glide slope. This is a more difficult scenario for the controllers to handle, hence the degraded performance. The cross-track errors are significantly higher

than their runway landing equivalent values which is due to the aircraft tracking the platform's lateral position. The aircraft's lateral performance is not as responsive as the moving platform, causing increased error. Nonetheless, both controllers can land on the virtual platform within the 1.5 m limits for in-track and cross-track error, and therefore their performance is acceptable.

The moving platform model is given a small velocity perpendicular to the runway so that it can be slightly offset from the centreline. This is done to test that the aircraft can track the platform even if the platform is not aligned with the centreline. Figures 7.14a and 7.14b show the lateral tracking performance of the cross-track controller when executing the moving platform landing in the SITL simulation, using either the classical controllers or the MPC. The lateral position in the figures is the cross-track position of the aircraft perpendicular to the runway with respect to the runway frame's origin. The figures show that the virtual platform, and hence the moving platform, have a trajectory slightly towards the right of the runway centreline. The cross-track controller can track the moving platform fairly well as the cross-track error just before touchdown is in the centimetre level.



(a) Classical Controller Horizontal Trajectory **(b)** MPC Horizontal Trajectory

Figure 7.14: The aircraft horizontal plane position trajectories for a moving platform landing in SITL simulation, using either the classical altitude and airspeed controllers or the MPC controller.

7.3.3 Runway and Moving Platform Landings with Wind

The wind models from section 4.4 were implemented in the SITL simulation so that the control system performance can be tested when the aircraft experiences wind disturbances. Since the aircraft is small, it is very sensitive to changes in the wind. Therefore, the control system needs to be able to compensate for wind, to be effective on the practical UAV. The maximum wind speed at which the aircraft will be expected to land is 3.1 m/s. This is because when a crosswind has this wind speed, it requires the aircraft to have the maximum acceptable crab angle (10°) for a landing. The gust, turbulence, wind

shear, and ground effect models were implemented in the gazebo aerodynamic plugin. The turbulence, wind shear, and ground effect models are reactive, as they depend on the aircraft states. The gust model values, however, need to be selected depending on what type of gust is desired. The gust amplitude is set to the maximum wind speed value. The gust is only applied when the aircraft is on the glide slope as the glide slope tracking performance determines the landing accuracy. The gust is applied as a headwind, tailwind, starboard-side crosswind, and port-side crosswind.

The stationary runway and moving platform landings were retested with the different wind conditions, but only for the MPC controller as it was determined as the better controller in the previous section.

7.3.3.1 Runway Landing

Figure 7.15 shows the touchdown points in SITL simulation, for the aircraft performing the runway landing in different wind conditions. The ideal touchdown point is the origin, with zero in-track and cross-track errors. The MPC controller is used to perform the landings.

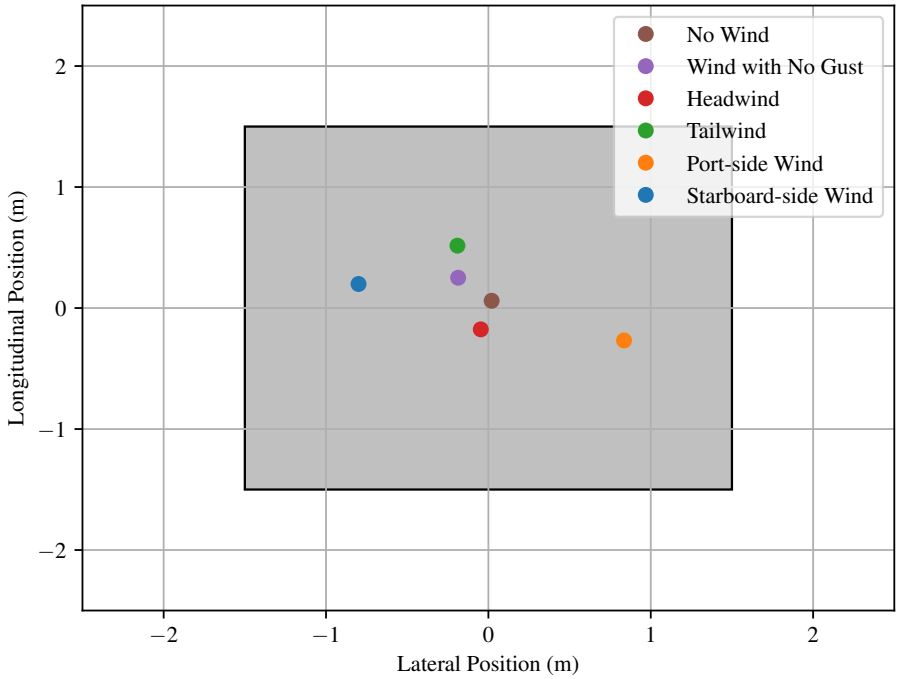


Figure 7.15: Stationary runway landing touchdown locations in SITL simulation, for the aircraft in different wind conditions.

The landing accuracy degrades relative to the ideal landing performance when wind is introduced. The runway landings in wind have higher in-track and cross-track errors than the runway landings with no wind. However, all of the touchdown points are within the required 3 m x 3 m landing zone. Gust has the most significant impact on the landing accuracy of the aircraft compared to the other wind effects and therefore its results are

distinctly shown. The crosswind touchdown points have a slight increase in in-track error and a large increase in cross-track error when compared to the no wind touchdown point. This is expected as the de-crabbing manoeuvre causes the aircraft to align with the runway and therefore to no longer counteract the wind, increasing the cross-track error. The tailwind causes the aircraft to significantly overshoot the desired touchdown point as the aircraft's ground speed is much higher than its airspeed. Conversely, the headwind causes the aircraft to only slightly undershoot the desired touchdown point due to a slower ground speed. The ground speed determines how fast the aircraft completes the glide slope. Therefore, a faster ground speed results in the aircraft having less time to capture the glide slope. The tailwind decreases the time the aircraft has to correct its altitude error, while the headwind provides more time, hence the difference in landing accuracy. All the touchdown points are within the 1.5 m limit for the in-track and cross-track errors therefore the control system response is adequate for a runway landing. The control system performance can now be tested for the moving platform landing with wind scenario.

7.3.3.2 Moving Platform Landing

Figure 7.16 shows the touchdown points in SITL simulation, for the aircraft performing the moving platform landing in different wind conditions. The MPC controller is used to perform the landings.

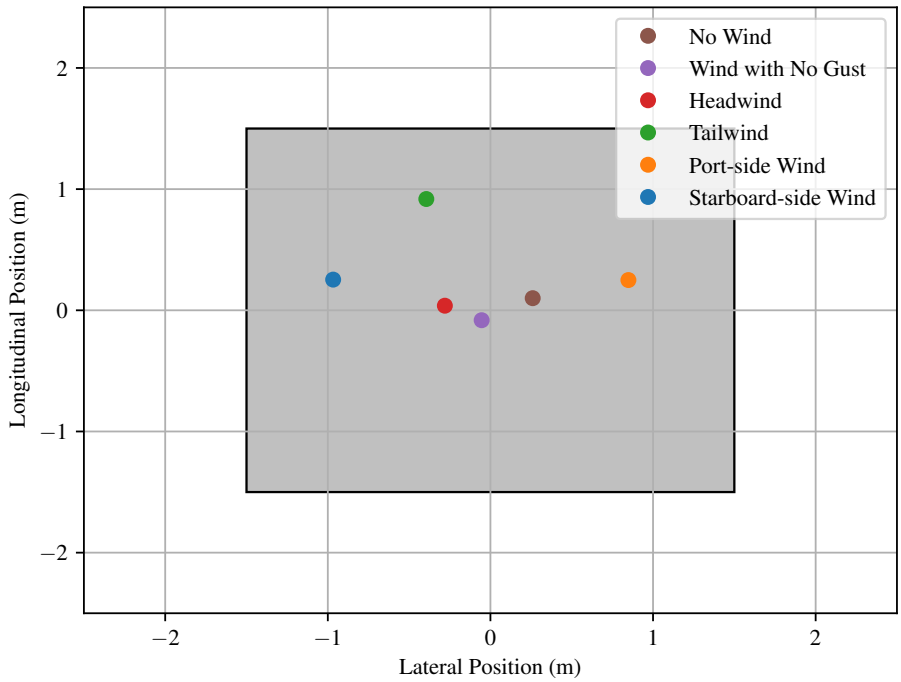
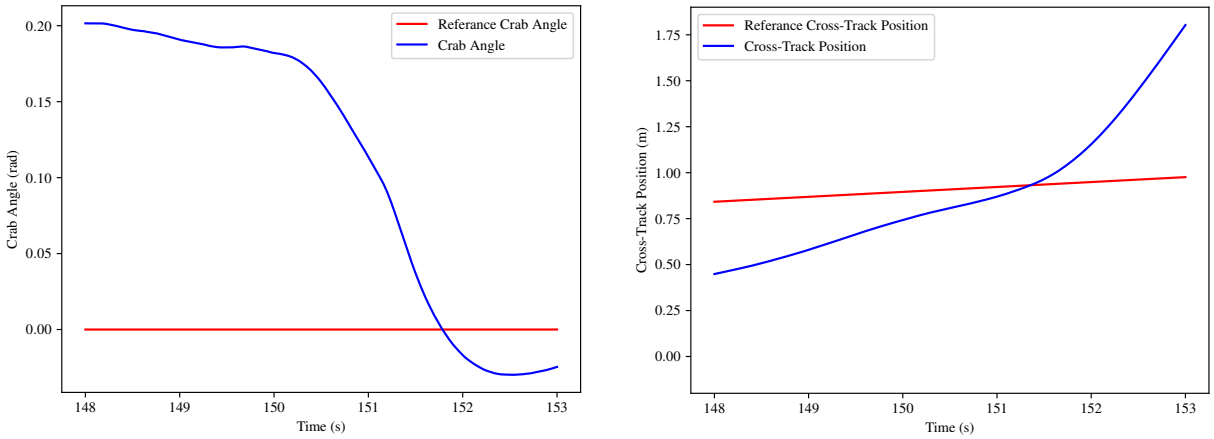


Figure 7.16: Moving platform landing touchdown locations in SITL simulation, for the aircraft in different wind conditions.

All of the touchdown points are within the required 3 m x 3 m landing zone on the virtual moving platform. The wind with no gust landing has similar performance to the

no wind landing. However, the latter has increased cross-track error. This is due to the aircraft tracking the lateral position of the platform, which gradually changes, introducing variance in the cross-track error on landing. The crosswind landings significantly increased the cross-track error due to the aircraft performing the de-crab manoeuvre, while the in-track error only slightly increased, which is expected. The tailwind landing once again caused the aircraft to significantly overshoot the intended touchdown point due to the aircraft's increased ground speed. The time the aircraft has to capture the glide slope for the moving platform landing is shorter than the runway landing as the aircraft is not slowed down, and this time is further reduced by the tailwind. The headwind landing has the least in-track error on touchdown as the aircraft in this wind condition has the lowest ground speed and hence the most time to capture the glide slope. All the touchdown points are within the 1.5m limits for the in-track and cross-track errors. Therefore, the control system can land the aircraft onto the moving platform in the SITL simulation within the specified wind limits.

Figure 7.17 shows the aircraft performing the de-crab manoeuvre in SITL simulation when executing the port-side wind moving platform landing.



(a) Aircraft Crab Angle **(b)** Aircraft Cross-Track Position

Figure 7.17: The aircraft's crab angle and cross-track position in SITL simulation, when performing the port-side wind moving platform landing.

The plots are shown during the last few seconds that lead to the aircraft touching down. Figure 7.17a shows that just before performing the de-crab manoeuvre, the aircraft has a crab angle of approximately 0.175 rad (10°). At touchdown, the aircraft has a crab angle of approximately -0.03 rad (−1.72°). While the aircraft at touchdown is not perfectly aligned with the moving platform, the crab angle controller is able to reduce the crab angle fairly quickly, which is its main purpose. As has been previously mentioned, the aircraft's cross-track error (aircraft's cross-track position with respect to the reference cross-track position) increases rapidly when the de-crab manoeuvre is executed, as the aircraft no longer counteracts the wind. The rapid increase in cross-track error is shown at the end of Figure 7.17b. At touchdown, the cross-track error is still within the 1.5 m limit,

which is due to the crab angle controller quickly minimising the crab angle, allowing for the controller's late activation. It can therefore be deduced that the crab angle controller is able to successfully execute the de-crab manoeuvre in the SITL simulation.

7.4. Summary

This chapter first discussed the Simulink non-linear model. This was followed by describing the software-in-the-loop implementation. The individual controllers in the FCS were then tested in the SITL simulation, and their step responses were analysed. All the tested controllers performed well, as their step responses were within their designed requirements. The MPC had better altitude control compared to the classical altitude controller, but slightly worse airspeed control compared to the classical airspeed controller. The runway and moving platform landings were then tested in the SITL simulation with no wind. The designed control systems could successfully execute the landing scenarios, with the MPC obtaining better landing accuracies compared to the classical airspeed and altitude controllers. The control systems were then tested to execute the runway and moving platform landings during different wind conditions. The designed control systems could successfully land the aircraft, within the maximum limits, on the runway and the moving platform for the wind conditions specified. It has now been determined that the designed control system can land the aircraft on the runway and the moving platform in the non-linear simulations. This provides more confidence that the control systems should be capable of controlling the physical UAV in real life. The focus now shifts to verifying the control systems' real-world performance by performing practical flight tests. The outcome of these tests will be discussed in the next chapter.

Chapter 8

Practical Tests Overview and Results

This chapter presents the practical flight tests that were performed to verify the control systems' performance on a physical UAV in the real world. The hardware used for the practical flight tests was already introduced in Chapter 3. This chapter will first provide an overview of the practical flight test logistics and then discuss the practical flight test campaign. This chapter will then conclude by examining the practical flight test results and determining if the developed control systems can land the physical UAV onto a moving platform.

8.1. Practical Flight Test Logistics

This section presents the environment where the practical flight tests were performed and then discusses the procedure used during the practical flight tests.

8.1.1 Practical Flight Test Environment

The practical flight tests were performed at the Helderberg Radio Flyers (HRF) Club near Maccassar, South Africa. The satellite view of the HRF club is shown in Figure 8.1.

The flight tests were performed on days with high visibility and low wind (less than 6 knots) as the aircraft flies far from the runway and is susceptible to wind. The flight tests were done in the morning when the wind was the lowest. The circuit around airfield can only be completed in an anti-clockwise direction due to electric pylons and the N2 highway limiting the direction of travel north of the runway. The touchdown location for a stationary runway landing is the start of the runway so that the UAV has enough distance on the runway to come to a complete stop. The HRF runway length is 150 m long, therefore the moving platform needs to travel a distance less than the runway length during the moving platform landing so that no runway excursion occurs. It was found that if the moving platform's speed is limited to 3 m/s, then its travelling distance will be well within the runway length, and therefore the moving platform's speed was set to this value.

Figure 8.1: Aerial view of the HRF airfield and runway. The image is obtained from Google Maps [72].

8.1.2 Practical Flight Test Procedure

A practical flight test refers to all the objectives to be completed on a flight test day. These objectives are usually testing individual controllers or completing a landing scenario. A practical flight session refers to the duration between when the UAV takes off from the runway to the time the UAV lands back on the same runway. The sensors on the UAV are calibrated at the start of the flight test day by following the instructions given in QGC.

The procedure followed for a practical flight session consists of first placing the UAV on the runway and then have the pilot take off the UAV from the runway. The pilot then manoeuvres the aircraft into level flight on a long straight and switches to Autopilot mode. The autopilot will then capture the altitude at this point and look for the closest waypoint, which it will use to join the circuit. The aircraft then continuously circuits around the airfield until a command is given by the GCS operator. The operator will enter the UAV into the appropriate sub-mode to complete the flight session objective. If the objective is not a landing scenario, then after the objective has been completed, the pilot will switch the UAV into Manual mode and land the UAV back on the runway.

Checklists are used to ensure that all the tasks required for the flight test are completed. The checklists used for the flight test are shown in appendix C.

8.2. Flight Test Campaign

The practical flight tests began once the control systems were implemented on the hardware and the physical fixed-wing UAV was fully assembled. The ideal progression of the flight tests consisted of first manually flying the aircraft and then gradually activating the controllers, increasing the autonomy of the autopilot. The first flight involved the human safety pilot manually flying the UAV around the airfield. This was to ensure that the UAV was controllable and maintained stabilised flight.

The following flight tests involved testing the individual controllers in the FCS, by commanding step responses when the fixed-wing UAV was on long straights in the circuit. First, the longitudinal controllers were tested, with the safety pilot still having manual lateral control of the UAV. Next, the lateral controllers were activated, and the UAV flew fully autonomously. The MPC was then tested to ensure that it had adequate control of the UAV's altitude and airspeed. It should be noted that some of the flight tests were redone, as the practical data obtained was used to improve the aircraft model. The improved model required the controllers to be retuned and then retested. In the end, all the controllers performed successfully on the practical vehicle.

After the FCS controllers' practical performance was approved, the landing scenarios could then be practically tested. The runway landing scenario was first performed using the classical airspeed and altitude controllers and then the MPC. Once their runway landing performances were deemed sufficient, the moving platform landing tests were then performed. All the flight tests were successfully completed, except for the moving platform landing onto the physical moving platform. Due to late complications with the DGPS configuration on the moving platform hardware, the moving platform landing was instead performed onto a virtual moving platform. This will be discussed in more detail later in this chapter.

8.3. Practical Flight Test Results

This section presents the practical flight test results. The first subsection analyses the individual FCS controller responses. The second subsection discusses the navigational performance of the control system in circuiting the aircraft around the airfield. The third subsection analyses the practical runway landing performances, and the final two subsections discuss the moving platform landing results.

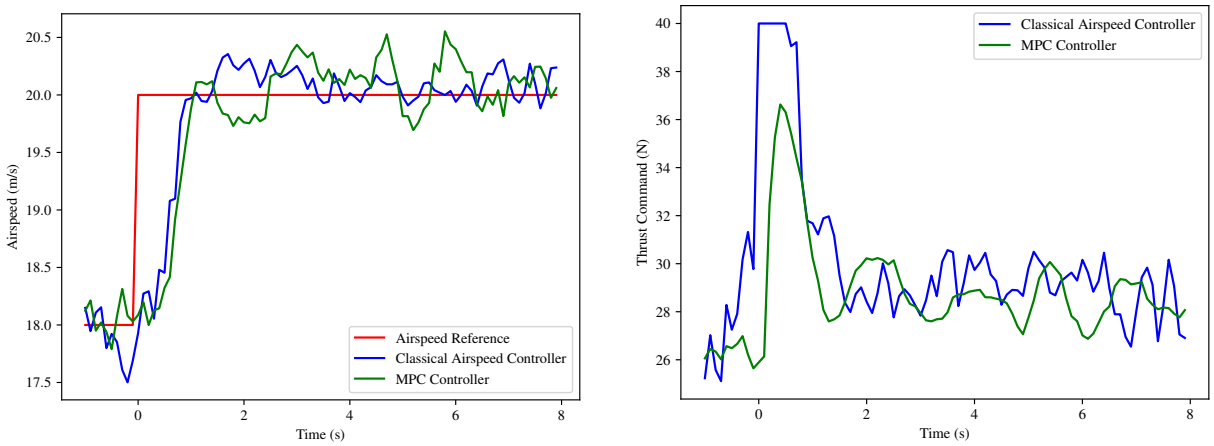
8.3.1 Flight Controller Responses

The individual controllers that were tested in SITL were retested on the practical UAV to verify their real world performance. The inner-most controllers were not tested as they

cause the UAV to depart from trim flight. During initial flight testing, the controllers were activated in stages to gain confidence in them. The pilot would still have partial control of the UAV until all the controllers were active. For example, when only the longitudinal controllers were active, the pilot was instructed to control the UAV's lateral dynamics. For the controller responses shown in this subsection, all the controllers were active so that the practical results can be fairly compared with the SITL results.

8.3.1.1 Airspeed Controller

Figure 8.2 shows the airspeed step response obtained from the practical flight data, for the classical airspeed and MPC controllers, with their associated thrust commands.



(a) Airspeed Step Response **(b)** Thrust Command

Figure 8.2: Airspeed step response and corresponding thrust command for the classical airspeed and MPC controllers operating on the physical UAV.

Both controllers can regulate the airspeed state with reasonable performance, however, both their responses contain more steady-state variations than the corresponding SITL results. These variations contain high-frequency components caused by sensor noise and low-frequency components caused by disturbances and model uncertainty. The disturbances refer to the wind whose velocity changes continuously, affecting the airspeed. The model uncertainty refers to the uncertainty in thrust produced, which is due to the changing atmospheric conditions and the motor's dependence on the battery voltage.

The classical airspeed step response has a rise time of 0.82 seconds and zero steady-state error. The overshoot of the response is difficult to determine due to the steady-state variations. However, it seems to be less than 20%. The rise time practically measured is faster than the simulated rise time in the SITL simulation. This is most likely due to the thrust produced by the motor being higher than the 40N limit in the model. This is expected as the thrust limit was conservatively chosen. Therefore if the maximum thrust produced deviated from this value, it would impact performance.

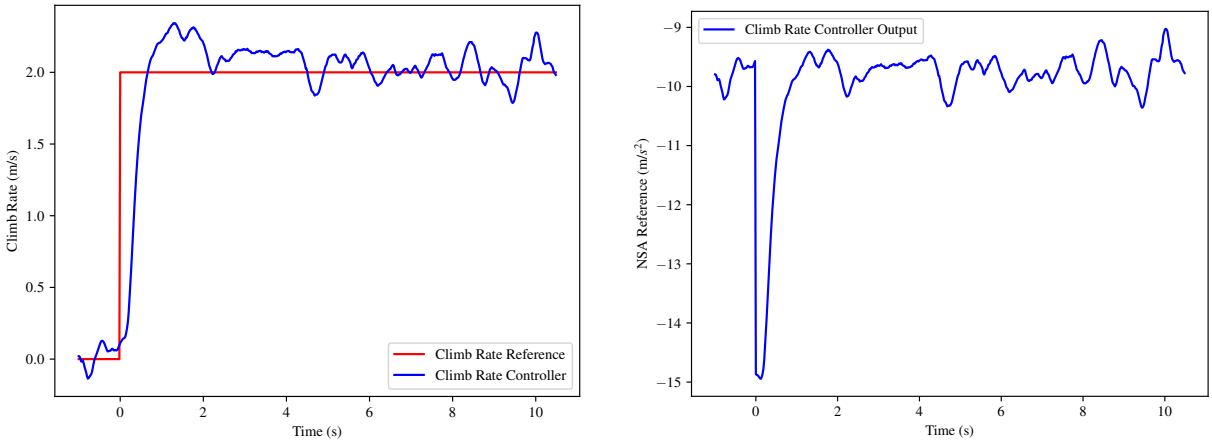
The MPC response has a rise time of 0.98 seconds and zero steady-state error. Once

again the overshoot is difficult to determine due to the steady-state variations. However, it seems to be less than 20%. The rise time and steady-state error of the practically measured response both improve on those of the SITL simulated response. The steady-state error improvement is due to the addition of the K_{T_c} scalar, which compensates for some of the thrust model inaccuracies.

The classical airspeed controller performs better than the MPC controller due to its response having superior transient response characteristics. The two controllers were tested on different days with dissimilar weather conditions which may have affected their performances slightly differently. This was not by choice but was due to the MPC requiring a readjustment and hence a retest. The thrust commands of both controllers in Figure 8.2b are within the limits set for the motor. The main function of the airspeed controllers is to maintain the airspeed state, which both controllers achieve, and therefore their results are acceptable.

8.3.1.2 Climb Rate Controller

Figure 8.3 shows the practically measured climb rate step response and the associated NSA command for the classical climb rate controller.



(a) Climb Rate Step Response **(b)** NSA Command

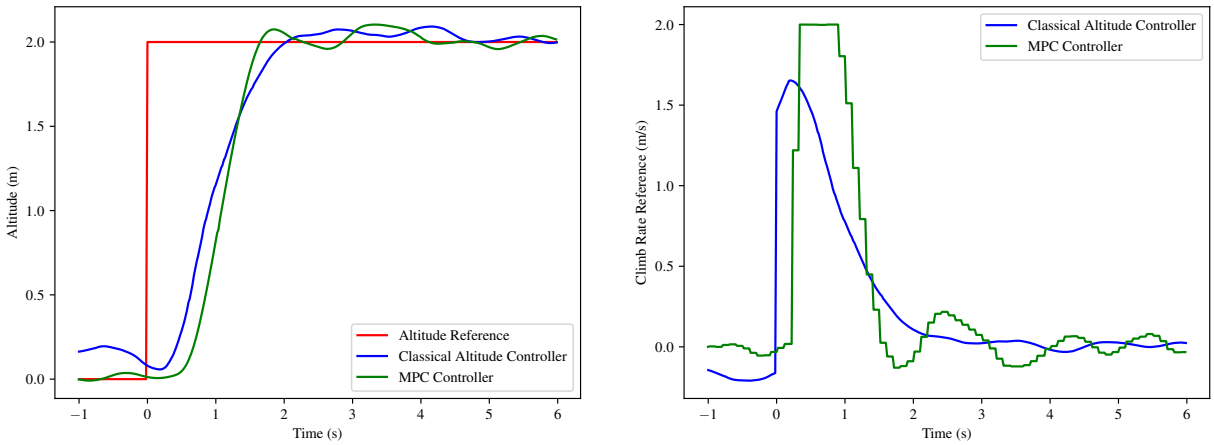
Figure 8.3: Climb Rate step response and corresponding NSA command for the climb rate controller operating on the physical UAV.

The climb rate response has a rise time of 0.56 seconds, an overshoot of 17.15%, and zero steady-state error, which are within the controller requirements. Unfortunately, no 2% settling time can obtained as the response continuously deviates beyond the 2% envelope due to external disturbances. This behaviour is partially caused by the imperfect control surface deflection on the UAV which is attributed to the mechanical linkages that connect to them. De Bruin [16] found that these linkages contain backlash that contribute to oscillatory behaviour. Since the same airframe as De Bruin is used, this effect would still be present in this research project's responses. The practical PX4 software mixer also

approximately produces the commanded deflections. The climb rate state produced by the EKF also contains some noise. When considering all these factors, the climb rate response is acceptable, as it can sufficiently regulate the climb rate state so that the altitude controllers can perform their function. The NSA command produced by the climb rate controller is within its specified limits.

8.3.1.3 Altitude Controller

Figure 8.4 shows the altitude step response, obtained from the practical flight data, for the classical altitude and MPC controllers, with their associated climb rate commands.



(a) Altitude Step Response **(b)** Climb Rate Command

Figure 8.4: Altitude step response and corresponding climb rate command for the classical altitude and MPC controllers operating on the physical UAV.

Both the classical altitude and MPC controller responses contain oscillation at steady-state, partially due to the imperfect control surface angle deflections mentioned in the climb rate controller section. The oscillation magnitude in the step response is low enough to perform accurate glide slope tracking for the landing tests and is therefore acceptable. The imperfect control surface deflections could be corrected with a more sophisticated mechanical mechanism to move the control surfaces. However, this is outside the scope of the current research project. The change in the environmental conditions also effects the UAV's aerodynamic performance and hence its response. Some of the control surfaces on the UAV were slightly warped due to usage which would impact performance.

The classical altitude response has a rise time of 1.64 seconds, an overshoot of 4.65%, and zero steady-state error. The classical altitude response also has a 2% settling time of 4.50 seconds, which is well within the controller requirement. The rise time and 2% settling time for the classical altitude controller are both faster on the physical UAV compared to the SITL simulation, which is due to the limited integrator being active on the UAV, improving its performance. The limited integrator provides additional climb rate command that is added to the climb rate command produced by the classical altitude

controller's proportional component.

The MPC response has a rise time of 1.49 seconds, a 5.15% overshoot, and zero steady-state error. The MPC response also has a 2% settling time of 5.20 seconds, which is well within the controller requirement. The MPC rise time and 2% settling time are both slower on the physical UAV compared to the SITL simulation. This is expected due to the MPC's model dependence. The MPC model does not capture all the uncertainties in the real world which will degrade its performance. The MPC's response has a delay of around 200 ms when it starts to respond to the step that is applied. This delay is the result of the MPC only running every 100 ms compared to the 20 ms execution of the classical controllers. Running the MPC at a faster rate would decrease the delay. However, the benefit of this would be minimal, and it would cost an increase in computational resources. The MPC is therefore left to run every 100 ms.

Comparing the classical altitude and MPC responses it can be seen that the classical altitude has a faster 2% settling time, while the MPC has a faster rise time. It is difficult to determine which controller performs better as their performance metrics are similar and are well within the requirements. Wind and other environmental changes cause the UAV to slightly deviate from the altitude reference at steady-state. This causes the responses to occasionally violate the 2% envelope about the reference. The MPC's altitude response reached steady-state faster than the classical altitude controller's response. However, the MPC's altitude response then violated the 2% envelope, which most likely was caused by an external wind disturbance. Wind disturbances will always be present in the controllers' response as they are tested in the real world. When acknowledging this point, it can be deduced that the MPC's altitude control is slightly better than the classical altitude controller. Nonetheless, both the controller responses are acceptable as they meet the requirements, and they both should be viable to track the glide slope for an accurate landing.

The climb rate references produced by both controllers, shown in Figure 8.4b, are within the limits set for the reference.

8.3.1.4 Roll Angle Controller

Figure 8.5 shows the roll angle step response obtained from the practical flight data for the roll angle controller. Unfortunately, the roll rate reference produced by the roll angle controller was not logged during the flight test and therefore cannot be plotted.

The roll angle reference step magnitude was set to 20° (0.349 rad). The roll angle response has steady-state variations due to imperfect aileron deflections and the roll angle state containing noise from the EKF. These variations have to be accepted on the UAV as no reasonable solution was available to mitigate this behaviour. This results in the response being unable to achieve a 2% settling time due to it continuously exiting the 2% envelope. Instead, the time that the response reaches steady state is examined and is

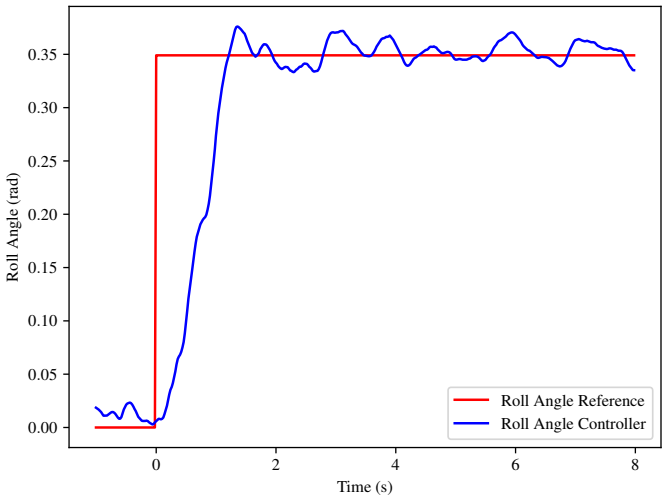


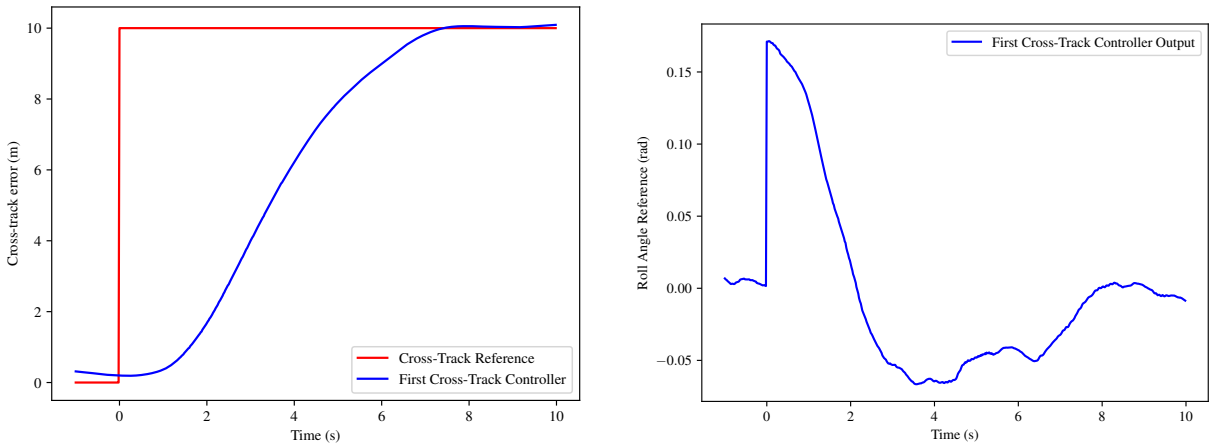
Figure 8.5: Roll angle step response for the roll angle controller operating on the physical UAV.

found to be 1.95 seconds, which is less than the 2% settling time requirement. The roll angle response has a 7.74% overshoot, which is higher than the overshoot in the SITL simulation. The roll angle response has a steady-state error. However, the variations make it difficult to obtain an exact value for the error. The steady-state error is small and should be compensated for by the outer first cross-track controller.

The roll angle response is acceptable even though the variations makes it difficult to measure the 2% settling time. The response's steady-state variations will be filtered out by the outer cross-track controllers making the response usable.

8.3.1.5 First Cross-Track Controller

Figure 8.6 shows the practically measured cross-track step response and the associated roll angle command for the first cross-track controller.



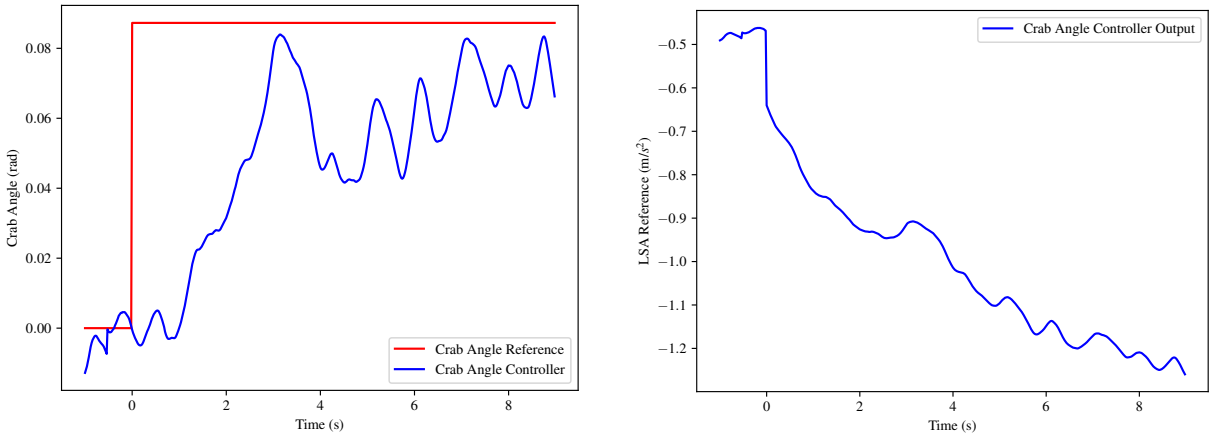
(a) Cross-Track Step Response **(b)** Roll Angle Command

Figure 8.6: Cross-track step response and corresponding roll angle command for the first cross-track controller operating on the physical UAV.

The cross-track response has a 2% settling time of 6.96 seconds and zero steady-state error. These results are similar to the SITL simulation response and are within the requirements for the controller. The cross-track response on the UAV is smooth, which shows that the first cross-track controller is able to tolerate the high-frequency variations of the roll angle response. The cross-track response's performance is acceptable to reduce the cross-error so that the UAV can have an accurate landing. It should be note that when a crosswind is applied to the UAV, the cross-track controller will attempt to reject the disturbance. However, if the crosswind is applied close to landing then there will be a cross-track error on landing, as was shown in simulation. The roll angle command, in Figure 8.6b, produced by the first cross-track controller is within its limits.

8.3.1.6 Crab Angle Controller

Figure 8.7 shows the practically measured crab angle step response and the associated LSA command for the crab angle controller. The step magnitude is 5° (0.0873 rad).



(a) Crab Angle Step Response

(b) LSA Command

Figure 8.7: Crab angle step response and corresponding LSA command for the first crab angle controller operating on the physical UAV.

The crab angle response struggles to reach the reference value due to the rudder not having sufficient authority to produce the yaw moment required to crab the UAV. The crab angle response also has a significant amount of oscillation due to the imperfect rudder deflection angle commanded. The physical UAV's directional dynamics does not seem to be fully captured in the model, hence the sub-optimal performance. The rise time of the crab angle response is 2.99 seconds, which is slower than the SITL response but is within the requirement for the controller. Even though the UAV cannot maintain the exact crab angle commanded, it can reach close to it fairly quickly, which is sufficient to perform the de-crab manoeuvre. The response is therefore acceptable. The LSA reference that is commanded by the crab angle controller, shown in Figure 8.7b, is within the limits specified for it.

8.3.2 Airfield Waypoint Navigation

Now that the individual controllers' performances have been analysed, their combined performance to navigate the UAV around the airfield can be investigated. Figure 8.8 shows the UAV's practically measured trajectory around the HRF airfield as it follows the waypoints.

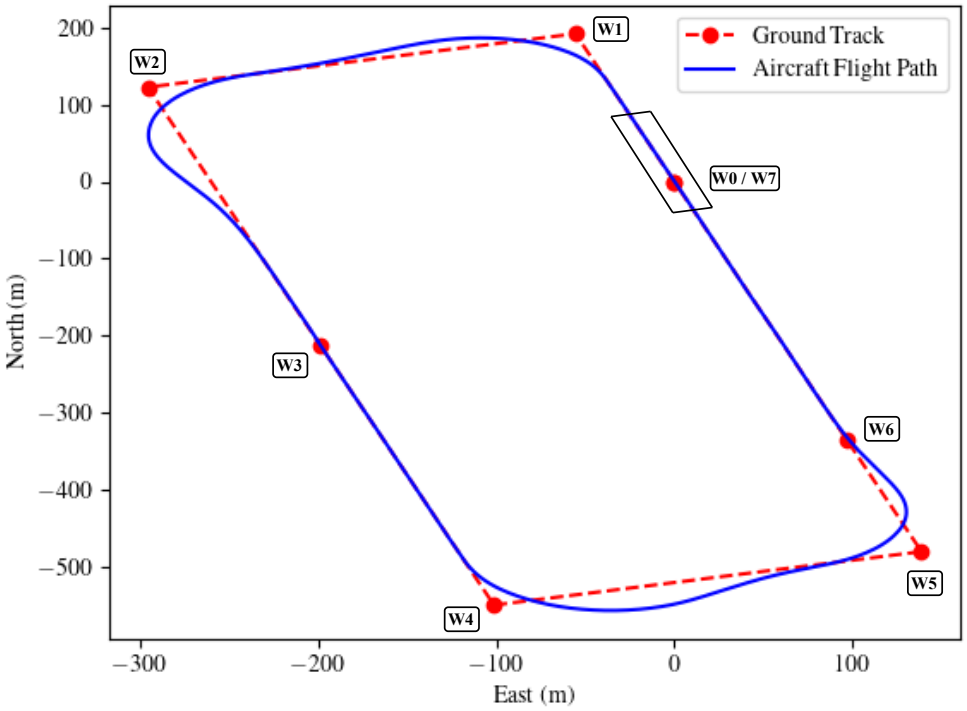
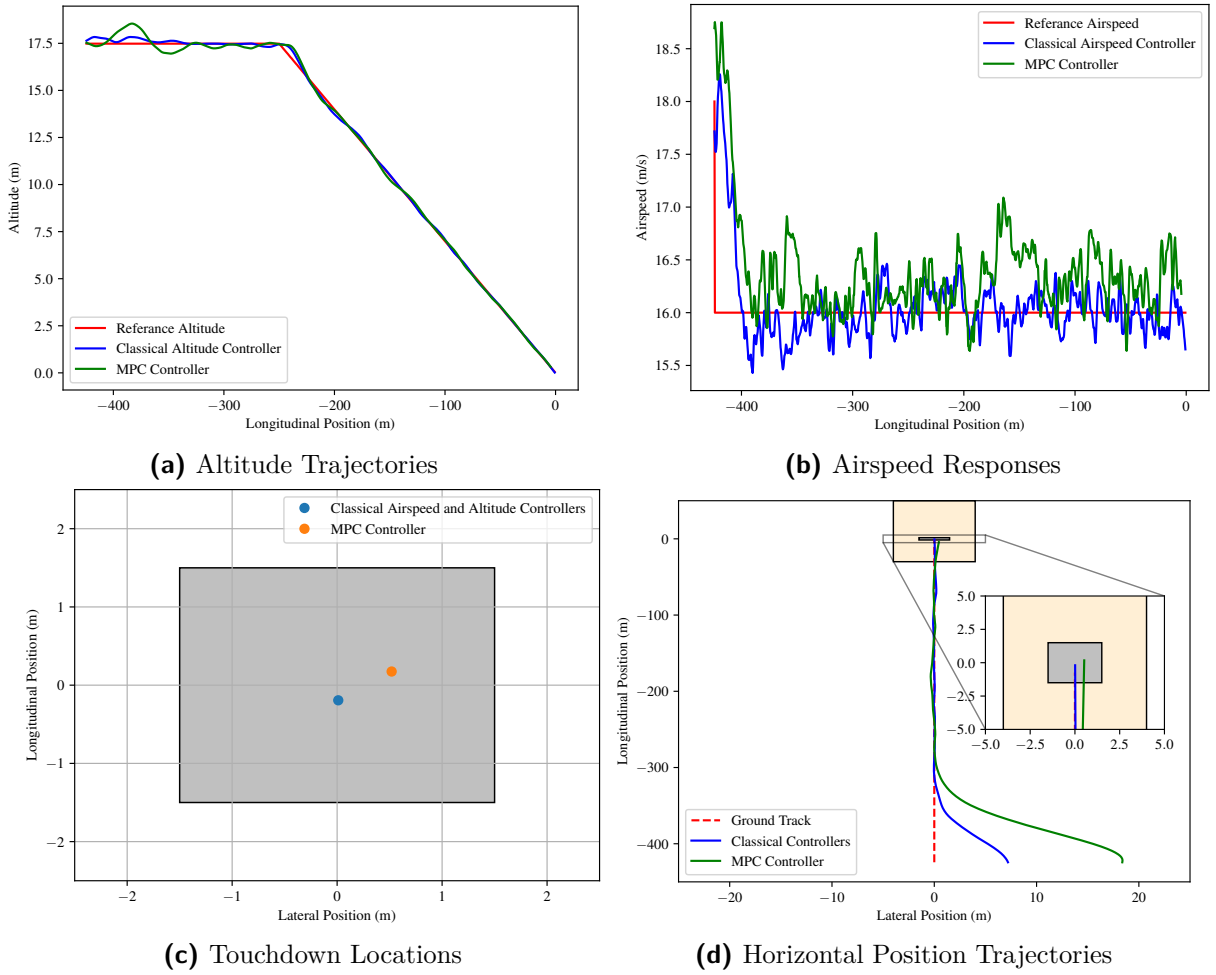


Figure 8.8: Diagram showing the ground track reference and the UAV flight path at the HRF airfield. The waypoints forming the ground track are labeled and the runway diagram is added at waypoint 0 / 7.

The UAV is able to follow the ground track reference. However, it does slightly overshoot at the corner waypoints. This is likely due to the look ahead distance being too small for the practical UAV even though it was sufficient for the simulation. Unfortunately, this behaviour was only discovered after the flight tests were completed and therefore could not be corrected. The overshoot at waypoint 2 and 4 are larger than the other corner waypoints due to the UAV experiencing a tail wind. The tail wind increases the UAV's ground speed causing the UAV to have less time to turn. A solution to the issue is to dynamically select the look ahead distance based on the UAV's ground speed, as suggested by Le Roux [14]. This solution only works if the changes in the wind occur before the turn. Any changes in the wind during the turn will have to be compensated for by the cross-track controllers. Even though the UAV overshoots the ground track, it can still minimise its cross-track error fairly quickly. The UAV has a very small cross-track error when passing over the runway, therefore it should be able to achieve the required lateral landing accuracy on touchdown. The UAV's trajectory around the airfield is therefore acceptable.

8.3.3 Runway Landing

Now that it has been established that the fixed-wing UAV can navigate around the airfield, the runway landing performance of the UAV can be examined. Figure 8.9 shows the practically measured performance of the physical UAV for the practical runway landing, using either the classical airspeed and altitude controllers or the MPC controller.



(a) Altitude Trajectories

(b) Airspeed Responses

(c) Touchdown Locations

(d) Horizontal Position Trajectories

Figure 8.9: The fixed-wing UAV trajectories, airspeed and touchdown locations for a practical runway landing, using either the classical altitude and airspeed controllers or the MPC controller.

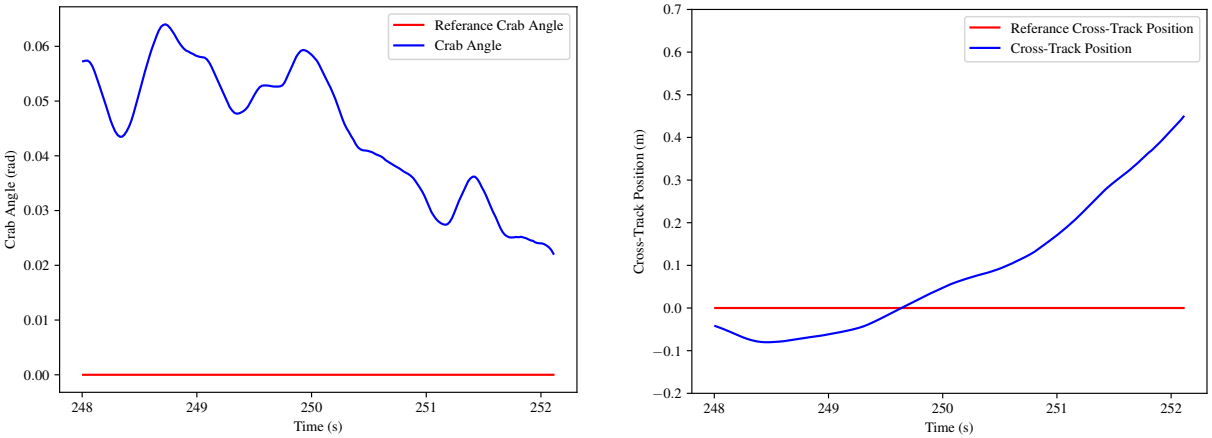
Both controllers were successfully able to track the glide slope and land the UAV on the runway, within the designated 3 m x 3 m landing zone. Similar to the SITL simulation, the UAV is commanded to slow down to 16 m/s for the runway landing. As shown in Figure 8.9b, both the classical airspeed controller and the MPC can slow the aircraft down. The MPC uses the altitude to help slow down the UAV, and this is represented by a small oscillation at the beginning of the MPC's altitude trajectory, shown in Figure 8.9a. This oscillation does diminish before the UAV descends to capture the glide slope. Both airspeed responses contain steady-state variations caused by high-frequency noise and low-frequency disturbances, as previously mentioned during the analysis of the practical

airspeed step responses (Section 8.3.1.1). The variations are acceptable as the airspeed does not reach the stall speed. The variations also do not hinder the UAV from capturing the glide slope. The UAV begins to descend on the glide slope at -250 m from the touchdown point. However, unlike the SITL simulation, it is difficult to observe if there is an increase in airspeed during the descent due to the steady state variations masking the possible increase in airspeed. The classical airspeed controller has better control over the airspeed due to the MPC airspeed response containing a steady-state error. The magnitude of the steady-state error is hard to determine due to the steady-state variations. Nonetheless, both controllers' airspeed responses are acceptable for the runway landing.

Both the classical altitude controller and the MPC can accurately track the glide slope, as shown by their altitude trajectories in Figure 8.9a. However, they do have slightly more oscillation than the corresponding SITL simulation altitude trajectories due to imperfect control surface deflections and wind. The oscillation does diminish for both altitude trajectories as the UAV reaches the touchdown point. As shown in Figure 8.9c, both controllers can land the UAV on the runway within the 1.5 m limits for the in-track and cross-track errors. The classical airspeed and altitude controllers obtained an in-track error of 19 cm and a cross-track error of 1 cm, while the MPC controller obtained an in-track error of 17 cm and a cross-track error of 52 cm. These landing accuracies are slightly worse than the SITL simulation, which was anticipated. The landing accuracies for both controllers are similar, as the MPC only has a slightly better in-track error compared to the classical controllers. The MPC's cross-track error is significantly higher than the classical controllers' cross-track error, which is due to the difference in the environmental conditions during the two landings, as the same cross-track controller is shared between them. The classical controllers and MPC runway landings were performed on different days with different wind conditions due to the MPC requiring readjustment. The UAV experienced a cross-wind during the MPC runway landing, which caused the increased cross-track error.

Figure 8.9d shows the lateral tracking performance of the cross-track controller during both practical runway landings. As was discussed in the airfield waypoint navigation section (Section 8.3.2), the UAV overshoots the corner waypoints due to the selected look-ahead distance being too small. This behaviour was present during both practical runway landings. The cross-track controller was therefore required to recover from the overshoot and return the aircraft to the ground track. The effect of the recovery is shown at the beginning of the horizontal trajectories in Figure 8.9d. The MPC horizontal trajectory has a larger overshoot as the crosswind on landing was a tailwind before turning on the final approach. The tailwind caused an increase in the UAV's ground speed and this provided less time to execute the turn resulting in a larger overshoot. The cross-track controller can successfully recover from the overshoot and minimises the cross-track error to a centimetre level before touchdown for both practical runway landings. In the MPC

practical runway landing just before touchdown, the UAV performs the de-crab manoeuvre to align the UAV with the runway. This increases the UAV's cross-track error, hence the larger cross-track error for the MPC controller touchdown point in Figure 8.9c. The UAV's performance when executing the de-crab manoeuvre is shown in Figure 8.10.



(a) UAV Crab Angle **(b)** UAV Cross-Track Position

Figure 8.10: The fixed-wing UAV's crab angle and cross-track position when executing the de-crab manoeuvre during the MPC practical runway landing.

The plots are shown during the last few seconds that lead to the UAV touching down on the runway. Figure 8.10a shows that just before the UAV performs the de-crab manoeuvre, it has a crab angle of approximately 0.06 rad (3.44°). At touchdown, the crab angle is reduced to approximately 0.02 rad (1.15°) by the crab angle controller. The crab angle controller struggles to change the crab angle of the UAV, which is expected as the practical crab angle step response in Section 8.3.1.6 exhibited similar behaviour. While the crab angle controller could not completely align the UAV with the runway, it could minimise the crab angle fairly quickly, which is what the de-crab manoeuvre requires. Figure 8.10b shows that the cross-track error increases rapidly as the crab angle changes. This is due to the aircraft no longer counteracting the crosswind. At touchdown, the cross-track error is still within the 1.5 m limit, making it acceptable.

The control system performance for the practical runway landing using either the classical controllers or the MPC is acceptable, as the landing accuracies are within the 1.5m in-track and cross-track limits.

Figure 8.11 shows the UAV just before touchdown on the runway for the classical airspeed and altitude controllers. A 3 m by 3 m bounding box was created on the runway to visually observe the UAV's touchdown point on the runway. As shown in the image, the UAV lands within the bounding box.

Figure 8.11: Image showing the UAV landing on the runway within the bounding box when using the classical airspeed and altitude controllers.

8.3.4 Moving Platform Landing

It has now been determined that the developed control systems are capable of accurately landing the physical fixed-wing UAV on the runway. Therefore, the final moving platform landing test can be performed. Unfortunately, it was discovered that the base GPS module on the moving platform was not operating correctly as it produced false measurements. The false measurements were observed in the log and the effect of this could be seen on the physical UAV. During this flight test the UAV began to randomly descend even though the autopilot thought that it was maintaining its altitude. This is very dangerous behaviour which cannot be allowed.

After investigating the complication and contacting the manufacturer, it was identified that the problem was caused by a driver issue between the Drotek GPS module and the Raspberry Pi 4. The Raspberry Pi 4 was not officially supported by the GPS module and therefore it was not guaranteed that these two components would work together. The GPS module operated correctly on a Laptop which proved that the hardware was not faulty. Alternative options to the Raspberry Pi 4 were considered, such as the NVIDIA Jetson Nano, Mecer PC Stick, and Intel NUC. The Jetson Nano could not be used as it would not run the ground station software (QGroundControl). The PC Stick could also not be used as it would not provide enough power to run the telemetry module required for the ground station. The Intel NUC had a steep power requirement with power electronics that would not compactly fit on the RC car, therefore it could not be used. Due to no alternatives being found, the RC car had to be abandoned.

It should be noted that the modified u-blox driver in the PX4 software did not influence the Raspberry Pi's fault, as the u-blox driver was located on the Pixhawk mounted on the UAV. The modified u-blox driver worked correctly when using a Laptop with the GPS base module, as the relative distance between the rover and base GPS modules was accurate at

the centimetre level. As the physical RC car was no longer used, the effectiveness of the modification could not be determined.

An alternative option for the moving platform is to use the 3 m by 3 m trailer that was used by Möller [19] and planned to be used by Le Roux [14]. This trailer would be towed by a full size vehicle on a long straight stretch of road. The GCS operator would then be inside the tow vehicle with the ground station hardware connected to the laptop. This method would have required a vast amount of preparation using a significant amount of time that was not available, and therefore it could not be implemented.

It became evident that a physical moving platform could not be used for the moving platform landing due to the project time constraint and the RC car not functioning correctly. It was still desired to perform the moving platform landing test with the physical UAV, therefore a substitution for the physical moving platform was conceptualised. The substitution consisted of using a virtual car as the moving platform. The virtual car would be simulated in PX4 on the physical UAV. Since the landing position predictor only requires the moving platform's position and velocity, this approach would work. While this approach is not the most ideal solution, it still tests all the control systems used to perform the moving platform landing, which was an objective of this project. The implementation and results of the moving platform landing test using the physical UAV and virtual car is presented in the next subsection.

8.3.5 Virtual Moving Platform Landing

The procedure followed for the virtual moving platform landing is the same as the standard moving platform landing except for the physical RC car. The moving platform's motion is simplistic, as it just needs to move at a constant speed in a straight line. This can be easily simulated using the moving platform model presented in Chapter 4 (Section 4.5). The moving platform model is restated for convenience as,

$$\mathbf{p}_{\text{mp}}(t) = \int_0^t \mathbf{V}_{\text{mp}}(\tau)\, d\tau + \mathbf{p}_{\text{mp}}(0) \tag{8.1}$$

where $\mathbf{p}_{\text{mp}}(t)$ and $\mathbf{V}_{\text{mp}}(t)$ are the instantaneous position and velocity of the virtual car at time t in the runway frame, and $\mathbf{p}_{\text{mp}}(0)$ is the initial position of the virtual car at time $t = 0$ in the runway frame. The instantaneous velocity of the virtual car $\mathbf{V}_{\text{mp}}(t)$ at time t in the runway frame is given as,

$$\dot{x}_{\text{mp}} = V_{\text{mp}} + \eta_{vx} \tag{8.2}$$

$$\dot{y}_{\text{mp}} = \eta_{vy} \tag{8.3}$$

$$\dot{z}_{\text{mp}} = 0 \tag{8.4}$$

where V_{mp} is the nominal speed of the virtual car, and η_{vx} and η_{vy} are in-track and cross-track velocity disturbances, respectively. The nominal speed of the virtual car was set to 3 m/s, to ensure that the fixed-wing UAV would not overshoot the runway when performing the landing. The in-track and cross-track velocity disturbances were set to zero for the virtual moving platform landing to maximise the chances of physical UAV landing on the virtual moving platform. The virtual car's position and velocity are given in the inertial frame to the landing position predictor. The virtual car's position and velocity are converted from the runway frame to the inertial frame using the following transformation matrix,

$$(\mathbf{R}_{\text{r}\to\text{i}}) = \begin{bmatrix} \cos\Psi_{\text{r}} & -\sin\Psi_{\text{r}} & 0 \\ \sin\Psi_{\text{r}} & \cos\Psi_{\text{r}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.5}$$

where Ψ_{r} is the runway heading.

The virtual platform, which is located above the moving platform, is still the touchdown target for the UAV and is therefore still implemented with the virtual car. Figure 8.12 shows the virtual platform and car configuration for the virtual moving platform landing.

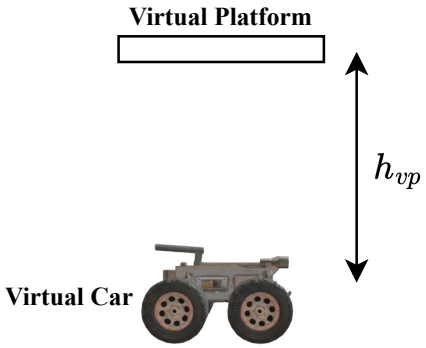
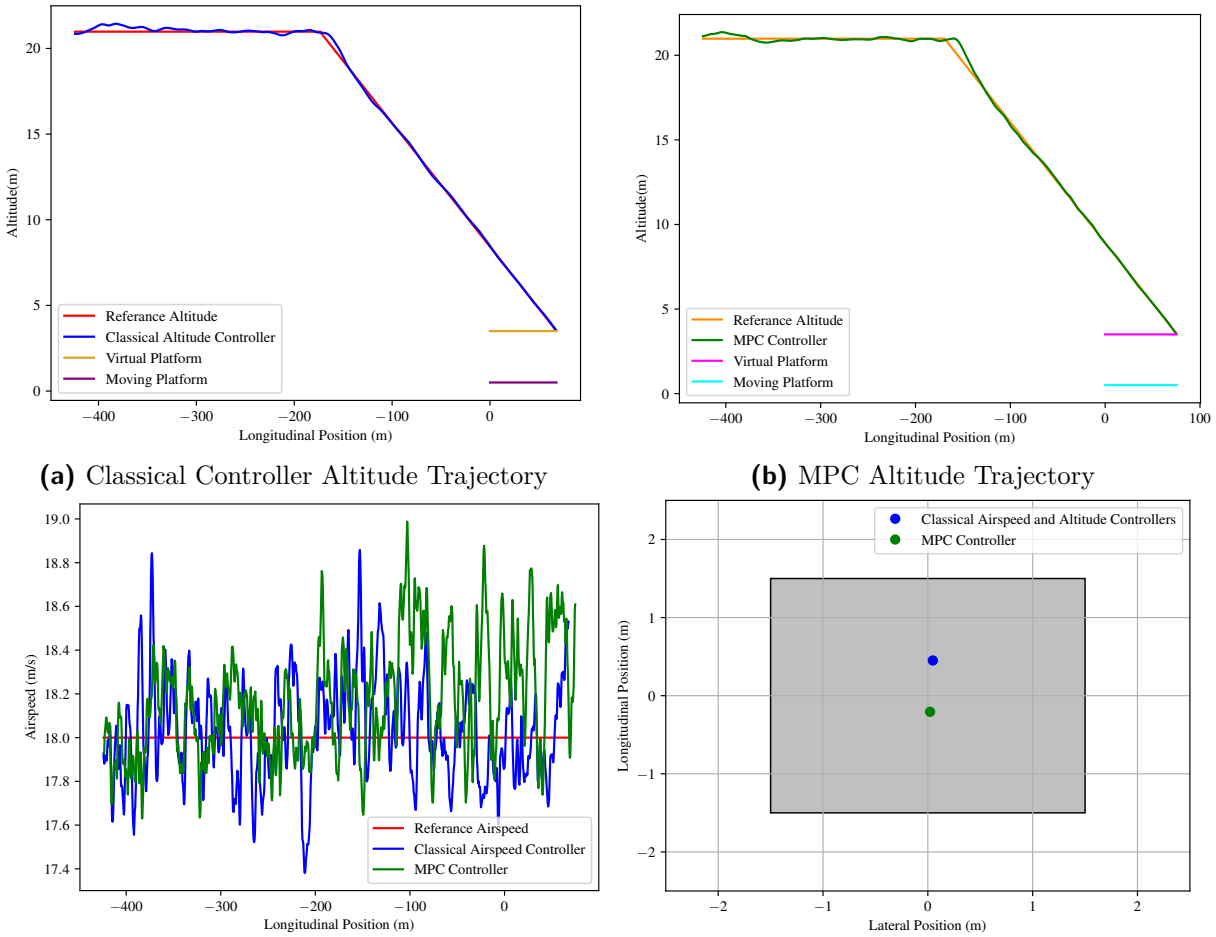


Figure 8.12: Diagram showing the virtual car and the corresponding virtual platform above the car.

The height h_{vp} is kept to the same value of 3 m. Figure 8.13 shows the practically measured performance of the fixed-wing UAV when performing the virtual moving platform landing using either the classical airspeed and altitude controllers or the MPC controller.

The classical controllers and MPC moving platform landings were tested on different days with dissimilar wind conditions, which caused the UAV's ground speed to differ between the two landings. The predicted touchdown points for the two landings therefore varied, which caused their altitude references to be distinct from one another. This is the reason why the two controller altitude trajectories in Figures 8.13a and 8.13b are plotted separately from one another. Both controllers were successfully able to track the glide slope and landed the physical fixed-wing UAV onto the 3 m x 3 m virtual platform.

Similar to the SITL simulation, the UAV is kept at the trim speed for the moving platform landing, therefore there is no initial oscillation in the MPC's altitude trajectory



(a) Classical Controller Altitude Trajectory

(b) MPC Altitude Trajectory

(c) Airspeed Responses

(d) Touchdown Locations

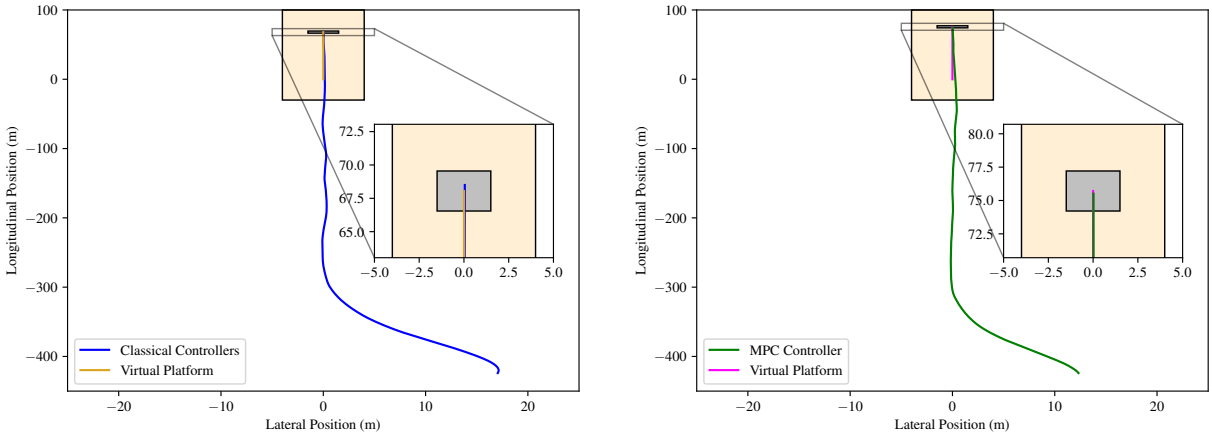
Figure 8.13: The fixed-wing UAV altitude trajectories, airspeed and touchdown locations for a practical virtual moving platform landing, using either the classical altitude and airspeed controllers or the MPC controller.

as shown in Figure 8.13b. Both the airspeed responses of the classical airspeed controller and the MPC, shown in Figure 8.13c, contain steady-state variations, which were also present during the practical runway landings. The variations are acceptable as they do not cause the airspeed to go close to the stall speed and do not hinder the UAV from capturing the glide slope. When the UAV descends on the glide slope, the airspeed of the UAV is expected to increase. However, the steady-state variations mask the increase in airspeed, which makes it difficult to observe the controllers' responses to the sudden increase in airspeed. Once again, the classical airspeed controller performs better than the MPC as the MPC contains a steady-state error. Nonetheless, both controllers' airspeed responses are acceptable for a virtual moving platform landing.

Both the classical altitude controller and the MPC were able to track the glide slope fairly well with minimal oscillation in their altitude trajectories, as shown in Figures 8.13a and 8.13b. Figure 8.13d shows that the MPC controller has a better landing accuracy with an in-track error of 21 cm and a cross-track error of 2 cm compared to the classical airspeed and altitude controllers, which achieve an in-track error of 45 cm and a cross-track

error of 5 cm. It should be noted that during the classical controller's landing, the UAV experienced a tail wind which, as shown in the SITL results, would cause the UAV to overshoot the intended touchdown point. It is quite challenging to obtain landing results for the controllers that are not influenced by external factors when testing in the real world. Testing the two controller landings on the same day with multiple landing attempts would paint a clearer picture of the controllers' landing performance, which was the original plan. This method, however, would require a lot of time and batteries, which were not available when completing the flight tests.

Figure 8.14 shows the lateral tracking performance of the first cross-track controller for the virtual moving platform landing when using either the classical controllers or the MPC.



(a) Classical Controller Horizontal Trajectory

(b) MPC Horizontal Trajectory

Figure 8.14: The aircraft horizontal position trajectories for a virtual moving platform landing on the physical fixed-wing UAV, using either the classical altitude and airspeed controllers or the MPC controller.

For both virtual moving platform landings in Figure 8.14, the UAV overshoots the corner waypoints around the airfield, which was also the case for the practical runway landing. The cross-track controller recovers from the overshoot and returns the UAV to the ground track for both virtual moving platform landings. The cross-track controller successfully minimises the cross-track error of both horizontal trajectories and brings the error down to a centimetre level before touchdown. As a result, the cross-track error at touchdown is within the 1.5 m limit. The cross-track controller's performance is therefore acceptable for the virtual moving platform landing.

The aim for the practical flight tests was to validate that the control system designed could accurately land a physical UAV onto a moving platform. This aim has been achieved by both sets of controllers, as their touchdown points are within the virtual platform's 1.5 m limits.

8.4. Summary

This chapter first discussed the practical flight test logistics. This was followed by a presentation of the practical flight test campaign. Finally, the results from the practical flight tests were then analysed. The analysis of the practical flight test results first considered the individual controller step responses. All the controllers performed adequately on the physical fixed-wing UAV. The classical airspeed controller outperformed the MPC in controlling the physical UAV's airspeed. However, the MPC slightly outperformed the classical altitude controller in controlling the physical UAV's altitude. The control system could navigate the physical UAV around the airfield but with overshoots at corner waypoints. For the practical runway landing, both the classical airspeed and altitude controllers and the MPC could land the physical UAV on the runway. The landing performances for the classical controllers and the MPC were similar, but the MPC had a larger cross-track error due to a crosswind experienced by the UAV on landing. The physical moving platform landing could not be performed due to complications with the RC car hardware. A virtual car was used as a substitute for the physical RC car so that the virtual moving platform landing test could be still performed with the physical fixed-wing UAV. The MPC obtained a better virtual moving platform landing accuracy than the classical airspeed and altitude controllers. However, the physical UAV experienced a tailwind during the classical controllers' landing, reducing its landing accuracy. Based on the results, it can be deduced that the MPC outperformed the classical controllers for the virtual moving platform landing. However, the SITL simulation (Section 7.3.3) showed that the wind has a significant effect on landing accuracy. The aim of the practical flight tests was to verify that developed control systems could land a physical fixed-wing UAV onto a moving platform. This aim has been achieved as the control systems landed the physical fixed-wing UAV within the limits of the virtual platform. The next chapter concludes the thesis.

Chapter 9

Conclusion and Recommendations

9.1. Conclusion

A control system was designed, implemented and practically tested to automatically land a fixed-wing unmanned aerial vehicle onto a moving platform. A non-linear model was created that captured all the relevant flight dynamics of the aircraft and this model was then linearised around a trim point so that it could be used for control system development. A flight control system (FCS) was designed to control the local states of the aircraft. The FCS combines classical control with model predictive control (MPC). The MPC was added to improve the UAV's landing accuracy by having better glide slope tracking. The guidance control system (GCS) was then developed to provide references to the FCS so that the aircraft can follow a trajectory and land on the moving platform. The GCS consists of: the guidance algorithm that provides the guidance variables to the FCS to allow the aircraft to follow a trajectory; the waypoint scheduler which selects the current waypoints being tracked; the landing position predictor which predicts the touchdown point between the aircraft and moving platform; and the state machine that provides the references to the FCS. The control systems' performances were verified in non-linear simulations, first using Simulink, then performing software-in-the-loop (SITL) simulation using Gazebo. The SITL simulation consists of implementing the control systems into PX4 Autopilot software and ROS. Once their performances were deemed acceptable, preparations were made to perform the practical flight tests to verify the control systems' real-world performance. First, the new avionics stack was developed by procuring commercial hardware that was proven to work reliably with fixed-wing UAVs. The software used for the avionics was built on open-source software, as it was proven to work for many different applications. The physical fixed-wing UAV was then assembled by placing new avionics stack into a model RC plane airframe. The individual controllers in the FCS were then practically tested and the practical runway landing scenario was executed. Once these practical results were considered satisfactory, then the physical moving platform was assembled by mounting electronics to an RC car chassis. Unfortunately, the GPS module on the RC car did not function correctly due to a driver issue resulting in the physical moving platform tests being abandoned. It was decided to substitute the physical RC car with a

virtual RC car that would be simulated on the physical UAV. This solution would still allow the control systems on the physical UAV to be practically tested, which was the main aim of the practical tests. The virtual moving platform landing scenario was then successfully executed by the physical UAV. All the objectives listed in Section 1.2 were successfully achieved, except for the creation of the physical moving platform to be used for the practical moving platform landing tests. However, the virtual moving platform substitute was an acceptable replacement as the virtual moving platform landing test still verified that the physical aircraft could successfully land on a virtual moving platform with the required landing accuracy.

All the controllers that were designed performed well in both linear and non-linear simulations, as well as in practice on the physical UAV. Even though the MPC normally replaced the classical airspeed and altitude controllers for landing, the latter were still developed to execute the landing scenarios so that their results could serve as a baseline for the MPC. In linear simulation, the MPC had faster response times in airspeed and altitude compared to the classical controllers. For the non-linear SITL simulation, the MPC had a superior altitude response. However, its airspeed response was slightly worse than the classical airspeed controller due to the MPC's airspeed response containing a steady-state error. This was tolerable as accurate control of the airspeed is not as important as accurate control of the altitude trajectory. Consequently, the MPC had better landing accuracy than the classical controllers for the runway and moving platform landing scenarios in SITL simulations. The MPC achieved a landing accuracy (in-track error) of 6 cm for the SITL simulation runway landing, and a landing accuracy (in-track error) of 10 cm for the SITL simulation moving platform landing. For the practical tests, the MPC once again achieved better altitude control but worse airspeed control compared to the classical controllers. For the practical runway landing, both the MPC and classical controllers achieved similar landing accuracies. However, for the practical virtual moving platform landing, the MPC achieved a better landing accuracy. The MPC achieved a landing accuracy (in-track error) of 17 cm for the practical runway landing, and a landing accuracy (in-track error) of 21 cm for the virtual moving platform landing. When considering the linear and non-linear simulation tests, and the practical flight tests, the MPC outperforms the classical airspeed and altitude controllers, especially in the landing scenarios. It was therefore the correct decision to add the MPC to the FCS. Nonetheless, both the MPC and classical controllers were able to successfully land the fixed-wing UAV onto a platform that moved at 3 m/s (or 10 km/h) within the 1.5 m limit from the intended touchdown point.

This project's landing results can be compared to De Bruin [16] and Le Roux [14] to gauge how well it performed. De Bruin only performed the runway landing scenario, therefore only this scenario's results can be compared. De Bruin achieved a landing accuracy (in-track error) of 15 cm in his practical runway landing which is close to this project's practical runway landing accuracy (in-track error) of 17 cm. It should

be noted that the wind conditions during the De Bruin's and this project's flight test days were different therefore the landing accuracies would be affected. Le Roux did perform the moving platform landing scenario but only in simulation. Le Roux's moving platform landing accuracy (in-track error) was 13 cm on average compared to this project's simulation moving platform landing accuracy (in-track error) of 10 cm. Le Roux's system did not adapt well to the physical UAV as his practical performance contained excessive oscillation. This would have resulted in inadequate glide slope tracking performance and hence unacceptable landing accuracy. In contrast, the practical performance of this project's control system was well adapted to the physical UAV, as it could perform the practical virtual moving platform landing with a high accuracy. Therefore, it can be deduced that this project's control system is an improvement on Le Roux's system.

Since all the research project objectives have been achieved and the moving platform landing accuracy is within 1.5 m, it can be concluded that a control system was designed, implemented, and physically tested to automatically land a fixed-wing UAV onto a moving platform. Both the primary and secondary goals of the research project have been accomplished.

9.2. Research Contributions

The following research contributions were made to the fixed-wing UAV group in the ESL:

- The FCS and GCS designed were able to land a fixed-wing UAV onto a moving platform in simulation and on a virtual moving platform in practice. If in the future it is desired to create a system to land the fixed-wing UAV on physical moving platform, then this project's system would serve as a good base for further development.
- The FCS and GCS are able control the aircraft in stable flight and navigate it around the airfield. These control systems could serve as low-level component for projects investigating high-level applications of fixed-wing UAVs, such as collision avoidance and aerial surveying.
- This is the first project in the ESL to implement a fixed-wing UAV system in PX4 and Gazebo, therefore any future fixed-wing UAV projects that were to use these software packages can use this project as a reference.
- A physical fixed-wing UAV was fully assembled using commercial off the shelf hardware with full support. The UAV was proven to be reliable as it did not have any major issues during the flight tests. This UAV can be used for practical tests in future ESL projects.

- A physical moving platform was assembled and even though its GPS did not function correctly it is complete and mobile. If the GPS issue is fixed in the future, then the RC car could once again be used as a moving platform.
- Obtaining the relative position between the base and rover RTK D-GPS modules is not supported in PX4, therefore custom support was made to utilise this feature. This support can used by future ESL projects that require the feature.

9.3. Recommendations for Future Work

Although the system designed was able to accomplish the goals of the research project, there are aspects of the project that could be improved. These improvements will be suggested for the different components in the project.

Control Systems:

- A larger thrust jig should be used that can measure the full capabilities of the motor. The values in the thrust model should be updated to more accurately represent the thrust.
- The Athena Vortex Lattice (AVL) method used by De Bruin to obtain the aerodynamic coefficients has limitations which require the coefficients to be manually adjusted using practical data. It is recommended that the coefficients be obtained using a different method, so that the aerodynamic characteristics of the aircraft are more accurately represented.
- The MPC performance deteriorates in SITL simulation and on the practical vehicle, compared to its designed performance. This is due to its model dependency, therefore it is recommended to use a more complex form of MPC, such as non-linear or adaptive MPC, to limit the performance deterioration.
- The early switching point in the waypoint scheduler should be dynamic depending on the ground speed of the UAV and not a fixed value as it was in this project. This will ensure that the UAV does not overshoot the ground track no matter the airspeed.
- A controller should be created that can operate directly on the relative data from the GPS, instead of the indirect method that was considered for this project. As the physical moving platform was not used, the indirect method could not be tested.
- The moving platform model could be improved to be more representative of aircraft carrier motions, such as pitching, rolling, and heaving. This will require more complex methods to be developed that can allow the aircraft to track the moving platform.

Software Utilisation:

- Hardware in the loop (HITL) mode is supported in PX4, but not for a fixed-wing model using Gazebo. Custom support should be made to add the functionality, as the HITL simulation will provide results that are closer to the practical data.
- The practical mixer that was created to map the controller commands to the physical actuators was derived using a simple method. A more sophisticated method should be used to derive the mixer, so that the actuator deflections are more accurate on the physical UAV.
- A custom visual interface should be made in QGroundControl (QGC) to quickly change the sub-modes and interact with the aircraft. The current method uses the MAVLink console in QGC which is time consuming and prone to errors.

Hardware Configuration:

- The avionics shelf is mounted with self-tapping screws in the airframe. The PLA material used to form the shelf gets worn down every time the screw is tightened. It is therefore suggested to either 3D print a new shelf or add threaded inserts into the shelf.
- The vibrations produced by the motor cause some nuts in the avionics shelf to loosen. Washers were added to reduce the effect of vibrations. However, this was insufficient. A more elegant solution is required to isolate the shelf from the vibrations.
- A new airspeed sensor mount should be created to prevent the rotational motion present in the current airspeed sensor mount.
- High-precision servos should be used to gain more angle resolution in the actuator deflections.
- The Intel NUC should be considered for the mini computer on the RC car. However, this would require a practical solution to power the NUC on the RC car.
- Additional sensors, such as an IMU, GPS and rotary encoder, could be added to the moving platform. The sensor data could be fused using an EKF to obtain faster updating and more accurate measurements of the platform's state.

Flight Tests Operation:

- The UAV should be landed on the runway at a slower speed (less than 16 m/s) to prevent a prop strike on touchdown. The propeller tips may chip causing vibrations that can influence sensor measurements.
- The practical moving platform flight tests could be performed using the 3 m x 3 m trailer similar to the one that was used by Möller [19] to perform the automated quadcopter landing onto a moving platform. This allows the GCS operator to sit in the tow vehicle with a laptop that can connect to the base GPS module. This would, however, require a significant amount of preparation as Möller's flight tests took place on a naval base.

Bibliography

[1] J. J. Buckley, *Air Power in the Age of Total War*. Bloomington: Indiana University Press, 1999.

[2] S. Mills, *The Dawn of the Drone: From the Back-Room Boys of World War One*. Casemate, 2019.

[3] General Atomics Aeronautical, "MQ-9B-Capability-Profile," 2017, https://www.ga-asi.com/images/products/aircraft_systems/pdf/MQ-9B-Capability-Profile-II.pdf.

[4] Airbus, "Accidents by Flight Phase," https://accidentstats.airbus.com/statistics/accident-by-flight-phase, 2022, Accessed: July 2022.

[5] J. Hayward, "How Do Autoland Systems Work?" https://simpleflying.com/how-do-autoland-systems-work/, August 2022, Accessed: July 2022.

[6] R. Pavithran, V. Lalith, C. Naveen, S. P. Sabari, M. A. Kumar, and V. Hariprasad, "A prototype of Fixed Wing UAV for delivery of Medical Supplies," *IOP Conference Series: Materials Science and Engineering*, vol. 995, no. 1, p. 012015, nov 2020. [Online]. Available: https://dx.doi.org/10.1088/1757-899X/995/1/012015

[7] H. Aetnaean, "What is the aircraft loss-rate from Nimitz-class aircraft carriers?" https://aviation.stackexchange.com/questions/34175/what-is-the-aircraft-loss-rate-from-nimitz-class-aircraft-carriers, December 2016, Accessed: July 2022.

[8] I. K. Peddle, "Autonomous Flight of a Model Aircraft," Master's thesis, Stellenbosch University, 2005.

[9] I. K. Peddle, "Acceleration Based Manoeuvre Flight Control System for Unmanned Aerial Vehicles," Ph.D. dissertation, Stellenbosch University, 2008.

[10] J.-C. Roos, "Autonomous Take-Off and Landing of a Fixed Wing Unmanned Aerial Vehicle," Master's thesis, Stellenbosch University, 2007.

[11] R. D. de Hart, "Advanced Take-off and Flight Control Algorithms for Fixed Wing Unmanned Aerial Vehicles," Master's thesis, Stellenbosch University, 2010.

[12] F. N. Alberts, "Accurate Autonomous Landing of a Fixed-Wing Unmanned Aerial Vehicle," Master's thesis, Stellenbosch University, 2012.

[13] S. J. A. Smit, "Autonomous Landing of a Fixed-Wing Unmanned Aerial Vehicle using Differential GPS," Master's thesis, Stellenbosch University, 2013.

[14] C. T. Le Roux, "Autonomous Landing of a Fixed-Wing Unmanned Aerial Vehicle onto a Moving Platform," Master's thesis, Stellenbosch University, 2016.

[15] G. L. Hugo, "Autonomous Landing of a Fixed-Wing Unmanned Aircraft With Partial Wing and Stabiliser Losses," Master's thesis, Stellenbosch University, 2017.

[16] A. de Bruin, "Accurate Autonomous Landing of a Fixed-Wing Unmanned Aircraft under Crosswind Conditions," Master's thesis, Stellenbosch University, 2017.

[17] A. M. de Jager, "The design and implementation of vision-based autonomous rotorcraft landing," Master's thesis, Stellenbosch University, 2011.

[18] A. D. Swart, "Monocular Vision Assisted Autonomous Landing of a Helicopter on a Moving Deck," Master's thesis, Stellenbosch University, 2013.

[19] P. D. S. Möller, "Automated Landing of a Quadrotor Unmanned Aerial Vehicle on a Translating Platform," Master's thesis, Stellenbosch University, 2015.

[20] C. K. Fourie, "The Autonomous Landing of an Unmanned Helicopter on a Moving Platform," Master's thesis, Stellenbosch University, 2015.

[21] P. G. Ioppo, "The Design, Modelling and Control of an Autonomous Tethered Multirotor UAV," Master's thesis, Stellenbosch University, 2017.

[22] J. T. Mfiri, "Autonomous Landing of a Tethered Multi-Rotor Unmanned Aerial Vehicle on a Stationary Platform," Master's thesis, Stellenbosch University, 2019.

[23] P.R. Grobler, "Automated Recharging and Vision-Based Improved Localisation for a Quadrotor UAV," Master's thesis, Stellenbosch University, 2021.

[24] J. Moring, "Taking Off and Landing on an Aircraft Carrier," https://illumin.usc.edu/taking-off-and-landing-on-an-aircraft-carrier/, September 2005, Accessed: August 2022.

[25] S. Fritz, "How do naval aviators land on a carrier in dense fog when they cannot see the Fresnel Lens Optical Landing System?" https://www.quora.com/How-do-naval-aviators-land-on-a-carrier-in-dense-fog-when-they-cannot-see-the-Fresnel-Lens-Optical-Landing-System, February 2019, Accessed: August 2022.

[26] T. Harris, "How Aircraft Carriers Work," https://science.howstuffworks.com/aircraft-carrier4.htm, August 2002, Accessed: August 2022.

[27] L. Chien, "Do airplanes land on moving aircraft carriers, or does the ship stop for airplanes to land?" https://www.quora.com/Do-airplanes-land-on-moving-aircraft-carriers-or-does-the-ship-stop-for-airplanes-to-land, March 2019, Accessed: August 2022.

[28] B. J. Visser, "Die presisie landing van 'n onbemande vliegtuig," Master's thesis, Stellenbosch University, 2008.

[29] G. J. Goosen, "Automatic Upset Recovery for Small Fixed-Wing UAVs," Master's thesis, Stellenbosch University, 2018.

[30] J. Jantawong and C. Deelertpaiboon, "Automatic Landing Control Based on GPS for Fixed-Wing Aircraft," in *2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2018, pp. 313–316.

[31] A. Dharmawan, A. Ashari, A. M. Handayani, R. Meirani Utami, and I. M. Tresnayana, "Auto Landing System on A Fixed Wing Unmanned Aerial Vehicle Using Linear Quadratic Approach," in *2019 5th International Conference on Science and Technology (ICST)*, vol. 1, 2019, pp. 1–6.

[32] B. Brukarczyk, D. Nowak, P. Kot, T. Rogalski, and P. Rzucidło, "Fixed Wing Aircraft Automatic Landing with the Use of a Dedicated Ground Sign System," *Aerospace*, vol. 8, p. 167, 06 2021.

[33] M. Laiacker, K. Kondak, M. Schwarzbach, and T. Muskardin, "Vision Aided Automatic Landing System for Fixed Wing UAV," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2971–2976.

[34] N. P. Santos, V. Lobo, and A. Bernardino, "Autoland project: Fixed-wing UAV Landing on a Fast Patrol Boat using Computer Vision," in *OCEANS 2019 MTS/IEEE SEATTLE*, 2019, pp. 1–5.

[35] J. Wang, W. Lou, Y. Zhao, and W. Liu, "Fixed-wing UAV Recovery Reliably by Moving Platforms based on Differential Games," in *2019 IEEE International Conference on Unmanned Systems (ICUS)*, 2019, pp. 694–698.

[36] Y. Feng, C. Zhang, S. Baek, S. Rawashdeh, and A. Mohammadi, "Autonomous Landing of a UAV on a Moving Platform Using Model Predictive Control," *Drones*, vol. 2, no. 4, 2018. [Online]. Available: https://www.mdpi.com/2504-446X/2/4/34

[37] L. Persson, "Autonomous and Cooperative Landings Using Model Predictive Control," Licentiate Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2019.

[38] C. A. Amadi, "Design and Implementation of Model Predictive Control on Pixhawk Flight Controller," Master's thesis, Stellenbosch University, 2018.

[39] "Pixhawk 4," https://docs.px4.io/main/en/flight_controller/pixhawk4.html, August 2022, Accessed: August 2022.

[40] "Jetson Nano Developer Kit," https://developer.nvidia.com/embedded/jetson-nano-developer-kit, 2022, Accessed: August 2022.

[41] "Holybro Pixhawk 4 Power Module (PM07)," https://docs.px4.io/main/en/power_module/holybro_pm07_pixhawk4_power_module.html, May 2022, Accessed: August 2022.

[42] "DP0601 RTK GNSS (XL F9P)," https://store-drotek.com/891-1024-rtk-zed-f9p-gnss.html#/106-case-with, 2021, Accessed: August 2022.

[43] "Digital Air Speed Sensor," http://www.holybro.com/product/digital-air-speed-sensor/, 2022, Accessed: August 2022.

[44] "Professional Grade Magnetometer RM3100," https://store-drotek.com/893-professional-grade-magnetometer-rm3100.html, 2021, Accessed: August 2022.

[45] "TBS CROSSFIRE NANO RX - FPV LONG RANGE DRONE RECEIVER," https://www.team-blacksheep.com/products/prod:crossfire_nano_rx, 2022, Accessed: August 2022.

[46] "Radio Telemetry Kit (433 / 915 Mhz - Soon to EOLA)," https://drotek.gitbook.io/additional-devices/telemetry/radio-telemetry-kit-433-915-mhz, 2022, Accessed: August 2022.

[47] J. Baker, "8 open source drone projects," https://opensource.com/article/18/2/drone-projects, February 2018, Accessed: August 2022.

[48] J. M. Louw, "Data-Driven System Identification and Model Predictive Control of a Multirotor with an Unknown Suspended Payload," Master's thesis, Stellenbosch University, 2022.

[49] A. Erasmus, "Stabilization of a Rotary Wing Unmanned Aerial Vehicle with an Unknown Suspended Payload," Master's thesis, Stellenbosch University, 2020.

[50] "MAVLink Developer Guide," https://mavlink.io/en/, Accessed: August 2022.

[51] vooon, "mavros," http://wiki.ros.org/mavros, March 2018, Accessed: August 2022.

[52] *STM32G4 Nucleo-32 board (MB1430)*, STMicroelectronics, 9 2019, rev. 2.

[53] "Raspberry Pi 4 Model B - 4GB," https://www.robotics.org.za/PI4-4GB?search=raspberry%20pi%204, Accessed: August 2022.

[54] D. Leone, "US Navy Aircraft Carriers are so fast they can outrun virtually every surface ship and submarine in most sea states," https://theaviationgeekclub.com/us-navy-aircraft-carriers-are-so-fast-they-can-outrun-virtually-every-surface-ship-and-submarine-in-most-sea-states/, January 2022, Accessed: August 2022.

[55] M. V. Cook, *Flight Dynamics Principles*. Elsevier Butterworth-Heinemann, 1997.

[56] B. Etkin and L. D. Reid, *Dynamics of Flight, Stability and Control*, 3rd ed. John Wiley & Sons, 1996.

[57] J. H. Blakelock, *Automatic Control of Aircraft and Missiles*, 2nd ed. John Wiley & Sons, 1991.

[58] FoxFX, "Cessna 152 Plane," https://sketchfab.com/3d-models/cessna-152-plane-5c6849f1458f45398e93e1a7887d688e, 2020, Accessed: August 2022.

[59] *Flying Qualities of Piloted Aircraft MIL-F-8785C*, U.S. Department of Defence, November 1980.

[60] *Flying Qualities of Piloted Aircraft MIL-HDBK-1797*, U.S. Department of Defence, December 1997.

[61] D. G. Hull, *Fundamentals of Airplane Flight Mechanics*, 1st ed. Springer, 2007.

[62] L. Wang, *Model Predictive Control System Design and Implementation Using MATLAB®*, 1st ed. Springer, 2009.

[63] E. F. Camacho and C. Bordons, *Model Predictive Control*, 2nd ed. Springer, 2007.

[64] J. A. Rossiter, *Model-Based Predictive Control: A Practical Approach*, 1st ed. CRC Press, 2004.

[65] L. Dai, Y. Xia, M. Fu, and M. S. Mahmoud, "Discrete-time model predictive control," in *Advances in Discrete Time Systems*, M. S. Mahmoud, Ed. Rijeka: IntechOpen, 2012, ch. 4. [Online]. Available: https://doi.org/10.5772/51122

[66] J. Mattingley, Y. Wang, and S. Boyd, "Receding Horizon Control: Automatic Generation of High-Speed Solvers," *IEEE Control Systems Magazine*, p. 52–65, June 2011.

[67] Mathworks, "White paper: Three ways to speed up model predictive controllers," MathWorks, White Paper, 2021.

[68] S. Lee, "Ghostbusters: Afterlife - RTV - RC Car," https://sketchfab.com/3d-models/ghostbusters-afterlife-rtv-rc-car-632609ac302447c3ab197c45f88a334b, 2022, Accessed: October 2022.

[69] "ROS - Robot Operating System," https://www.ros.org/, Open Robotics, 2021, Accessed: October 2022.

[70] "Using the ECL EKF," https://docs.px4.io/main/en/advanced_config/tuning_the_ecl_ekf.html, 2022, Accessed: October 2022.

[71] "Mixing and Actuators," https://docs.px4.io/v1.12/en/concept/mixing.html, June 2021, Accessed: October 2022.

[72] "Google Maps," https://www.google.com/maps/@-34.0471565,18.7407475,1101m/data=!3m1!1e3, Google, Accessed: October 2022.

[73] Sanghyuk Park, "Avionics and Control System Development for Mid-Air Rendezvous of Two Unmanned Aerial Vehicles," Ph.D. dissertation, Massachusetts Institute of Technology, 2004.

[74] P. Carpenter, "RC Airplane World flight school - lesson #4 : weight and balance," https://www.rc-airplane-world.com/rc-airplane-weight-and-balance.html, 2022, Accessed: November 2022.

Appendix A

Aircraft Specifications

This appendix provides the specifications of the fixed-wing UAV used in this research project. These specification were used to form the model of the UAV and to design the control systems.

A.1. General Specifications

For the practical flight tests, the fixed-wing UAV will be flown at an airfield that is close to sea level. The air density can therefore be approximated as,

$$\rho = 1.255 \text{ kg/m}^3 \tag{A.1}$$

This value is the density of air density at sea level at 15°C.

It is assumed that the earth's gravitational field is uniform so that gravitational acceleration can be constant. The gravitational acceleration is assumed to be,

$$g = 9.81 \text{ m/s}^2 \tag{A.2}$$

To measure the mass of the fixed-wing UAV, the UAV was first fully assembled with all the components it would use during the practical flight tests. The UAV was then placed on an accurate digital scale and the mass value was recorded. The mass of the fixed-wing UAV was found to be,

$$m = 5.885 \text{ kg} \tag{A.3}$$

The fixed-wing UAV's moment of inertia were found using the double pendulum method. This method was used by many previous ESL students to find the moment of inertia for their fixed-wing UAVs. The method consists of having the aircraft suspended by two strings of equal length that are parallel to the moment of inertia axis that is of interest. The aircraft is then deviated by a small angle from its rest position, and then its period of oscillation is timed [8]. The period of oscillation is related to the moment of inertia by the following equation [73],

$$I = \frac{mgd^2}{4\pi^2 l}T^2 \tag{A.4}$$

where T is the period of oscillation, d is the distance between each string and the moment of inertia axis of interest, and l is the length of each string. This equation is only used to calculate the principle moments of inertia (I_{xx}, I_{yy} and I_{zz}) of the fixed-wing UAV. The products of inertia are set to zero as the aircraft is symmetrical about the XZ-plane and the I_{xz} moment of inertia is small. The fixed-wing UAV's moment of inertia is therefore,

$$\boldsymbol{I} = \begin{bmatrix} 0.486602 & 0 & 0 \\ 0 & 0.47552 & 0 \\ 0 & 0 & 0.86461 \end{bmatrix} \text{kg}\cdot\text{m}^2 \tag{A.5}$$

Figure A.1 shows the fixed-wing UAV suspended along its X-axis so that the I_{xx} moment of inertia can be calculated.

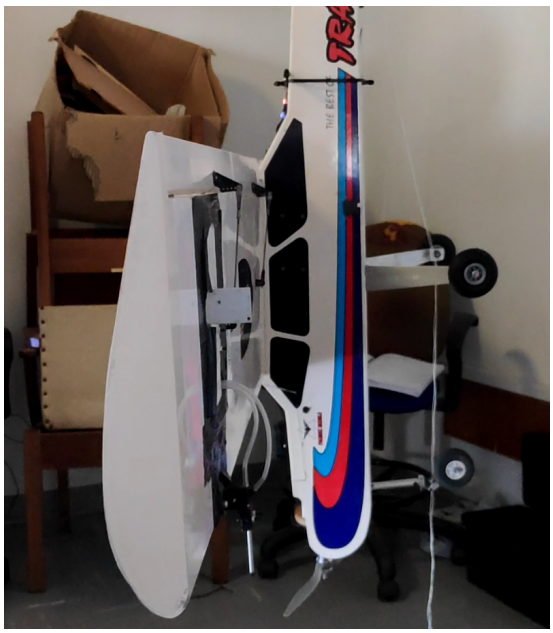


Figure A.1: Image showing the fixed-wing UAV suspended along the X-axis to calculate the I_{xx} moment of inertia.

A.2. Motor Thrust Specifications

The thrust produced by the motor propeller combination needs to be analysed to accurately model the thrust. The RCbenchmark series 1580 thrust jig was used to measure the thrust produced by the motor as it was available to be used in the laboratory. Figure A.2 shows the thrust measurement experiment with the motor connected to the thrust jig.

A barrier was placed between the operator and the motor to ensure safety. Unfortunately, this thrust jig could not measure the full thrust capabilities of the motor as the jig hit its maximum vibration limit before maximum thrust could be produced. No larger thrust jig was available during the test. Therefore, it was decided to reduce the 6S LiPo

Figure A.2: Image showing the setup of the thrust jig.

battery voltage from full charge (25.2 V) to 50% charge (23.1 V). At 50% charge, the maximum thrust could be obtained. Figure A.3 shows the thrust force measured by the thrust jig during the thrust experiment.

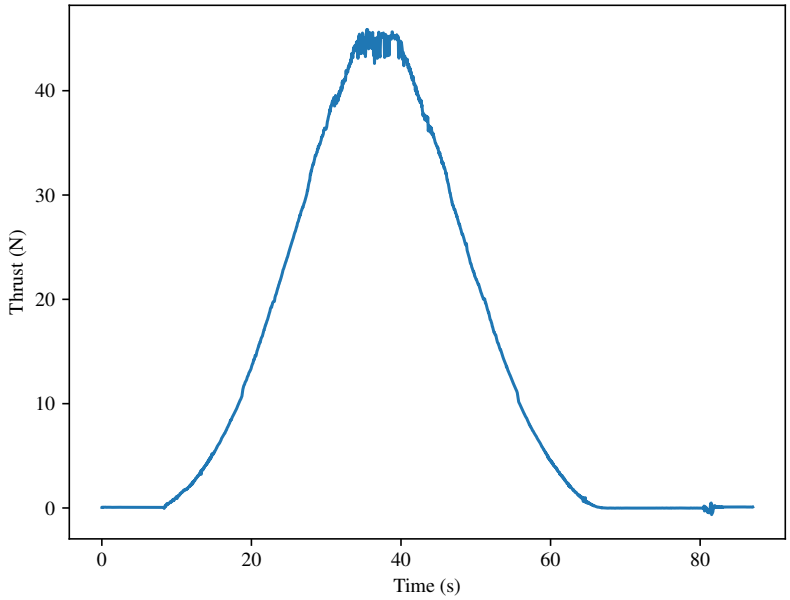


Figure A.3: Plot showing the thrust measured from the motor.

The motor started at no thrust, and then its speed was gradually increased until it produced the maximum thrust. The motor was then gradually slowed down until it stopped. The maximum thrust produced by the motor was 45.86 N. The thrust test took place in a laboratory with no wind and a controlled temperature. In the real world, these factors are not controlled, and other uncertainties exist. Therefore, it was decided to conservatively limit the maximum thrust to 40 N. This would ensure that the motor would always produce enough thrust commanded by the airspeed controllers preventing a stall. The thrust produced by the motor depends on various factors, including the battery voltage and atmospheric conditions that continuously change. Selecting a lower thrust

estimate would be safer for the aircraft. Therefore the thrust range of the motor is,

$$0\,N \leq T \leq 40\,N \tag{A.6}$$

The time delay from when the thrust command was given to the generated thrust was measured to be,

$$\tau_e = 0.25 \text{ s} \tag{A.7}$$

This project uses the same motor and propeller combination as De Bruin [16]. De Bruin obtained the same thrust range and time delay, which gives confidence that the values obtained are correct.

A.3. Airframe Aerodynamic Specifications

The airframe for this project's fixed-wing UAV is the same as the airframe used by De Bruin [16]. This means that De Bruin's aerodynamic values can be used for this project.

A.3.1 Airframe Geometry

Table A.1 shows the airframe geometry values for the fixed-wing UAV.

Table A.1: Airframe geometry values

Meaning	Symbol	Value
Wing Area	S	0.6975
Wing Span	b	1.918
Mean Chord	$\bar{c}$	0.363
Wing Aspect Ratio	A	5.28
Oswald Efficiency Factor	e	0.858

A.3.2 Aerodynamic Coefficients

De Bruin [16] used the Athena Vortex Lattice (AVL) method to obtain the aerodynamic coefficients for his airframe. As this project uses the same airframe as De Bruin, his aerodynamic coefficients can be used for this project. De Bruin found that the AVL method did not completely capture the aerodynamic characteristics of the airframe as he had to manually adjust the parasitic drag coefficient (C_{D_0}) based on practical data. For this research project, the C_{D_0} coefficient was readjusted and it was also required to manually adjust the $C_{L_\alpha}, C_{m_\alpha}$ and C_{m_Q} coefficients due to the practical controller performances being too dissimilar from the simulation. Table A.2 shows the aerodynamic coefficients used in this research project after the coefficients were manually adjusted.

Table A.2: Aerodynamic Coefficients of airframe

Coefficient	Value	Coefficient	Value	Coefficient	Value
C_{L_0}	0.243200	$C_{y_{\delta_R}}$	0.115794	C_{m_Q}	-6.220962
C_{L_α}	3.040906	C_{D_0}	0.180000	$C_{m_{\delta_E}}$	-0.922107
C_{L_Q}	7.046092	C_{l_β}	-0.056602	$C_{m_{\delta_F}}$	0.111822
$C_{L_{\delta_E}}$	0.419064	C_{l_P}	-0.415489	C_{n_β}	0.038208
$C_{L_{\delta_F}}$	0.936323	C_{l_R}	0.127831	C_{n_P}	-0.031465
C_{y_β}	-0.211019	$C_{l_{\delta_A}}$	-0.257631	C_{n_R}	-0.067882
C_{y_P}	0.108287	$C_{l_{\delta_R}}$	0.000920	$C_{n_{\delta_A}}$	0.007213
C_{y_R}	0.150403	C_{m_0}	-0.026700	$C_{n_{\delta_R}}$	-0.049972
$C_{y_{\delta_A}}$	0.000780	C_{m_α}	-0.380993		

A.3.3 Stability and Control Derivatives

The derivation of the stability and control derivatives for the longitudinal and lateral dynamics used in this research project are shown in Tables A.3 and A.4. The derivations are obtained from Alberts [12].

Table A.3: Stability and Control Derivatives for the longitudinal dynamics

Caused By	Lift Forces	Pitch Moments
Angle of attack (α)	$L_\alpha = q_T S C_{L_\alpha}$	$M_\alpha = q_T S \bar{c} C_{m_\alpha}$
Pitch rate (Q)	$L_Q = q_T S \frac{\bar{c}}{2\overline{V}_T} C_{L_Q}$	$M_Q = q_T S \bar{c} \frac{\bar{c}}{2\overline{V}_T} C_{m_Q}$
Flap deflection (δ_F)	$L_{\delta_F} = q_T S C_{L_{\delta_F}}$	$M_{\delta_F} = q_T S \bar{c} C_{m_{\delta_F}}$
Elevator deflection (δ_E)	$L_{\delta_E} = q_T S C_{L_{\delta_E}}$	$M_{\delta_E} = q_T S \bar{c} C_{m_{\delta_E}}$

Table A.4: Stability and Control Derivatives for the lateral dynamics

Caused By	Sideslip Forces	Yaw Moments	Roll Moments
Angle of sideslip (β)	$Y_\beta = q_T S C_{y_\beta}$	$N_\beta = q_T S b C_{n_\beta}$	$L_\beta = q_T S b C_{l_\beta}$
Roll rate (P)	$Y_P = q_T S \frac{b}{2\overline{V}_T} C_{y_P}$	$N_P = q_T S b \frac{b}{2\overline{V}_T} C_{n_P}$	$L_P = q_T S b \frac{b}{2\overline{V}_T} C_{l_P}$
Yaw rate (R)	$Y_R = q_T S \frac{b}{2\overline{V}_T} C_{y_R}$	$N_R = q_T S b \frac{b}{2\overline{V}_T} C_{n_R}$	$L_R = q_T S b \frac{b}{2\overline{V}_T} C_{l_R}$
Rudder deflection (δ_R)	$Y_{\delta_R} = q_T S C_{y_{\delta_R}}$	$N_{\delta_R} = q_T S b C_{n_{\delta_R}}$	$L_{\delta_R} = q_T S b C_{l_{\delta_R}}$
Aileron deflection (δ_A)	$Y_{\delta_A} = q_T S C_{y_{\delta_A}}$	$N_{\delta_A} = q_T S b C_{n_{\delta_A}}$	$L_{\delta_A} = q_T S b C_{l_{\delta_A}}$

Appendix B

Additional Development and Derivations

This appendix presents the additional development of components that was performed for this research project. Additional derivations for this thesis are also presented in this appendix.

B.1. Software Safety Layers

The PX4 autopilot software already contains built-in software protections for the UAV. However, as a custom module was created for the UAV with new control systems, it was necessary to implement additional software safety layers to protect the fixed-wing UAV from errors. Some of the safety layers developed are as follows:

- When the RC transmitter signal is lost, the UAV will switch to or remain in autopilot mode. The UAV will then continuously circuit around the airfield until a command is given.
- For the moving platform landing, the state machine will continuously check if the moving platform is in a safe position for landing. Otherwise, the landing attempt is aborted.
- When the aircraft is switched between autopilot to manual mode or vice versa, the sub-modes are deactivated to prevent the UAV from behaving abnormally when the autopilot is reengaged.
- When communication between the Pixhawk 4 and the Jetson Nano is lost, the MPC becomes inactive and the classical airspeed and altitude controllers on the Pixhawk 4 become active. The classical airspeed and altitude controllers will take over from the MPC.
- Only one autopilot sub-mode can be activated at a time. This is to prevent the UAV from behaving abnormally due to a combination of commands.
- During the MPC ramp response testing, the altitude floor is set to 15 m. This is to prevent the UAV from descending to a lower altitude than 15 m during the response, protecting it from a possible crash.

Two additional features were developed for the autopilot, namely altitude capture and waypoint capture. These features do not increase the safety of the autopilot but rather increase its functionality. The altitude capture saves the altitude value of the UAV when the autopilot is engaged. The UAV will then maintain this altitude until a command is given. Waypoint capture finds the closest waypoint in the circuit when the autopilots are engaged. The UAV will then join the circuit using this waypoint.

B.2. RTK GPS Operation

This section discuss the RTK GPS configuration and the custom Ublox driver that was created to allow the GPS configuration to work in the PX4 software.

B.2.1 RTK GPS Configuration

The standard configuration for using real-time kinematic (RTK) GPS is to have a rover module on the moving unit, e.g. a person, vehicle, or UAV, and a stationary base module connected to the ground station. The base module will first survey its global position using satellite data to a certain accuracy limit, which in QGC is set to 2 m. The base module data is then sent via telemetry radio to the rover module which will use this data combined with satellite data to obtain the rover's position at centimeter level accuracy. This high precision position accuracy is only with respect to the base module and therefore it is called differential GPS (DGPS). In this research project, this configuration is used for testing the individual controllers and for performing the runway landing scenario.

The standard RTK GPS configuration cannot be used for the moving platform landing as the platform imitates an aircraft carrier in the ocean with no stationary component. The RTK GPS should therefore be operated in a mode where the both base station module and the rover unit module are allowed to move while still proving relative position accuracy to within centimeter level accuracy. The Drotek RTK DGPS module was chosen for this project because it contains a u-blox ZED-F9P chip that supports moving base station mode. In this mode both the rover and base modules can move while the relative position and heading between these modules are accurately determined. Unfortunately, the PX4 Autopilot software stack does not fully support this mode, and only provides the relative heading and not the relative position. Custom support was therefore added by modifying the u-blox driver in the PX4 software to also provide the relative position. This modification in PX4 autopilot is experimental as there is no documented attempt of someone using this method online. The modification in the u-blox driver is discussed in more detail in Appendix B.2.2.

Since the relative position between the UAV and the moving platform is required, the DGPS base module must be located on the moving platform. To transmit the RTK packets

from the base module to the rover module on the UAV, two different configurations are considered:

1. the DGPS base module and the ground control station hardware can both be physically located on the moving platform,

2. the DGPS base module can be physically located on the moving platform, and can transmit the RTK packets wirelessly to stationary ground station,

3. the DGPS base station module can be physically located on the moving platform and can transmit the RTK packets wirelessly directly to the DGPS rover module on the UAV, without using the ground control station as an intermediary.

The first option was chosen for this research project as it more closely represents a typical real-world scenario, such as a fixed-wing UAV landing on an aircraft carrier in the ocean. The second option would introduce an additional transmission delay which could impact the relative position accuracy. The third option would require a dedicated communication link between then moving platform and the UAV to be added.

B.2.2 Ublox RTK GPS PX4 Driver Modification

In the PX4 software, the ublox GPS driver is located in the "PX4_devices" submodule in the "ubx.cpp" file. The relative position between the base and rover modules is calculated by the rover module on the UAV. The relative position is then outputted by the rover module via the "UBX_ID_NAV_RELPOSNED" message. This message is read by the "ubx.cpp" file in the PX4 software however only the relative heading between the two modules is extracted. It was therefore required to manually extract the relative position from the "UBX_ID_NAV_RELPOSNED" message.

For the rover module to operate correctly for the chosen RTK configuration it is required that the "GPS_UBX_MODE" PX4 parameter be set to the default value. However, in the default mode, the "UBX_ID_NAV_RELPOSNED" message is not enabled. Therefore, the message was forced to be enabled in the default mode.

The relative position data needs to be sent to the guidance control system, which is located in a custom PX4 module. This process is done with a uorb topic which allows data to be sent between PX4 programs. Unfortunately, this could not be done directly from the "ubx.cpp" file as the ublox driver is a class. Instead, the ublox class constructor was modified to extract the relative position data from the driver. The "gps.cpp" file creates an object of the ublox driver class, and therefore in this file, the relative position data is sent via the uorb topic to the guidance control system.

B.3. Practical Mixer and Aircraft Weight Distribution

As was mentioned in section 7.2.2, PX4 contains a mixer to convert the FCS control surface deflection and thrust commands to actuator commands. By default, the commands from the FCS does not cause the intended physical actuator movement. For the control surfaces, this inaccuracy is due to the linkage geometry on the UAV while for the motor thrust, the inaccuracy is due to change in battery voltage and atmospheric conditions. The mixer's purpose is to allow the physical actuator to move as commanded and this achieved by adjusting the mixers values (explained in [71]). For the control surfaces, the mixer values are obtained by performing practical experiments with a digital angle gauge. The process for these experiments is to command certain angles then measure the physical deflection with the angle gauge. Average mixer values are then derived from these measurements. The physical defections are approximately at their commanded values which sufficient for the controllers. Unfortunately, no mixer adjustment is created for the thrust command as the change in battery voltage and atmospheric conditions are difficult to compensate with a mixer. The classical airspeed controller has an integrator that compensates for the thrust uncertainties and the MPC's thrust model has been adjusted based on the practical data. This is deemed sufficient to perform practical airspeed control.

For the UAV to be stable in flight its weight distribution needs to be correct. The UAV's centre of mass is required to be in front of its centre of lift and this can be check using the standard weight distribution method for RC airplanes [74]. The centre of mass was found to be too far back and therefore lead sinkers were placed close to the nose of the UAV to correct the centre of mass location. The aircraft was setup to be slightly nose heavy so that it would be stable in flight.

B.4. Trim Variables Equation Derivation

The trim variables equation is derived to obtain the trim values, which are used during the linearisation process of the equations of motion. The trim values are the trim angle of attack, trim elevator deflection and the trim thrust. The derivation continues from the discussion in Section 4.6.1.

During level trim flight the pitch angle is small therefore it can be assumed that,

$$\Theta_T = \alpha_T \tag{B.1}$$

The pitch rate at trim flight is $Q = 0$ and therefore the required trim variables to be calculated are $(\alpha, \delta_E, T)_T$. As the sum of the forces and moments are equal to zero, with

reference to Figure 4.13, the following body axes equations hold,

$$\sum \mathbf{F}_X = -q_T S C_{D_T} cos(\alpha_T) + q_T S C_{L_T} sin(\alpha_T) + T_T - mg\, sin(\Theta_T) = 0 \tag{B.2}$$

$$\sum \mathbf{F}_Z = -q_T S C_{L_T} cos(\alpha_T) - q_T S C_{D_T} sin(\alpha_T) + mg\, cos(\Theta_T) = 0 \tag{B.3}$$

$$\sum \mathbf{M}_Y = q_T S\, \bar{c}\, C_{M_T} = 0 \tag{B.4}$$

These equation are simplified by assuming that the trim angle of attack(and pitch) is small and the lift force is an order of magnitude greater then drag at trim. Applying these assumptions to equations B.3 and B.4 results in,

$$-\, q_T S C_{L_T} + mg = 0 \tag{B.5}$$

$$q_T S\, \bar{c}\, C_{M_T} = 0 \tag{B.6}$$

Substituting the coefficients of lift and pitching moment from equations 4.47 and 4.49 and then making trim angle of attack and trim elevator deflection the subject of the formula gives,

$$\begin{bmatrix} \alpha_T \\ \delta_{E_T} \end{bmatrix} = \begin{bmatrix} C_{L_\alpha} & C_{L_{\delta_E}} \\ C_{m_\alpha} & C_{m_{\delta_E}} \end{bmatrix}^{-1} \begin{bmatrix} \dfrac{mg}{q_T S} - C_{L_0} \\ -C_{m_0} \end{bmatrix} \tag{B.7}$$

Substituting this result into equation B.2 and making trim thrust the subject of the formula gives,

$$T_T = q_T S C_{D_T} cos(\alpha_T) - q_T S C_{L_T} sin(\alpha_T) + mg\, sin(\alpha_T) \tag{B.8}$$

where,

$$C_{D_T} = C_{D_0} + \frac{C_{L_T}^2}{\pi A e} \tag{B.9}$$

B.5. MPC controller

This section presents the MPC thrust model constants derivation and the implementation of the MPC in the ROS node.

B.5.1 MPC Thrust Model Constants

For the MPC's practical airspeed step response test (before the K_T and K_{T_c} constants were added), it was noted that the MPC's response had a steady-state error. This steady-state error was not present in the non-linear simulation airspeed step response before the practical flight tests were performed. The steady-state error was most likely due to the uncertainties that exist in the thrust produced by the motor, and also that the MPC thrust

model was not completely accurate. Adding an integrator in parallel with the MPC thrust command could have potentially solved this problem, as it did for the classical airspeed controller. However, the integrator would have affected the MPC's altitude performance, which was not desired hence the integrator was not added. It was therefore decided to add scaling terms to the thrust lag model to more accurately represent the actual thrust produced by the motor. The thrust lag model was already introduced in the thesis however it is restated for convenience,

$$\dot{T} = -\frac{K_T}{\tau_e}T + \frac{K_{T_c}}{\tau_e}T_c \tag{B.10}$$

where K_T is the thrust magnitude constant and K_{T_c} is the thrust command constant. The thrust commanded by the MPC is around the trim thrust T_T, therefore the thrust model that is augmented into the MPC plant is represented as,

$$\Delta\dot{T} = -\frac{K_T}{\tau_e}\Delta T + \frac{K_{T_c}}{\tau_e}\Delta T_c \tag{B.11}$$

where $\Delta\dot{T}$ is the rate of the thrust magnitude from trim, ΔT is the thrust magnitude from trim, and ΔT_c is the thrust commanded from trim. At steady-state, ideally the $\Delta\dot{T}$ term is zero, therefore,

$$\frac{K_T}{\tau_e}\Delta T = \frac{K_{T_c}}{\tau_e}\Delta T_c \tag{B.12}$$

For convenience the K_T constant was set to 1, therefore the equation could be simplified to,

$$K_{T_c}\Delta T_c = \Delta T \tag{B.13}$$

In both the non-linear simulation and the practical tests, the airspeed response was tested for a ± 2 m/s step from the trim airspeed (18 m/s). For the practical tests, the actual average thrust that was commanded from trim by the MPC for these steps was,

$$\Delta T_c(\text{actual}) = \frac{2.4487 + 1.5513}{2} = 2 \text{ N} \tag{B.14}$$

The thrust command that the MPC expected would be based on the model it uses for the predictions. In the simulation, no thrust uncertainties exist, therefore it can be deduced that the thrust commanded by the MPC in the simulation would be the thrust it expects in its predictions. For non-linear simulations, the expected average thrust that was commanded from trim by the MPC for the ± 2 m/s steps was,

$$\Delta T_c(\text{expected}) = \frac{5.461 + 4.858}{2} = 5.1595 \text{ N} \tag{B.15}$$

The K_{T_c} constant can then be calculated as,

$$K_{T_c} = \frac{\Delta T_c(\text{expected})}{\Delta T_c(\text{actual})} = 2.57975 \tag{B.16}$$

By considering Equation B.13, it can be seen that adding the thrust constants essentially informs the MPC that its thrust command has a greater effect (2.57975 times) on the thrust produced by the motor. Placing the thrust constants into the MPC's plant allows the MPC to know beforehand that it should be more gentle with its thrust command. The MPC should then potentially perform better on the physical UAV for the practical flight tests.

It should be noted that once the thrust constant values were obtained, the MPC model was updated and then the MPC was retuned and retested. The results of the MPC that are shown in the thesis are from the final iteration of the MPC after the thrust constants were added.

B.5.2 MPC implementation Process

This subsection discusses the implementation of the MPC in the ROS node for the SITL simulation and the practical flight tests. The implementation of the MPC in the ROS node for the SITL simulation and the practical flight tests is mostly the same with a few minor differences accounting for the physical vehicle. The code used for the MPC's implementation was based on Amadi's code but was modified to work for a fixed-wing UAV, being implemented in ROS and executing on a Jetson Nano. Figure B.1 shows a simplified overview of the MPC implementation in the ROS node.

The MPC matrices that remain constant in the MPC algorithm are first defined. These matrices are the discrete MPC model ($\boldsymbol{A}_d$, $\boldsymbol{B}_d$ and $\boldsymbol{C}_d$), the constraint matrices ($\boldsymbol{CC}$, $\boldsymbol{dd}$ and $\boldsymbol{d}_{u_{\text{past}}}$), the prediction matrices ($\boldsymbol{P}$ and $\boldsymbol{H}$), and the optimiser matrix ($\boldsymbol{E}$). The derivation for the majority of these matrices is discussed in Section 5.3. The MPC algorithm runs in a loop until the ROS node is stopped. The MPC algorithm loop executes for every sample time of the MPC (T_s). The inputs to the MPC, which are the airspeed and altitude references and states, are processed for the MPC algorithm. The references are processed depending if the reference response is a step or ramp. For a step response, the entire prediction horizon is set to the reference value. For a ramp response, a special calculation is performed, which was discussed in Subsection 5.3.2.4. The states are assigned to appropriate variables.

The MPC internal model states are then updated using the discrete MPC model matrices, the previous states and the previous inputs. The previous states are augmented with the airspeed and altitude states that are provided to the MPC. The dynamic MPC matrices, which depend on the MPC states at execution, are then calculated. These matrices are the optimiser matrix $\boldsymbol{F}$ and the constraint matrix $\boldsymbol{d}$. Hildreth's quadratic

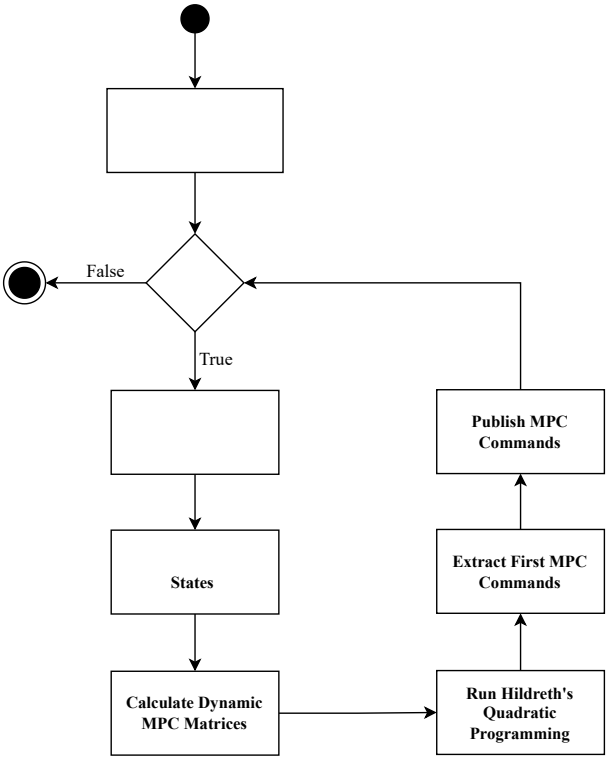


Figure B.1: Simplified overview of the MPC implementation in the ROS node.

programming procedure is then run to obtain the ideal control actions (MPC commands) to apply to the aircraft over the entire control horizon. Only the first set of control actions (MPC commands) are extracted and the remainder is discarded. The MPC commands are the climb rate reference ($\dot{h}_{ref}$) and the thrust command from trim (ΔT_c). These commands are then published and sent to MAVROS, which transmits the commands to the PX4 software via MAVLink.

Appendix C

Checklist Development

Checklists were created for the practical flight tests to ensure that all the required items were completed. The checklists were based on De Bruin's [16] checklists but were extensively altered to account for the new avionics hardware and software. Three documents were created for a flight test day, namely the flight plan, the packing checklist, the setting up checklist and the flight session checklists. The flight plan informs the pilot and the flight crew about the flight tests that are going to take place on the day. The pilot and flight crew are also informed of their duties to be performed for the day. The packing checklist contains the items that need to be done in the last few days leading to the flight test day, to prepare the fixed-wing UAV and the other hardware for the flight test day. The setting up checklist contains hardware components that need to be present and configured for the flight sessions. The flight session checklists contain the items that need to be completed before, during and after the flight session. The flight crew mainly consisted of two groups which were the pilot crew and the ground station crew. The pilot crew consisted of the pilot and his two assistants. The assistants performed the checks on the UAV and placed the UAV on and off the runway. The ground control station crew just consisted of the ground control station operator. The operator monitored the ground station software (QGroundControl) to ensure that the UAV functioned correctly. The operator also changed the autopilot sub-mode to test the different components of the control system or to execute a landing scenario. As two distinct flight crew groups existed, it was required to create two separate flight session checklists. The two checklists contained items that would ensure the two groups would be performing the correct tasks at the designated time. The checklists were improved as the flight tests were performed until the final iteration was derived. The flight test day 6 flight plan, packing and setting up checklists, and single flight session checklist for both crews are shown in the remainder of this appendix.

Flight Plan for Flight Test Day 6

Flight Test Details	
Test Name	MPC Runway + Moving Platform Retest. RC car test
Test Date	19 October 2022
Test Location	Helderberg Radio Flyers Club

I, the undersigned, fully understand my role in the execution and safety of the above flight test:

Team Member Role	Member Name	Signature	Date
Coordinator and GC Operator	Mohamed Zahier Parker		
Safety Pilot	Michael Basson		
Safety Officer and Cameraman	Dr JAA Engelbrecht		
Assistant	Clayton Pheiffer		
Assistant	Merrick Hughes		

Test Objectives

The objectives of the flight test are as follows:

- Retest the MPC for a moving platform land with the virtual car.
- Retest MPC for runway landing.
- Possibly test moving platform landing on actual car if it works correctly.

These objectives will be completed by having two-four flight sessions depending on if the physical car works. The first flight session is used to complete objective one, the second flight session is used for objective two and the third and fourth flight session is used for objective three.

Briefing

Pilot

- Complete check list before every flight.
- Aircraft has not been updated since last time. We should not be required to reprogram it like last time as I left the same values which will force it to land.
- The RC car GPS sensor works now however its accuracy is not great. We will do the flight test as see if it works accurately otherwise, we will stop with that test.
- Please fly further out compared to last time as you saw how far the circuit is. This is to allow waypoint capture to be smooth and not let the autopilot travel to far to get on to the track.
- For the **Land Mode (ALL Flight sessions)** the aircraft must be **armed at the location you want it to land**. I will mention it again before that session.

- The **first session** consists of testing the aircraft's ability to do a **moving platform landing with a virtual car**. When the autopilot is engaged, the aircraft align itself with the runway on final. It will then descend on a glideslope and almost land at the end of the runway. It will stop at a predetermined height (default is 3m) then abort and go around. The aircraft is then programmed to go around and land on the runway as we did previously however I most likely will tell you to take over and land yourself.
  If we go with auto runway land then when the aircraft touches down you must immediately switch to manual mode and bring it to a stop.

In autopilot mode you will not control of any control surface (ailerons, elevator, rudder and flaps) and throttle while in manual mode you control everything. The tasks for this flight session are as follows:

1. **Place the aircraft and arm it on the location you want it to land(for auto land). The aircraft saves the arming location as the landing point.**
2. Take-off with the aircraft from the runway and get the aircraft airborne. Keep a low altitude(20m) as the autopilot altitude is low. Start performing a circuit around the airfield.
3. Once in stable flight and on a long straight switch to **AUTOPILOT MODE** by toggling the mode switch (**SE**) to the centre position on the remote.
4. When moving platform land mode is selected by the GCS operator, he will inform you and then observe the aircraft as it completes the circuit and lines up with the runway on final.
5. Observe the aircraft as it catches the glideslope and comes into almost land on the runway.
6. **IT SHOULD NOT ACTAULLY LAND BUT SHOULD ABORT AT THE SPECIFIED HIEGHT. 3M IS THE DEFAULT.**
7. If the aircraft is not satisfied with its glideslope tracking, then it will abort and go around to try the moving platform landing again. If it fails on a third attempt, then switch to **MANUAL MODE** and land the aircraft on the runway. I will tell you if it aborts.
8. If it is successful, I will tell you. Wait for me to tell you if you should go into manual mode to land it yourself.

!!NB!!: The **second session** consists of testing the aircraft's ability to do the **Runway Landing**. When the autopilot is engaged, the aircraft align itself with the runway on final. It will then descend on a glideslope and **land** at the spot it was armed (beginning) on the runway. **When the aircraft touches down you must immediately switch to manual mode and bring it to a stop.**

In autopilot mode you will not control of any control surface (ailerons, elevator, rudder and flaps) and throttle while in manual mode you control everything. The tasks for this flight session are as follows:

1. **Place the aircraft and arm it on the location you want it to land. The aircraft saves the arming location as the landing point.**
2. Take-off with the aircraft from the runway and get the aircraft airborne. Keep a low altitude(20m) as the autopilot altitude is low. Start performing a circuit around the airfield.
3. Once in stable flight and on a long straight switch to **AUTOPILOT MODE** by toggling the mode switch (**SE**) to the centre position on the remote.
4. When land mode is selected by the GCS operator, he will inform you and then observe the aircraft as it completes the circuit and lines up with the runway on final.
5. Observe the aircraft as it catches the glideslope and comes into land on the runway.
6. **ON TOUCHDOWN IMMEDIALITY GO INTO MANUAL MODE AND BRING IT TO A STOP.**
7. If the aircraft is not satisfied with its glideslope tracking, then it will abort and go around to try the landing again. If it fails on a third attempt, then switch to **MANUAL MODE** and land the aircraft on the runway.

- The **third** and **fourth** flight test procedure is the same as the first one with the only difference being that the plane will track the car instead a specific point on the runway. The almost land point is around the same location as flight session 1 at the end of the runway.
- **MODES**
    1. For the **1st, 3rd and 4th** sessions the plane must start in **MANUAL MODE** then switch to **AUTOPILOT(POSITION) MODE** and finally return to **MANUAL MODE after the GCS operator says to land manually.**
    2. For the **2nd** session the plane must start in **MANUAL MODE** then switch to **AUTOPILOT(POSITION) MODE** and finally return to **MANUAL MODE on touch down.**
- **IF AT ANYTIME YOU FEEL THAT THE AIRCRAFT IS FLYING UNSAFLY THEN YOU CAN RETURN IT TO THE RUNWAY AND LAND OR IF NOT POSSIBLE THEN LAND WHERE IT IS SAFE.**

Pilot Assistant

- Take note of the current task being performed and inform the pilot.
- You are responsible for communication (Walkie Talkie) with the Ground Control Station (GCS) crew and should relay any information between the pilot and GCS crew. This means that you will do the following:
    - Receive instructions from the GCS Crew and relay them to the pilot.
    - Communicate any situation reports from the pilot to the GCS Crew.
- You must carry the checklist and ensure it is completed by the pilot and yourself. Mark the checklist every time an item is completed.
- Execute the Pilot Crew checklists when prompted by the GCS Operator.

Ground Control Station Operator

- Explain the remote controls and the check lists with all the required personnel.
- Monitor QGroundControl(QGC) during the flight and report any anomalies to the pilot.
- Make sure that the aircraft is flight ready at all times and in the “green” zone on QGC.
- Complete the check list before every flight.
- Ensure that the pilot knows what flight task is being performed.
- Give authorisation to start and end the corresponding flight task.
- Communicate with the pilot crew using the walkie talkie.
- For each Flight task confirm what mode the pilot is in during the relevant times.
- Check that the pilot is conforming to the flight task plan by monitoring QGC.
- Tell the pilot any important information they should know.
- During flight session 1 and 2 apply steps for the appropriate controller during the flight.
- During flight session 3 and 4 activate land mode.
- Inform the pilot when the flight step tests are complete.
- You must carry the checklist and ensure it is completed. Mark the checklist every time an item is completed.

Safety officer

- Ensure that all personnel are operating in a safely manner. (**Do not put hands near the propeller!!**).
- If there is an injury attend to them with the first aid kit and if need be, then call an ambulance.
- If there is a fire on the aircraft, then be ready to extinguish it with the fire extinguisher. If the fire is out of control, then call the fire department.

Cameraman

- Responsible for taking the video during the flight task. Wait for the GCS operator's instruction when to start and stop recording.
- Make sure that the video files are saved for each flight test.

Safety briefing (All Personnel)

- Do not touch or place any obstruction near the propeller once the battery is installed into the aircraft and the avionics are powered on.
- Whoever plugs in the battery note that there will a spark due to the ESC capacitors. This is normal and do not be alarmed.
- Nobody should approach the aircraft when it is armed unless it has crashed (be careful though).
- If anyone has an injury, then inform the safety officer so they can provide you with relief with the first aid kit.
- If you spot the UAV on fire, then inform the safety officer and help them with extinguishing it with the fire extinguisher.
- Look out for any other aircraft operating on the same airfield.

Debriefing

Pilot

- How did the aircraft fly?
- Was there any point you felt that it was unsafe?
- What improvements, if any, to the aircraft do you suggest I do to make it operate better?
- Thank you for your time.

Pilot Assistant

- Were all the checklist items completed?
- Was it difficult to follow the checklist and inform the pilot of things?
- Could you clearly communicate with the QCS crew?
- Was the work assigned to you overwhelming?
- Any suggestions for the next flight test?
- Thank you for your time.

GCS Operator

- Were all the checklist items completed?
- Was it difficult to follow the checklist and inform the GCS operator of things?
- Was the work assigned to you overwhelming?
- Any suggestions for the next flight test?
- Thank you for your time.

Safety Officer

- Where there any injuries to anyone due to the operation of the flight test?
- Was there a fire that occurred on the aircraft?
- Have any personnel acted in an unsafe manner?

- Any suggestions to improve safety for the next flight test?
- Thank you for your time.

Cameraman

- Could you easily take a video of the plane for the flight tasks?
- Were there any communication issues between you and the GCS operator?
- Are there any suggestions to improve the video recording of the plane in the future?
- Thank you for your time.

Packing Checklist

Day Before the Flight Test

Items to Charge	
UAV Battery(6S Lipo)	
Back-up UAV Batteries(6S Lipos)	
RC Battery(2s Lion battery)	
Laptop Battery	
RC Car Batteries(2S Lipos)	
RC Car Remote Battery	
Olfi Camera(RC Car Camera) Battery	
Walkie-Talkie Batteries	
UPS Battery	

Physical Check	
Fuselage and wing	
Control surfaces (elevator, rudder, ailerons, and flaps)	
Wheels and propeller	
Servos and main motor	
Telemetry receiver mount	
GPS mount	
RC receiver mount	
Airspeed Sensor Mount	
Airframe weight distribution	
RC Car	

Systems Check	
Load the latest PX4 code onto the Pixhawk	
Boot up PX4 and connect it to QGC	
Calibrate the sensors and other settings if necessary	
Check that all sensors are ready in the QGC status menu	
Connect the RC to the UAV and that the signal is strong by checking RSSI/LQ value	
Calibrate the RC if necessary	
Arm the UAV and test the control surface direction and motor in **manual mode**	
Test that the behaviour is correct for **autopilot(position) mode**	
Check the logging was done correctly and the results is expected	
Check Pixhawk PX4 parameters in QGC	
Car Works	

Other	
Check the predicted weather at the airfield	
Contact the safety pilot	
Attach Propeller onto plane	

Morning of Flight Test

Pack	
UAV airframe + wings	
UAV Battery (6s Lipo)	
UAV Backup Batteries (6s Lipo)	
RC Car	
RC Car Batteries	
RC Car Remote (Tango)	
RC Car Extras bag + RC Camera Extras Case + Extra Opamp	
Lipo Charger	
Telemetry transceiver + Antenna	
Micro USB cable	
Radio Controller (RC) (with accessories)	
RC Battery	
Crossfire RC transmitter	
GPS Base Module	
USB hub	
Laptop	
Laptop Charger	
Laptop Mouse	
Extension lead and multiplug (2 prong adapter as well)	
UPS	
Grey and Black Toolkits	
Fire Extinguisher	
First Aid Med kit	
Wing bolts and bracket	
Walkie Talkies(4 of them)	
Go Pros/Camera	
GPS Tripod	
Table	
Chairs	
Pens and Papers	
Clipboards	
Multimeter	
Double sided tape, Duct tape and insultation tape	
Paper Roll / Tissues	
Chalk	
Telemetry antenna box for table	
Voltage tester	
Router + Lan cables (30m and others)	
Wi-Fi Extender	
UPS	
Checklists	

Setting Up Checklist

Items Present and Airfield	
Fire extinguisher present	
Medkit present	
Aircraft present	
RC Car + Accessories(Remote and extras) present	
Grey and Black Toolkits present	
RC Box present	
Laptop bag and normal bag present	
Extension lead present	
Table, chairs and broom present	
GPS Tripod present	

Setup	
Aircraft Assembled	
Wing nuts tightened	
Table assembled	
Laptop booted up	
GPS Tripod Setup **on the right of laptop to not block telemetry**	
USB hub connected to laptop	
GCS Telemetry sticked to table **in correct orientation with antenna 30° angle from vertical and away from runway**	
GCS Telemetry connected to USB hub	
GCS GPS sticked to table	
GCS GPS connected to Laptop	
Mouse connected to USB hub	
RC setup	
Power connected with lead	
Laptop charger connected	

Pilot Crew Checklists

Before Flight Test

Location: Grass near GCS operator

Pilot Crew

Physical Check (0A)	
The UAV 6S LIPO battery is **DISCONNECTED**	
Fuselage and wing	
Control surfaces (elevator, rudder, ailerons, and flaps)	
Wheels and propeller	
Servos and main motor	
Telemetry receiver mount	
GPS mount	
RC receiver mount	
Airspeed Sensor Mount	
Runway condition + Cleanup(broom)	

Aircraft (0B)	
Check that the UAV 6S LIPO battery voltage is acceptable (more than 24v)	
Open the aircraft's battery cover	
UAV 6S LIPO battery is placed into airframe and is secure.	
Close the aircraft's battery cover	
Airframe weight distribution	
All the **RC switches** are in their **lowest positions** and **throttle stick** is **zeroed** (lowest position)	
Turn on RC if it is not on already (If there are switch/stick warnings provide by the RC then follow them to put the switches/stick in the correct positions)	
Open the aircraft's battery cover	
Connect the UAV 6S LIPO battery	
Check that the Jetson light is on (green)	
Close the aircraft's battery cover(**Make Sure Plastic Screws are fastened all the way)**	
Pixhawk start up beeping sequence heard	
Telemetry transceiver lights up	
RC receiver lights up **Green**	
Beeping sound from main motor	
Confirm with GSC that the Telemetry, GPS, airspeed sensor and RC connections are picked up	
Complete Sensor Calibration Process by following GCS crew's instructions	

Flight Session 1 Checklists

Location: On grass near runway starting area

Pilot Crew

Arming checklist

Aircraft (1A)	
Aircraft is on grass near runway starting position	
UAV is placed in **Manual mode** by moving the MODE (**SE**) RC switch to the position **away from the pilot.** Set throttle to zero.	
Ask GCS crew if UAV is in Manual mode and it is ready to place on runway	
Place Aircraft at desired RUNWAY LANDING LOCATION on Runway	
Arm Vehicle by moving the ARM(**SF**) RC switch to the position **towards the pilot**.	
Test control surfaces direction (optional but should at least be done once on the day)	
Ask GCS crew if UAV is Armed and ready for takeoff.	
Start Flight Session /Takeoff	

Flight Session 1

Disarming checklist

Aircraft (1B)	
UAV on the edge of runway	
Disarm the Vehicle by moving the ARM(**SF**) RC switch to the position **away from the pilot.** Set throttle to zero.	
UAV is placed in **Manual mode** by moving the MODE(**SE**) RC switch to the position **away from the pilot**.	
Ask GCS crew if UAV is Disarmed, in **manual** mode and aircraft can be picked up	
Move aircraft to the grass near runway starting position	
Flight Session Complete	
Open the aircraft's battery cover	
Measure the UAV 6S LIPO battery voltage with the voltage tester while it is still plugged in. • If it is acceptable to continue using, then close the aircraft's battery cover(**Make Sure Plastic Screws are fastened all the way)** and begin next flight session checklist (2). • If it is too low, then begin the battery removal checklist (1C).	
Relay UAV 6S LIPO Battery status to GCS crew	

NB: If battery voltage is acceptable then move to flight session 2 checklist else complete the battery removal (1C) and battery replacement (1D) checklists on the next page.

UAV Battery Removal Checklist flight session 1

Location: On grass near runway starting area

Pilot Crew

Aircraft (1C)	
The aircraft's battery cover is open	
Disconnect Battery	
Check that all GPS, RC receiver and telemetry radio lights are off.	
Power off the RC	
Remove the UAV battery	
Inform GCS crew of battery removal	

UAV Battery Replacement Checklist flight session 1

Location: Grass near GCS operator

Pilot Crew

Aircraft (1D)	
Check that the replacement 6S LIPO battery voltage is acceptable (more than 24v)	
The aircraft's battery cover is open	
6S LIPO battery is placed into airframe and is secure.	
Close the aircraft's battery cover	
Airframe weight distribution	
All the **RC switches** are in their **lowest(away) positions** and **throttle stick** is **zeroed** (lowest position)	
Turn on RC if it is not on already (If there are switch/stick warnings provide by the RC then follow them to put the switches/stick in the correct positions)	
Open the aircraft's battery cover	
Connect the UAV 6S LIPO battery	
Check that the Jetson light is on (green)	
Close the aircraft's battery cover(**Make Sure Plastic Screws are fastened all the way)**	
Pixhawk start up beeping sequence heard	
Telemetry transceiver lights up	
RC receiver lights up **Green**	
Beeping sound from main motor	
Confirm with GSC that the Telemetry, GPS, airspeed sensor and RC connections are picked up	

QCS Operator Checklists

Before Flight Test

Ground Control Station Crew

Ground Control Station (0A)	
Boot up laptop	
Start QGroundControl	
Listen for Pixhawk start-up beep	
Wait for Telemetry signal to be received	
Monitor all sensors and confirm with pilot crew that all sensors and RC are working	
Check that RTK has enough satellites and is accurate enough	
Guide pilot crew with sensor calibration process	
Complete only once at the beginning of the day: • Inform the pilot crew about the RC mode switch positions. • Check power settings. • Check safety settings.	

Flight Session 1 Checklists

Ground Control Station Crew

FIRST SELECT DESIRED OPTIONS FOR CONTROLLER

- Virtual car: **commander fw_cust_opt virtual_car enable**
- Set virtual platform height if desired: **commander fw_cust_opt vp_adj 7.0(altitude value)**
- No mp land decrabing if desired(decrab by default).
- Check setting set status: **commander fw_cust_status options**

Arming checklist

GCS (1A)	
Check that the UAV is in **Manual mode** and Flight Ready	
Tell the pilot crew that the UAV is in **Manual mode,** and it is ready to place on runway	
Tell Cameraman to **Start** Camera recording	
Check that the Vehicle is **armed**	
Tell the pilot crew that the Vehicle is **armed** and ready for takeoff.	
Start Flight Session	
Assist pilot crew during flight	
Move to appropriate flight session checklist	

Flight Session 1

Flight Session Tasks

Virtual Moving Platform MPC Land Mode

- **Virtual car mode is ENABLED: commander fw_cust_status options**
- On first circuit in **downwind leg** enable moving platform land mode:
    - **commander fw_cust_mode_set mpc_mvland enable**
- Observe the aircraft in as it comes to do moving platform landing.
- Check status of state machine after land attempt
    - **commander fw_cust_status sm**
- **Three landing attempts only.**
- If moving platform land succeeded, then tell Michael to land plane manually.

Disarming checklist

GCS (1B)	
Check that the UAV is **Disarmed**	
Check that the UAV is in **Manual mode**	
Tell the pilot crew that the UAV is Disarmed, in **manual** mode and aircraft can be picked up	
Tell Cameraman to **End** Camera Recording	
Ask the pilot crew of the UAV Battery status.	

NB: If battery voltage is acceptable then move to flight session 2 checklist else complete the battery removal (1C) and battery replacement (1D) checklists.

UAV Battery Removal Checklist flight session 1

Ground Control Station Crew

GCS (1C)	
Confirm Battery has been removed	
Disconnect QGroundControl from the UAV	

UAV Battery Replacement Checklist flight session 1

Ground Control Station (1D)	
Listen for Pixhawk start-up beep	
Wait for Telemetry signal to be received	
Monitor all sensors and confirm with pilot crew that all sensors and RC are working	
Check that RTK has enough satellites and is accurate enough	
Tell pilot crew that the Telemetry, GPS, airspeed sensor and RC connections are picked up.	