MULTIPLE SIMULTANEOUS SPECIFICATION
ATTITUDE CONTROL OF A MINI FLYING-WING
UNMANNED AERIAL VEHICLE

by

Shael Markin

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Mechanical and Industrial Engineering
University of Toronto

© Copyright by Shael Markin (2010)
Abstract

Multiple Simultaneous Specification Attitude Control of a Mini Flying-Wing Unmanned Aerial Vehicle

Shael Markin
Master of Applied Science
Graduate Department of Mechanical and Industrial Engineering
University of Toronto
2010

The Multiple Simultaneous Specification controller design method is an elegant means of designing a single controller to satisfy multiple convex closed loop performance specifications. In this thesis, the method is used to design pitch and roll attitude controllers for a Zagi flying-wing unmanned aerial vehicle from Procerus Technologies. A linear model of the aircraft is developed, in which the lateral and longitudinal motions of the aircraft are decoupled. The controllers are designed for this decoupled state space model. Linear simulations are performed in Simulink, and all performance specifications are satisfied by the closed loop system. Nonlinear, hardware-in-the-loop simulations are carried out using the aircraft, on-board computer, and ground station software. Flight tests are also executed to test the performance of the designed controllers. The closed loop aircraft behaviour is generally as expected, however the desired performance specifications are not strictly met in the nonlinear simulations or in the flight tests.
ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor James Mills for providing me with the opportunity to engage in this research as well as for his guidance and support.

I would like to thank Professor Ruben Perez at the Royal Military College of Canada for providing me with the software tools to develop the dynamic model of the Zagi aircraft, and for his advice and guidance throughout the early stages of this project. I would also like to thank Dr. Baoquan Song for the time and effort he spent assisting me in various aspects early on in my research.

I would also like to acknowledge sponsorship for this research from the NSERC Strategic Project Grant, and would like to acknowledge the Koffler Center for providing my colleagues and myself with a flight testing zone on the Koffler Scientific Reserve at Joker’s Hill.

On a personal level, I would like to thank my colleagues, past and present, in the Laboratory for Nonlinear Systems Control, for their friendship, inspiration, and encouragement: Mr. Faysal Ahmed, Mr. Henry Chu, Mr. Joel Rebello, Dr. Lidai Wang, and Dr. Xuping Zhang. I would also like to personally thank my colleagues in the Aircraft Flight Systems and Control Laboratory at UTIAS, Mr. Difu Shi, Mr. Mingfeng (Jason) Zhang, and Mr. Rick Zhang, for their friendship, support, and sincere dedication. Additionally, I would like to thank Professor Hugh Liu for his approachability and constant enthusiasm, even when my colleagues and I returned early from flight tests with damaged aircraft.

I would also like to thank my Examination Committee, Professor William Cleghorn, Professor Hugh Liu, and Professor James Mills, for their time and feedback.

Finally, I would like to express my deepest gratitude to my wife, Rachel, my parents,
and my siblings, for their never-ending support and for always encouraging me to chase after my dreams.
# CONTENTS

Abstract .......................................................................................................................... ii
Acknowledgements ......................................................................................................... iii
List of Figures .................................................................................................................. viii
List of Tables .................................................................................................................. x
Nomenclature .................................................................................................................. xi

## CHAPTER 1 Introduction .................................................................................................. 1
  1.1 Unmanned Aerial Vehicles ......................................................................................... 1
  1.2 Typical UAV Control Structure ............................................................................... 2
  1.3 Aircraft Controller Design: Current Approaches .................................................... 3
  1.4 Introduction to the Zagi UAV Platform ..................................................................... 5
  1.5 Objectives and Contributions .................................................................................. 6
  1.6 Thesis Outline .......................................................................................................... 6

## CHAPTER 2 Aircraft Dynamics ........................................................................................ 8
  2.1 Introduction ............................................................................................................... 8
  2.2 Outline of Aircraft Systems and Notation .................................................................. 8
  2.3 Linear Dynamic Model Development ....................................................................... 11
  2.4 Modal Analysis ......................................................................................................... 18
     2.4.1 Longitudinal Modes ........................................................................................... 19
     2.4.2 Lateral Modes .................................................................................................. 20
  2.5 The Zagi Aircraft Model ......................................................................................... 22
  2.6 Modal Analysis of the Zagi Aircraft ........................................................................ 24
     2.6.1 Longitudinal Modes ......................................................................................... 24
     2.6.2 Lateral Modes ................................................................................................ 26
  2.7 Summary .................................................................................................................. 27

## CHAPTER 3 The Multiple Simultaneous Specification Controller Design Method ............ 28
  3.1 Introduction ............................................................................................................... 28
  3.2 Convex Specifications .............................................................................................. 28
  3.3 Controller Design Framework .................................................................................. 30
  3.4 MSS Controller Design: Problem Definition ......................................................... 31
  3.5 MSS Controller Design Procedure .......................................................................... 32
     3.5.1 Sample Controllers .......................................................................................... 32
     3.5.2 Linear Programming and Convex Combination ............................................... 33
     3.5.3 Extraction of MSS Controller .......................................................................... 35
     3.5.4 Summary of MSS Controller Design Procedure ............................................. 36
  3.6 Stability Analysis ..................................................................................................... 36
  3.7 Observations and Practical Notes on MSS Controller Design in MATLAB ............ 38
  3.8 Summary .................................................................................................................. 39

## CHAPTER 4 Attitude Controller Design and Simulation .................................................. 40
  4.1 Introduction .............................................................................................................. 40
  4.2 Selection of Closed Loop Performance Specifications .............................................. 41
APPENDIX B  Hardware-In-The-Loop Simulation Protocol ........................................ 103
LIST OF FIGURES

Figure 2.1: Aircraft body-axes coordinate system ................................................................. 9
Figure 2.2: Angle of attack and sideslip ............................................................................. 10
Figure 2.3: Phugoid mode oscillations .............................................................................. 20
Figure 2.4: Short period oscillations ................................................................................. 21
Figure 2.5: Dutch roll oscillations ....................................................................................... 22
Figure 2.6: Zagi flying-wing UAV ..................................................................................... 23
Figure 2.7: Zagi aircraft longitudinal modes: pole-zero map ........................................... 25
Figure 2.8: Zagi aircraft lateral modes: pole-zero map ....................................................... 26
Figure 3.1: Geometry of convex functionals ....................................................................... 30
Figure 3.2: Open loop plant framework ........................................................................... 31
Figure 3.3: Closed loop system framework, with \( w \) and \( y \) being of the same dimension .... 32
Figure 4.1: Simulation results for MSS longitudinal controller ......................................... 46
Figure 4.2: Simulation results for MSS lateral controller ................................................... 49
Figure 4.3: Discrete time simulation results for MSS longitudinal controller ..................... 50
Figure 4.4: Discrete time simulation results for MSS lateral controller ............................... 50
Figure 5.1: Kestrel autopilot [29] ....................................................................................... 54
Figure 5.2: UAV platform: aircraft and ground station ....................................................... 55
Figure 5.3: Kestrel autopilot serial ports and sensors [29] .................................................... 56
Figure 5.4: Overo Fire COM [31] ....................................................................................... 57
Figure 5.5: Tobi expansion board [32] ............................................................................. 58
Figure 5.6: Communications block diagram: flight configuration ....................................... 59
Figure 5.7: Communications block diagram: alternative configuration ............................... 59
Figure 5.8: On-board main electronic components ............................................................ 61
Figure 5.9: Flight controller state machine diagram ............................................................ 67
Figure 6.1: Basic operation of HIL simulators ................................................................. 70
Figure 6.2: Virtual Cockpit main window .......................................................................... 72
Figure 6.3: Aviones aircraft simulator .............................................................................. 73
Figure 6.4: Hardware/communication block diagram: normal flight ............................... 76
Figure 6.5: Hardware/communication block diagram: HIL simulation mode ................... 76
Figure 6.6: HIL simulation results - pitch up manoeuver (step applied at time \( t=0' \)) ....... 79
Figure 6.7: HIL simulation results - roll manoeuver (step applied at time t=0') .................80
Figure 7.1: Panoramic view of Joker's Hill flight zone .................................................................82
Figure 7.2: Flight zone and ground station location (courtesy of Google Maps) .................83
Figure 7.3: Pitch-up manoeuver flight test results (step applied at time t=0') .........................86
Figure 7.4: Roll manoeuver flight test results (step applied at time t=0') ...............................87
LIST OF TABLES

Table 2.1: Longitudinal and lateral variables .................................................................14
Table 2.2: Reference flight condition..............................................................................23
Table 2.3: Zagi longitudinal modes...............................................................................25
Table 2.4: Zagi Lateral modes .......................................................................................27
Table 4.1: Performance of longitudinal sample systems .............................................44
Table 4.2: Performance of lateral sample systems .........................................................48
Table 4.3: Continuous and discrete time MSS controller simulation results ..............51
Table 5.1: Aircraft thrust equation parameters ............................................................62
Table 5.2: Throttle mapping summary ...........................................................................64
Table 6.1: Virtual Cockpit main window legend ............................................................72
Table A.1: Longitudinal Stability and Control Dimensional Derivatives .....................101
Table A.2: Lateral Stability and Control Dimensional Derivatives ..............................102
**NOMENCLATURE**

**Aircraft Dynamics Nomenclature**

\( A_{lat} \)  
aircraft lateral state space system matrix

\( A_{lon} \)  
aircraft longitudinal state space system matrix

\( B_{lat} \)  
aircraft lateral state space control matrix

\( B_{lon} \)  
aircraft longitudinal state space control matrix

\( C \)  
center of mass

\( C_m \)  
aircraft non-dimensional pitching moment coefficient

\( C_{m_0} \)  
value of \( C_m \) in trim flight

\( C_{m_\alpha} \)  
derivative of \( C_m \) with respect to angle of attack, \( rad^{-1} \)

\( C_{m_\delta_e} \)  
derivative of \( C_m \) with respect to elevator deflection, \( rad^{-1} \)

\( CX_u, CX_w, CX_q, \ldots \)  
aircraft non-dimensional stability derivatives

\( CX_{\delta_e}, CX_{\delta_r}, \ldots \)  
aircraft non-dimensional control derivatives

\( g \)  
acceleration due to gravity, \( ft/s^2 \)

\( h \)  
height above ground, \( ft \)

\( I_{xx} \)  
mass moment of inertia about \( x \)-axis, \( slug-ft^2 \)

\( I_{yy} \)  
mass moment of inertia about \( y \)-axis, \( slug-ft^2 \)

\( I_{zz} \)  
mass moment of inertia about \( z \)-axis, \( slug-ft^2 \)

\( I_{xz} \)  
mass product of inertia about \( x \) and \( z \)-axes, \( slug-ft^2 \)

\( K_m \)  
electric motor constant, \( ft/s \)

\( L, M, N \)  
external torques about \( x, y, z \) body-frame axes, respectively, \( slug-ft \)
$m$  mass, *slug*

$O.S.$  overshoot, *per cent*

$p, q, r$  angular velocities about $x, y, z$ body-frame axes, respectively, 
  $\text{rad/s}$

$\rho$  air density, *slug/ft}^3$

$S_{\text{prop}}$  circular area covered by propeller rotation, $\text{ft}^2$

$t_s$  settling time, $s$

$T$  temperature, $K$

$T_0$  equilibrium thrust force, $\text{lbf}$

$T_F$  total thrust force, $\text{lbf}$

$u, v, w$  linear velocities along $x, y, z$ body-frame axes, respectively, $\text{ft/s}$

$V, V_\infty$  relative wind, $\text{ft/s}$

$X, Y, Z$  external forces long $x, y, z$ body-frame axes, respectively, $\text{lbf}$

$X_0$  equilibrium force (x-axis component), $\text{lbf}$

$\Delta X$  disturbance force (x-axis component), $\text{lbf}$

$X_{u, X_w, X_q, \ldots}$  aircraft dimensional stability derivatives

$X_{\delta_e, X_{\delta_T}, \ldots}$  aircraft dimensional control derivatives

$\alpha$  angle of attack, $\text{rad}$

$\beta$  angle of sideslip, $\text{rad}$

$\gamma$  angle of climb, $\text{rad}$

$\delta$  generalized control surface deflection, $\text{rad}$

$\delta_e$  elevator deflection angle, $\text{rad}$

$\delta_{e_{\text{trim}}}$  elevator trim deflection angle, $\text{rad}$
\[ \delta_a \quad \text{aileron deflection angle, rad} \]
\[ \delta_r \quad \text{rudder deflection angle, rad} \]
\[ \delta_{\text{right-elevon}} \quad \text{right elevon deflection angle, rad} \]
\[ \delta_{\text{left-elevon}} \quad \text{left elevon deflection angle, rad} \]
\[ \delta_t \quad \text{throttle value} \]
\[ \delta_{\text{trim}} \quad \text{trim throttle value} \]
\[ \phi \quad \text{roll angle, rad} \]
\[ \theta \quad \text{pitch angle, rad} \]
\[ \psi \quad \text{yaw (heading) angle, rad} \]

**Controller Design Nomenclature**

\[ H, H(s) \quad \text{closed loop transfer function matrix} \]
\[ \mathcal{H} \quad \text{the set of all closed loop transfer functions} \]
\[ P(s) = \begin{bmatrix} P_{zw}(s) & P_{zu}(s) \\ P_{yw}(s) & P_{yu}(s) \end{bmatrix} \quad \text{plant transfer function matrix} \]
\[ P_{yu}(s) \quad \text{transfer function matrix from } u \text{ to } y \]
\[ P_{yw}(s) \quad \text{transfer function matrix from } w \text{ to } y \]
\[ P_{zu}(s) \quad \text{transfer function matrix from } u \text{ to } z \]
\[ P_{zw}(s) \quad \text{transfer function matrix from } w \text{ to } z \]
\[ u \quad \text{control signals} \]
\[ w \quad \text{exogenous input signals} \]
\[ y \quad \text{signals available to the controller} \]
\[ z \quad \text{regulated output signals} \]
\( \alpha_i \)  
the \( i^{th} \) closed loop performance specification

\[ \Lambda = [\lambda_1 \ \lambda_2 \ \lambda_3 \ldots \lambda_m]^T \]  
convex combination vector

\[ \psi = [\alpha_1 \ \alpha_2 \ \alpha_3 \ldots \alpha_n]^T \]  
vector of closed loop performance specifications

\( \phi(H) \)  
functional defined on \( H \)

\( \phi_i \)  
the \( i^{th} \) specification value

\( \phi_{ij} \)  
the \( i^{th} \) specification resulting from closed loop system \( j \)

\( \phi = \{ \phi_{ij} \} \)  
the \( n \times m \) matrix whose elements are \( \phi_{ij} \)

**Acronyms**

AI-FCS  
Autonomous Intelligent Flight Control System

COM  
Computer-on-Module

COTS  
Commercial off-the-shelf

DATCOM  
Data Compendium

DOF  
Degree(s) of Freedom

EPP  
Expanded Polypropylene

GPS  
Global Positioning System

IMU  
Inertial Measurement Unit

HIL  
Hardware-in-the-Loop

LiPo  
Lithium Polymer

LQR  
Linear Quadratic Regulator

MSS  
Multiple Simultaneous Specification

PID  
Proportional-Integral-Derivative
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC</td>
<td>Radio-Controlled</td>
</tr>
<tr>
<td>RPV</td>
<td>Remotely Piloted Vehicle</td>
</tr>
<tr>
<td>SAT</td>
<td>Small Angle Theory</td>
</tr>
<tr>
<td>SDT</td>
<td>Small Disturbance Theory</td>
</tr>
<tr>
<td>SIL</td>
<td>Software-in-the-Loop</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Transmission Control Protocol/Internet Protocol</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>USAF</td>
<td>United States Air Force</td>
</tr>
<tr>
<td>UTIAS</td>
<td>University of Toronto Institute for Aerospace Studies</td>
</tr>
<tr>
<td>VC</td>
<td>Virtual Cockpit</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Unmanned Aerial Vehicles

In recent years, unmanned aerial vehicles (UAVs) have become invaluable tools in applications such as aerial surveillance, mapping, reconnaissance, search and rescue, and more dangerous missions where there may be serious threats to human pilots. Many of the tasks required for these missions have been demonstrated as accomplishable without a pilot physically on-board the aircraft. Without the need for an on-board pilot, unmanned aircraft can be designed to be smaller, lighter, more agile, and less expensive than their manned counterparts, since no human support systems are required.

There are, in general, two categories of UAVs: remotely piloted vehicles (RPVs) and fully autonomous UAVs. RPVs are flown by specially trained pilots and crews from a remote base station, while autonomous UAVs intelligently navigate through their flight path with very little human intervention. This research focuses on the latter of the two categories, namely autonomous UAVs.

Following the advent and widespread availability of micro-electro-mechanical sensors (specifically inertial sensors such as accelerometers and gyroscopes), and small, long-range, low power radio communication systems, there has been a strong impetus amongst aerospace and control engineers to build their own autopilot systems and apply their own unique control theories to unmanned aircraft [1]. This is evidenced by the numerous
papers published on the topic of UAV control at a rapid pace.

Recently, a number of commercial off-the-shelf (COTS) small-scale autopilot systems that have become available to the public for civilian use. Examples of such pre-built COTS autopilot systems include the “Piccolo” autopilots from Cloud Cap Technology, the MP series autopilots from MicroPilot, the “Kestrel” autopilot from Procerus Technologies, and others. These COTS autopilots are small and light-weight, and can therefore be used on a variety of small aircraft, including radio-controlled hobby aircraft. Many of these commercially available autopilot systems stemmed from research from aerospace control laboratories in universities and colleges across North America. For example, The Kestrel autopilot and UAV platform under study in this thesis began as an experimental UAV control test bed at Bringham Young University in 2005 [2].

There has been particular focus amongst engineers on the control of so called ‘mini’ UAVs, a loose definition for aircraft with wingspans of approximately 3-5 feet. While both larger [3, 4] and smaller [1] UAVs are also popular areas of research, mini-UAVs are typically less expensive, easier to maintain and repair, and easier to operate logistically than larger aircraft, yet they are larger and therefore can handle difficult outdoor flight environments better than their smaller ‘micro-UAV’ counterparts.

1.2 Typical UAV Control Structure

The architecture of a control system (autopilot) for a UAV is typically multi-layered. In [5], Boskovic et al. describe an “Autonomous Intelligent Flight Control System” (AI-FCS) – a generalization of the control architecture used in most UAV autopilots, although some designs vary slightly. The AI-FCS is composed of four levels, which, from highest to lowest, are: decision making, path planning, trajectory generation, and inner loop control. In the
decision making layer, control decisions relating to overall mission objectives and situational awareness are made. The path planning layer involves generating waypoints for the UAV once the high-level objectives have been determined. The trajectory generation system then fits a feasible, smooth trajectory between waypoints. Finally, the goal of the inner loop is to ensure accurate trajectory following for the UAV.

The main benefit of working with COTS autopilot systems is that the ground-work in the development of the autopilot is complete. An engineer who is working with these systems can often focus control design and development, i.e. software development, at nearly any level in the autopilot’s control system. In this thesis, the focus is on the lowest, inner loop layer.

1.3 Aircraft Controller Design: Current Approaches

In low level aircraft controller design problems, it is often required that the closed loop aircraft system satisfy multiple performance-based specifications. Typically there are two approaches to solving this problem. One approach is trial-and-error based, using methods such as Proportional-Integral-Derivative (PID) controller tuning. PID-based control algorithms are used in many COTS autopilot systems due to their simplicity and low processor and memory requirements [6]. Unfortunately PID tuning is a highly iterative design process and can therefore be time consuming, and may result in a large amount of time being spent on a relatively simple control scheme. Furthermore, it may be difficult to use PID tuning techniques to design a controller to satisfy multiple closed loop performance specifications. The other approach is typically comprised of mathematically complicated, nonlinear controller techniques or concepts such as dynamic inversion [7], neural networks [8], fuzzy logic [9], and others. However, the practical significance of these controllers is
limited, as they are unlikely to be implemented in industry. This is due to the specialized mathematical training that would be required to be provided to engineers in the industry in order to be able to apply these techniques. In [10], Blight et al. note that, for this reason, post-1960 developments in control theory have seen relatively little application in production aircraft designs.

An alternative to these approaches is the Multiple Simultaneous Specification (MSS) controller design method. This method allows an engineer to satisfy multiple closed loop performance specifications using any linear controller design technique. The MSS design method was first introduced by Liu and Mills in the simulated control of a three degree of freedom robotic system [11]. The method takes advantage of the convexity inherent in many performance specifications. For this reason, the MSS controller design method has been referred to interchangeably as the “Convex Combination Method” [11, 12]. It transforms the problem of designing a single controller to satisfy $n$ performance specifications into a simpler, three-stage problem:

1) Develop a maximum of $n$ individual controllers, each of which satisfies at least one performance specification;

2) Use the properties of mathematical convexity to perform a linear combination of the resulting closed loop systems;

3) Extract a single controller from the linear combination of closed loop systems that satisfies all specifications.

The use of the MSS controller design method or the Convex Combination Method for aircraft controller design has been discussed previously in the aircraft control literature. In [13-15], the method is used for the design of longitudinal pitch/speed controllers for a Boeing 747
transport aircraft, and the closed loop system performance is verified through simulations. However, due to the nature of the aircraft under study, flight tests could not be performed. Furthermore, development of lateral controllers is not discussed.

1.4 Introduction to the Zagi UAV Platform

The UAV autopilot platform under study in this thesis consists of the Kestrel autopilot from Procerus Technologies. It is mounted on-board a 48-inch wingspan Zagi XS flying-wing foam aircraft [16]. Details of the aircraft and its associated hardware and software will be discussed in detail in later chapters.

The Kestrel autopilot is fully capable of autonomous control of the Zagi aircraft. All that is required during flight is a selection of desired waypoints for the flight path. The autopilot’s control system is multi-layered, in a manner similar to the AI-FCS described by Boskovic et al. However, the instruction manual for the autopilot discusses at length the steps required to empirically tune the various inner-loop PID controllers. Moreover, the tuning must be carried out while it is in flight. These steps are required because the autopilot is not designed for a specific airframe, rather it is generalized, and its gains must be tuned for the specific aircraft in which it is operating. The method by which the controllers must be tuned (in-flight) is not based on any mathematical procedure, but rather on the subjective visual perception of the user. For instance, various lateral motion PID controllers on the autopilot are tuned via trial-and-error until the aircraft ‘appears’ to make a steady turn [17]. It is therefore imperative to develop an alternative controller for the UAV based on a better mathematical foundation, using a properly-developed dynamic model of the aircraft.

Since the low level controller on the Kestrel autopilot cannot be modified directly, this alternative controller can be implemented in the following manner: an external computer
on-board the aircraft receives the aircraft states from the autopilot over a data stream; the external computer then calculates the system error and executes its own control laws, sending its own calculated control signals back to the autopilot; these new control signals over-write the control signals generated by the autopilot’s own PID controllers, and the control surfaces are deflected as commanded by the external computer.

1.5 Objectives and Contributions

The primary objective of this thesis is to design a pitch/roll attitude controller for Zagi flying-wing UAV using the MSS controller design method and to validate the controller performance with flight tests. To the best of the author’s knowledge, this is the first attempt at low level controller design and flight testing of a Zagi UAV using a controller designed via the MSS controller design method.

There are also a number of secondary objectives of this research that contribute to the achievement of the primary objective. These include:

1) Developing a dynamic model of the Zagi aircraft;

2) Developing an algorithm of the MSS controller design method;

3) Developing a familiarity with the Kestrel autopilot and its associated hardware and software in order to develop a protocol for executing Hardware-in-the-Loop simulations;

4) Developing a familiarity with the external computer and its associated software in order to program the MSS controllers and execute them remotely (from the ground) while the aircraft is in flight.

1.6 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 provides a basic
foundation in aircraft dynamics to a level required for understanding the remainder of the thesis. An analysis of the dynamics of the Zagi aircraft under study is also provided. In Chapter 3, the MSS controller design method is introduced, followed by discussions on the topic of convex specifications, the controller design framework, stability analysis, and other matters. Chapter 4 presents the design and linear simulation results of the longitudinal and lateral attitude MSS controllers. Chapter 5 then discusses the topics of Software and Hardware-in-the-Loop simulation and how they are applied with the Procerus UAV platform. Hardware-in-the-Loop simulation results are then provided for the Zagi UAV. Experimental flight test results are presented in Chapter 7. Finally, Chapter 8 offers concluding remarks as well as recommendations for future work.
CHAPTER 2

AIRCRAFT DYNAMICS

2.1 Introduction

This chapter serves as an introduction to aircraft dynamics at a level of detail suitable for understanding the remainder of the thesis. For a complete discussion on aircraft dynamics, the reader is referred to [18-21] and other texts on the matter.

2.2 Outline of Aircraft Systems and Notation

Prior to the discussion of the dynamic model of the aircraft used in this work, the reader should be familiar with certain aircraft-related terminology and notation. Figure 2.1 outlines the so called ‘body-axes’ co-ordinate system of an aircraft, as well as the forces, torques, velocities, and angular velocities about these axes.

As seen in the figure, in the body-axes coordinate system, the positive $x$-axis points out of the nose of the aircraft from the center of mass ($C$), the positive $y$-axis points out of the right wing of the aircraft, and the positive $z$-axis correspondingly points out of the bottom of the aircraft. This orthogonal body-axes system is fixed to the body of the aircraft as it moves through space.
The longitudinal aircraft system refers to motions along the $x$ and $z$ axes and rotations about the $y$-axis; the lateral system refers to motions along the $y$-axis and rotations about the $x$ and $z$-axes. The parameters $X$, $Y$, and $Z$ represent external forces (either aerodynamic or actuator-applied); $L$, $M$, and $N$ similarly represent external torques (aerodynamic or applied); $u$, $v$, and $w$ represent linear velocities; and $p$, $q$, and $r$ represent roll, pitch, and yaw angular velocities. The $x$-$z$, $x$-$y$, and $y$-$z$ planes can be determined from the figure. The $x$-$z$ plane is henceforth assumed to be the plane of symmetry in order to simplify subsequent linear model development [19]. The roll, pitch, and yaw angles, ($\phi$, $\theta$, and $\psi$, respectively) are used to describe the orientation of the body axes in space relative to another coordinate system, usually an inertial frame fixed to the earth’s surface.
The net velocity, or $V_\infty$ (often referred to as the relative wind) is calculated as the vector sum of velocities $u$, $v$, and $w$, and its magnitude is:

$$V = V_\infty = \sqrt{u^2 + v^2 + w^2} \quad (1)$$

There are two additional angles that are important in discussions of the dynamic model. The first is the angle of attack, $\alpha$, representing the angle between the $x$-axis and the relative wind projected onto the $x$-$z$ plane:

$$\alpha = \tan^{-1}\left(\frac{w}{u}\right) \quad (2)$$

The second important angle is the sideslip angle, $\beta$, representing the angle between the $x$-axis and the relative wind projected onto the $x$-$y$ plane:

$$\beta = \sin^{-1}\left(\frac{v}{V}\right) \quad (3)$$

These two angles are illustrated in the Figure 2.2:

![Figure 2.2: Angle of attack and sideslip](image-url)
2.3 Linear Dynamic Model Development

The general form of the nonlinear equations of motion of any object with six degrees of freedom (DOF) can be derived from first principles. However, since these equations are well known, they are provided as Eq. (4), below without derivation [19].

The first six equations describe the dynamics of the object, while the final three describe its kinematics. Additional terms that are used in these equations are \( I_{xx}, I_{yy}, I_{zz}, \) and \( I_{xz}, \) which represent the moments of inertia of the aircraft about its respective axes and planes; \( g \) represents the force of gravity, and \( m \) is the mass of the aircraft.

\[
\begin{bmatrix}
\dot{u} \\
\dot{v} \\
\dot{w} \\
\dot{\phi} \\
\dot{\theta} \\
\dot{\psi}
\end{bmatrix} = 
\begin{bmatrix}
-g \sin \theta + rv - qw + \frac{X}{m} \\
g \cos \theta \sin \phi - ru + pw + \frac{Y}{m} \\
g \cos \theta \cos \phi + qu - pv + \frac{Z}{m} \\
\frac{1}{I_{xx}} [L + I_{xx} (\dot{r} + pq) + (I_{yy} - I_{zz}) qr] \\
\frac{1}{I_{yy}} [M + I_{xx} (r^2 - p^2) + (I_{zz} - I_{xx}) rp] \\
\frac{1}{I_{zz}} [N + I_{xx} (\dot{p} - qr) + (I_{xx} - I_{yy}) pq]
\end{bmatrix}
\]

For aerial vehicles, these general equations become unique through the way the external force terms (\( X, Y, Z, L, M, N \)) are represented. As mentioned above, these can either be aerodynamic forces such as lift and drag, or they can be applied by actuators such as engines or control-surfaces.

On typical aircraft, there are four main types of control surfaces: engines, ailerons, elevators, and rudders. Some aircraft have additional control surfaces such as air-brakes (or
spoilers), flaps, or multiple sets of ailerons and/or elevators. Other aircraft, such as the flying-wing aircraft discussed in this thesis, may have only an engine and one other set of control surfaces that functions both as elevators and ailerons – appropriately called elevons. The engine and the elevators are used to control the pitch angle and airspeed of the aircraft. The ailerons are the primary control surfaces for controlling roll rate, and the rudder is the primary control surface for controlling yaw angle.

The small disturbance theory (SDT) and small angle theory (SAT) are used to develop a locally linearized aircraft model from the nonlinear equations. In the SDT, each dynamic variable is denoted as its respective reference (equilibrium) value plus a disturbance. For example, the force $X$ has the form $X = X_0 + \Delta X$. For simplicity in the derivations below, when a particular reference value is equal to zero, the prefix ‘$\Delta$’ is dropped on the corresponding disturbance term.

A number of assumptions are made about the reference (equilibrium) flight condition about which linearization occurs. Firstly, the reference flight condition is assumed to be symmetric with zero angular velocity and zero external lateral forces. This greatly simplifies the linearization process, because the following values are consequently equal to zero: $v_0$, $p_0$, $r_0$, $\Phi_0$, and $\Psi_0$. Furthermore, a slightly different coordinate frame is used, referred to as the ‘stability axis’ reference frame. In this reference frame, the $x$-axis is chosen to be pointing in the direction of the relative wind ($V_\infty$), thus $w_0$ and $\alpha$ are equal to zero in equilibrium. As a result, $u_0$ is equal to the reference flight speed and $\theta_0$ is equal to the reference angle of climb. Two further assumptions in this development are that the effects of spinning rotors are ignored and that the wind velocity is zero.

When SDT and SAT are applied to the nonlinear equations above, the following
When all disturbance quantities are set to zero in the above equations, the reference flight condition is obtained (recall that the ‘Δ’ notation has been dropped from some of the disturbance quantities):

\[
\begin{align*}
X_0 + \Delta X - mg(sin \theta_0 + \Delta \theta \cos \theta_0) &= m \Delta \dot{u} \\
Y_0 + \Delta Y + mg \phi \cos \theta_0 &= m(\dot{\nu} + u_0 r) \\
Z_0 + \Delta Z + mg(cos \theta_0 - \Delta \theta \sin \theta_0) &= m(\dot{\omega} - u_0 q) \\
L_0 + \Delta L &= I_{xx} \dot{\phi} - I_{xz} \dot{r} \\
M_0 + \Delta M &= I_{yy} \dot{q} \\
N_0 + \Delta N &= -I_{xz} \dot{p} + I_{zz} \dot{r} \\
\Delta \dot{\theta} &= q \\
\dot{\phi} &= p + r \tan \theta_0 \\
\dot{\psi} &= r \sec \theta_0 \\
p &= \dot{\phi} - \dot{\psi} \sin \theta_0
\end{align*}
\]

When all disturbance quantities are set to zero in the above equations, the reference flight condition is obtained (recall that the ‘Δ’ notation has been dropped from some of the disturbance quantities):

\[
\begin{align*}
X_0 - mg \sin \theta_0 &= 0 \\
Y_0 &= 0 \\
Z_0 + mg \cos \theta_0 &= 0 \\
L_0 = M_0 = N_0 &= 0
\end{align*}
\]

The equilibrium conditions are then substituted back into the previous equations, and after slight rearrangement, we arrive at the following:
Upon close observation of Eq. (7), one will note that the equations can be decoupled into two groups: those equations describing longitudinal motion and those describing lateral motion. The longitudinal and lateral variables are summarized in the following table:

### Table 2.1: Longitudinal and lateral variables

<table>
<thead>
<tr>
<th>Longitudinal Variables</th>
<th>Lateral Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u, w, q, \dot{\theta}$</td>
<td>$v, p, r, \phi, \dot{\psi}$</td>
</tr>
</tbody>
</table>

The next step in the dynamic model development is to appropriately describe the disturbance forces and moments applied to the aircraft. An ideal approach is to model each force as a nonlinear function of the states of the aircraft and their derivatives, as well as the current flight conditions. For example, the disturbance force $\Delta X$ can be described by an equation of the form:

\[
\Delta \dot{u} = \frac{\Delta X}{m} - g \Delta \theta \cos \theta_0
\]

\[
\dot{v} = \frac{\Delta Y}{m} + g \phi \cos \theta_0 - u_0 r
\]

\[
\dot{\omega} = \frac{\Delta Z}{m} - g \Delta \theta \sin \theta_0 + u_0 q
\]

\[
\dot{p} = (I_{xx} I_{zz} - I_{xz}^2)^{-1}(I_{zz} \Delta L + I_{xz} \Delta N)
\]

\[
\dot{q} = \frac{\Delta M}{I_y}
\]

\[
\dot{r} = (I_{xx} I_{zz} - I_{xz}^2)^{-1}(I_{xz} \Delta L + I_{xx} \Delta N)
\]

\[
\Delta \dot{\theta} = q
\]

\[
\dot{\phi} = p + r \tan \theta_0
\]

\[
\dot{\psi} = r \sec \theta_0
\]
As the level of model sophistication increases, the number of independent variables in the functions increases. There is no theoretical limit to the level of accuracy and complexity attainable in the development of a dynamic aircraft model. The limits, rather, are those of practicality: the engineer must keep in mind his or her design goals and choose a level of model sophistication according to the accuracy required and the available computational power. There are a number of disadvantages associated with increasing the level of model sophistication. The first is that very extensive and time consuming dynamic modeling techniques must be employed in order to develop the model in the first place. Second, a large database may be required in order to store all of the data acquired during the tests and a complex, high-dimensional look-up table must be used if the data is to be accessed in real time for control purposes. For this research, only the states and applied control forces (and not their derivatives) have been chosen to be the independent variables. Furthermore, due to the decoupling of the equations, the lateral variables are not considered in the longitudinal disturbance forces and vice versa. For example, we have:

$$\Delta X = f(u, w, q, \theta, \delta)$$  \hspace{1cm} (9)

where $\delta$ represents applied control forces with components along the $x$-axis.

The disturbance forces and torques are then chosen to be linear functions of the aircraft states and applied control forces via Taylor expansion [19]. For example, the disturbance force along the $x$-axis at time, $t$, is:

$$\Delta X(t) = \frac{\partial X}{\partial \Delta u} \Delta u(t) + \frac{\partial X}{\partial w} w(t) + \frac{\partial X}{\partial q} q(t) + \Delta X_c(t)$$  \hspace{1cm} (10)

$$\equiv X_u \Delta u(t) + X_w w(t) + X_q q(t) + \Delta X_c(t)$$

Here, the parameters $X_u, X_w,$ and $X_q$ are referred to as stability derivatives and the
subscript \( c \) indicates the actuator-applied control force (the \( x \)-axis component). In the case of the \( x \)-axis, these actuators include the elevons \( (\delta_e) \) and motor (denoted by throttle, \( \delta_T \)):

\[
\Delta X_c(t) = X_{\delta_e} \Delta \delta_e + X_{\delta_t} \Delta \delta_t
\]  

(11)

where the parameters \( X_{\delta_e} \) and \( X_{\delta_t} \) are referred to as the control derivatives. These stability and control derivatives are constant for a specific flight condition – the reference flight condition about which linearization has taken place. They are a function of the inertial and geometric properties of the aircraft and can be determined using empirical flight testing methods [22, 23], or through software tools such as the US Air Force (USAF) Stability and Control Digital Data Compendium (DATCOM). Again, as with a nonlinear model, a database can be used to store the stability and control derivatives for various flight conditions. Furthermore, other stability derivatives, such as those involving the derivatives of the aircraft states, may be added to the equations for increased accuracy.

For the purposes of this research, only a single flight condition within a much larger possible flight envelope is discussed. The stability and control derivatives are calculated for one flight condition and are assumed to be constant throughout the flight envelope. Although this is a simplified representation, it is used for three reasons. Firstly, developing a full model throughout the flight envelope is a very time consuming process; secondly, this thesis does not investigate gain-scheduling control techniques, which may be required if the dynamic model were to change mid-flight; thirdly, the autopilot computer hardware was not assumed to be powerful enough to be able execute the multi-dimensional look-up tables in real time as would be required.

When the Taylor expansions of each of the forces and torques are substituted into Eq. (7), the decoupled equations can be collected into the common linear state space format,
The state space equations for longitudinal motion are [19]:

\[
\begin{bmatrix}
\Delta \dot{u} \\
\Delta \dot{w} \\
\Delta \dot{q} \\
\Delta \dot{\theta}
\end{bmatrix} = A_{long} \begin{bmatrix}
\Delta u \\
\Delta w \\
\Delta q \\
\Delta \theta
\end{bmatrix} + B_{long} \begin{bmatrix}
\Delta \delta_e \\
\Delta \delta_T
\end{bmatrix}
\]

(12)

where:

\[
A_{long} = \begin{bmatrix}
\frac{X_u}{m} & \frac{X_w}{m} & 0 & -g \cos \theta_0 \\
\frac{Z_u}{m-Z_w} & \frac{Z_w}{m-Z_w} & \frac{Z_u + mu_0}{m-Z_w} & -mg \sin \theta_0 \\
0 & 0 & \frac{m-Z_w}{m-Z_w} & 1 \\
\frac{1}{l_{yy}} \left[ M_u + \frac{M_wZ_u}{m-Z_w} \right] & \frac{1}{l_{yy}} \left[ M_w + \frac{M_wZ_w}{m-Z_w} \right] & \frac{1}{l_{yy}} \left[ M_q + \frac{M_w(Z_q + mu_0)}{m-Z_w} \right] & \frac{1}{l_{yy}} \left[ -M_wmg \sin \theta_0 \right]
\end{bmatrix}
\]

\[
B_{long} = \begin{bmatrix}
\frac{X_{\delta_e}}{m} & \frac{X_{\delta_T}}{m} \\
\frac{Z_{\delta_e}}{m-Z_w} & \frac{Z_{\delta_T}}{m-Z_w} \\
0 & 0 \\
\frac{1}{l_{yy}} \left[ M_{\delta e} + \frac{M_wZ_{\delta e}}{m-Z_w} \right] & \frac{1}{l_{yy}} \left[ M_{\delta T} + \frac{M_wZ_{\delta T}}{m-Z_w} \right]
\end{bmatrix} \begin{bmatrix}
\Delta \delta_e \\
\Delta \delta_T
\end{bmatrix}
\]

The lateral equations are [19]:

\[
\begin{bmatrix}
\Delta \dot{\beta} \\
\Delta \dot{\phi} \\
\Delta \dot{\rho} \\
\Delta \phi
\end{bmatrix} = A_{lat} \begin{bmatrix}
\Delta \beta \\
\Delta \phi \\
\Delta \rho \\
\Delta \phi
\end{bmatrix} + B_{lat} \begin{bmatrix}
\delta_\alpha \\
\delta_r
\end{bmatrix}
\]

(13)

\[
\psi = r \sec \theta_0
\]

\[
I'_x = \left( I_x I_z - I_{xz}^2 \right) / I_z
\]

\[
I'_z = \left( I_x I_z - I_{xz}^2 \right) / I_x
\]

\[
I''_{xx} = I_{xx} / \left( I_x I_z - I_{xz}^2 \right)
\]

where:
\[
A_{lat} = \begin{bmatrix}
\frac{Y_v}{m} & \frac{Y_p}{m} & \left(\frac{Y_r}{m} - u_0\right) & g \cos \theta_0 \\
\left(\frac{L_v}{I_x} + I'_{zx}N_v\right) & \left(\frac{L_p}{I_x} + I'_{zx}N_p\right) & \left(\frac{L_r}{I_x} + I'_{zx}N_r\right) & 0 \\
\left(\frac{I'_{zx}}{I_z} + \frac{N_v}{I_z}\right) & \left(\frac{I'_{zx}}{I_z} + \frac{N_p}{I_z}\right) & \left(\frac{I'_{zx}}{I_z} + \frac{N_r}{I_z}\right) & 0 \\
0 & 1 & \tan \theta_0 & 0
\end{bmatrix}
\]

\[
B_{lat} = \begin{bmatrix}
\frac{Y_{\delta_a}}{m} & \frac{Y_{\delta_r}}{m} \\
\left(\frac{L_{\delta_a}}{I_x} + I'_{zx}N_{\delta_a}\right) & \left(\frac{L_{\delta_r}}{I_x} + I'_{zx}N_{\delta_r}\right) \\
\left(I'_{zx}L_{\delta_a} + \frac{N_{\delta_a}}{I_z}\right) & \left(I'_{zx}L_{\delta_r} + \frac{N_{\delta_r}}{I_z}\right) \\
0 & 0
\end{bmatrix}
\]

Note that the ‘Δ’ notation has been inserted for clarity and to remind the reader that all states are denoted as offsets from their equilibrium values. Note also that, as a result of the mathematical substitutions, the variable \(\psi\) is not included in the state vectors. Furthermore, these equations assume that the aircraft has four control surfaces – this issue will be addressed during later discussions of the Zagi aircraft model.

Since the nonlinear dynamic system has been decoupled into sets of linear longitudinal and lateral state-space equations, further dynamic analysis can be performed. This is discussed in the next section. Additionally, the controller design process has been greatly simplified, since the lateral and longitudinal control systems can be developed separately [19].

2.4 Modal Analysis

Textbooks on the topic of aircraft dynamics typically spend multiple chapters on the analysis of longitudinal and lateral dynamic modes. The interested reader may refer to [18-21] for in-depth discussions. The purpose here is to discuss these topics qualitatively in order
to provide the reader with an appreciation for their use in the discussion of aircraft dynamic performance. Furthermore, the reader will become aware of the qualitative dynamic characteristics of the Zagi aircraft used in this research and its inherent instability both in longitudinal and lateral manoeuvres. Some quantitative information is also provided for further insight.

Additionally, while the discussion and understanding of the dynamic characteristics and the modes of an aircraft is absolutely pertinent where piloted and/or passenger flight is involved, the discussion is not as critical for UAVs. In unmanned flight, one could argue that the closed loop system performance and controller design is limited only by the structural capabilities of the aircraft. On the other hand, in human-controlled flight, controller design must additionally take into account passenger comfort and pilot capability.

### 2.4.1 Longitudinal Modes

The eigenvalues of the system matrix $A_{lon}$ are used to characterize the open loop stability and dynamics of the longitudinal modes of an aircraft. The longitudinal system matrix of an aircraft has two sets of complex conjugate roots, representing two damped oscillations: one of low frequency and lightly damped oscillations, and the other of high frequency and more heavily damped. The conventional names for these modes are “phugoid” and “short-period” modes, respectively [19].

The phugoid, or long-period mode can be demonstrated in flight by trimming the aircraft in level flight, ‘pulling back’ on the control stick (to cause the aircraft to pitch up and lose airspeed), then returning the stick to the neutral position. The observed phugoid response consists of oscillations with significant variations in pitch attitude and airspeed, while the angle of attack remains relatively constant. It can be compared to the up and down motion of
a roller coaster, as there is a constant ongoing exchange between kinetic and potential energy. The oscillation starts with the exchange of airspeed for altitude as the aircraft climbs. As aircraft begins to slow, the pitch angle decreases, and the aircraft eventually pitches down. The descent is then followed by an increase in airspeed and pitch angle, and the aircraft returns to a climb again [21]. This is illustrated in Figure 2.3:

![Figure 2.3: Phugoid mode oscillations](image)

The short-period mode, on the other hand, is demonstrated in flight by trimming the aircraft and then subjecting it with a doublet of forward-aft-neutral pitch stick input, causing a sudden change in angle of attack. There is then a second order (or first order, in some cases) exponential decay to trim flight. The motion is characterized by significant oscillations in angle of attack and pitch attitude, while the airspeed remains essentially constant [21]. The short-period oscillations are illustrated in Figure 2.4, below.

### 2.4.2 Lateral Modes

The lateral dynamics typically consist of three modes: the ‘Dutch roll’ mode, the ‘roll’ mode, and the ‘spiral’ mode. The eigenvalues describing these methods can be found from the $A_{lat}$ matrix.
The Dutch roll mode is a second order response, typically consisting of simultaneous oscillations in sideslip angle, roll angle, and yaw angle. This mode is due to the coupling between yawing and rolling moments due to sideslip. These oscillations may be of high or low order and light or heavy damping, depending on the aerodynamic derivatives. The Dutch roll oscillations are typically initiated by a sideslip perturbation followed by oscillations in roll and yaw. The motion can be compared to that of an ice skater’s body weaving from side to side as weight shifts from one leg to another [21]. An illustration of Dutch roll oscillations is shown in Figure 2.5.

The roll mode eigenvalue has only a real component and its response is therefore first-order. The motion consists of nearly pure rolling action about the x-axis (in the stability reference frame). It may be excited by a rolling disturbance such as aileron input. With a step aileron input, there is an exponential increase in roll rate until a steady roll rate is achieved. The roll mode may be either stable or unstable depending on the angle of attack [21].
The spiral mode is also a first order response and it involves a relatively slow roll and yawing motion of the aircraft. It is usually initiated by a displacement in roll angle, and may be either stable or unstable. If it is stable, the aircraft returns to trim flight after the initial roll input. On the other hand, if it is unstable, the motion continues as a descending turn with increasing roll angle. Typically, the sideslip angle remains near zero while the roll and yaw angles vary.

2.5 The Zagi Aircraft Model

The aircraft under study in this research is a Zagi XS flying-wing with a 48-inch wingspan, shown in Figure 2.6. It is built of expanded polypropylene (EPP) foam and reinforced with multiple thin carbon-fibre beams. It is controlled by elevons and an electric
motor with an 8x6 propeller. Since there is no rudder, there is no method to directly control the yaw angle and therefore lateral mobility is less than ideal. This aircraft is part of the multi-UAV platform at UTIAS. These aircraft have been utilized extensively by UTIAS in the study of collaborative control methods [24, 25].

![Zagi flying-wing UAV](image)

The most relevant parameters that define the flight condition about which the aircraft dynamics have been linearized are provided in Table 2.2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Altitude</td>
<td>164.04 ft (50m)</td>
</tr>
<tr>
<td>Airspeed</td>
<td>45.93 ft/sec (14 m/s)</td>
</tr>
<tr>
<td>Weight</td>
<td>2.3 lb (1.04 kg)</td>
</tr>
<tr>
<td>Trim Angle of Attack</td>
<td>+1 degree</td>
</tr>
</tbody>
</table>

The stability and control derivatives for the aircraft model have been determined using software developed by Dr. Ruben Perez at the Royal Military College of Canada in collaboration with UTIAS [26] and are based on this information as well as other geometric and inertial properties of the aircraft. These aerodynamic derivatives can be found in Appendix A. Perez’s software emulates the DATCOM by calculating the stability and
control derivatives from the geometric and inertial properties of an aircraft at a specified flight condition. The longitudinal state space matrices have been determined and are provided below for reference.

\[
A_{\text{long}} = \begin{bmatrix}
0.0010 & 0.2733 & 0 & -32.1951 \\
-1.4040 & -9.5205 & 43.7724 & -0.5620 \\
0 & -0.5220 & -1.0904 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix},
B_{\text{long}} = \begin{bmatrix}
0 & 1.6631 \\
-159.7054 & 0 \\
-8.7899 & 0 \\
0 & 0 \\
\end{bmatrix}
\]

The lateral state space matrices are as follows:

\[
A_{\text{lat}} = \begin{bmatrix}
-0.0591 & -0.5945 & -45.8722 & 32.1951 \\
-0.6355 & -4.9037 & 1.3732 & 0 \\
-0.0040 & -0.3469 & -0.0104 & 0 \\
0 & 1 & 0.0175 & 0 \\
\end{bmatrix},
B_{\text{lat}} = \begin{bmatrix}
0 \\
84.8315 \\
-5.5444 \\
0 \\
\end{bmatrix}
\]

Note that there are no terms in the \( B_{\text{lat}} \) matrix corresponding to a rudder control surface and that the elevator angle is still represented as two different control surfaces in the lateral and longitudinal equations. On the Zagi aircraft, the elevons are used to control both pitch and roll motion through a process called ‘mixing.’ Since rolling motion is caused by differential deflection of the elevons and pitching motion is caused by symmetric deflections, the elevator mixing is performed by the as follows:

\[
\delta_{\text{right-elevator}} = \delta_e + \delta_a \\
\delta_{\text{left-elevator}} = \delta_e - \delta_a
\]

(14)

2.6 Modal Analysis of the Zagi Aircraft

A qualitative analysis of the dynamic modes of the Zagi aircraft is appropriate here in order to gain a basic understanding of its behaviour prior to controller design. It will be shown that the aircraft is open-loop unstable in both longitudinal and lateral motion.

2.6.1 Longitudinal Modes

The longitudinal motion pole-zero map of the Zagi aircraft is shown in Figure 2.7:
The short period mode is stable, with a frequency of 5.93 rad/sec (0.944 Hz) and damping of 0.912. The phugoid mode, on the other hand, is unstable, and has a frequency of 0.819 rad/sec (0.130 Hz) and poles at $s=0.102 \pm 0.813i$. The aircraft is therefore open-loop unstable to a step in pitch angle. This is an artefact of poor airframe design and causes the aircraft to be very challenging to control manually in the longitudinal directions. The pole locations are summarized in Table 2.3.

**Table 2.3: Zagi longitudinal modes**

<table>
<thead>
<tr>
<th>Pole (Mode)</th>
<th>Location</th>
<th>Stable/Unstable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phugoid</td>
<td>$s=0.102 \pm 0.813i$</td>
<td>Unstable</td>
</tr>
<tr>
<td>Short Period</td>
<td>$s=-5.41 \pm 2.43i$</td>
<td>Stable</td>
</tr>
</tbody>
</table>
2.6.2 Lateral Modes

The lateral motion pole-zero map of the aircraft is shown in Figure 2.8:

The Dutch roll mode is unstable with a frequency of 2.26 rad/sec (0.360 Hz) and poles at \( s = 0.422 \pm 2.22i \). The slow spiral mode is barely stable with a frequency of 0.0094 rad/sec (0.0015 Hz) and the faster roll mode is stable with a frequency of 5.81 rad/sec (0.925 Hz). The stability in the spiral and roll modes are good for open loop control purposes, however the unstable Dutch roll mode implies that some unstable oscillations may occur as a result of yaw inputs. The pole locations of the lateral modes are summarized in Table 2.4, below.
Table 2.4: Zagi Lateral modes

<table>
<thead>
<tr>
<th>Pole (Mode)</th>
<th>Location</th>
<th>Stable/Unstable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dutch Roll</td>
<td>0.422 ± 2.22i</td>
<td>Unstable</td>
</tr>
<tr>
<td>Roll</td>
<td>-5.81</td>
<td>Stable</td>
</tr>
<tr>
<td>Spiral</td>
<td>-0.0094</td>
<td>Stable</td>
</tr>
</tbody>
</table>

2.7 Summary

An outline of the development of a state-space model of aircraft motion has been provided and applied to the Zagi aircraft. The use of constant stability and control derivatives for these research purposes has been discussed as well. Finally, an analysis of the longitudinal and lateral dynamic modes of the Zagi aircraft has been performed.

In summary, the Zagi aircraft has unstable modes in both the longitudinal and lateral degrees of freedom. These open loop instabilities make the aircraft challenging to fly manually with a radio transmitter. Flying-wing aircraft with these dynamics are therefore often recommended for intermediate or advanced radio-controlled (RC) hobby aircraft pilots. Fortunately, since both the longitudinal and lateral dynamics are controllable, the unstable poles can be shifted into the open left hand side of the S-plane during controller design, ensuring closed loop stability.
CHAPTER 3

THE MULTIPLE SIMULTANEOUS
SPECIFICATION CONTROLLER DESIGN
METHOD

3.1 Introduction

The Multiple Simultaneous Specification (MSS) controller design method transforms
the problem of designing a single controller that satisfies \( n \) convex closed loop performance
specifications into one of designing simpler ‘sample’ controllers, each of which satisfies at
least one performance specification. This is followed by the convex combination of the
individual sample systems, and results in a single controller that satisfies all \( n \) performance
specifications. This final ‘MSS controller’ is extracted mathematically from the sample
controllers and plant dynamics and, therefore, no design is required beyond that of the
sample controllers. Consequently, the application of this design method presents the
possibility of greatly simplifying controller design problems in which many performance
specifications must be met. The following sections discuss in detail the mathematics of the
MSS controller design methodology.

3.2 Convex Specifications

The concept of convex specifications was introduced by Boyd, Barrat, and Norman in
the early 1990s [27]. The formal definition of a specification, \( D \), is a function or test on a
closed loop system, $H$:

$$D: \phi(H) < \alpha \rightarrow \{\text{Pass, Fail}\} \ H \in \mathcal{H}$$

where $\mathcal{H}$ is the set of all closed loop transfer functions, $\phi$ is a functional defined on $H$ (such as overshoot, etc.), and $\alpha$ is the required specification (a numerical value). The function $\phi$ is said to be convex if, for any two closed loop systems $H_1$ and $H_2$, and any $\lambda \in [0,1]$, we have:

$$\phi(\lambda H_1 + (1-\lambda)H_2) \leq \lambda \phi(H_1) + (1-\lambda) \phi(H_2)$$

(16)

This means that a specification is convex if its functional on the convex combination of two closed loop systems is less than or equal to the convex combination of the functional of the individual systems. This can also be described geometrically. If we define $F$ such that

$$F = \lambda H_1 + (1-\lambda)H_2$$

(17)

then at $\lambda = 0$, convexity requires

$$\phi(F) \leq \phi(H_2)$$

(18)

and at $\lambda = 1$, convexity requires

$$\phi(F) \leq \phi(H_1)$$

(19)

and everywhere in the range $\lambda \in (0,1)$,

$$\phi(F) \leq \lambda \phi(H_1) + (1-\lambda) \phi(H_2)$$

(20)

The right hand side of Eq. (20) is the equation of a straight line, $G$, passing through the points $(0,\phi(H_2))$ and $(1,\phi(H_1))$ and it can be graphed in the range $\lambda \in [0,1]$ along with $\phi(F)$. Therefore, a function is convex if, for every pair of transfer function matrices $H_1$ and $H_2$, the graph of $\phi(F)$ lies below the straight line $G$ in the range $\lambda \in (0,1)$. This is represented by the shaded region in Figure 3.1, below.

---

1 The definition of convexity in [27] states that $\lambda \in [0,1]$. However, when $\lambda=0$ or $\lambda=1$, even though convexity may hold, the statement loses some practical significance, since the weighting of one of the functions is then null and the result is trivial. Henceforth, the discussion assumes $\lambda \in (0,1)$. 

29
Many closed loop system performance specifications in both the time and frequency domains have been shown to be convex [27]. Among them are specifications such as overshoot, undershoot, settling time, and Bode magnitude plot envelopes.

### 3.3 Controller Design Framework

Consider a plant transfer function matrix, \( P(s) \), which is partitioned in the following format:

\[
P(s) = \begin{bmatrix} P_{zw}(s) & P_{zu}(s) \\ P_{yw}(s) & P_{yu}(s) \end{bmatrix}
\]

(21)

Here, \( w \) are the exogenous inputs to the system (disturbance signals, reference inputs), \( u \) are the control signals, \( z \) are the regulated signals, and \( y \) represents any and all of the signals available to the controller. As such, \( P_{zw} \) is the transfer function (matrix) from \( w \) to \( z \), \( P_{zu} \) is the transfer function (matrix) from \( u \) to \( z \), \( P_{yw} \) is the transfer function (matrix) from \( w \) to \( y \), and \( P_{yu} \) is the transfer function (matrix) from \( u \) to \( y \). The plant can also be written as follows:
A schematic of this system is shown in Figure 3.2:

![Diagram](image)

Figure 3.2: Open loop plant framework

A linear output feedback controller \( u(s) = K(s)y(s) \) can be designed for this system, and the corresponding closed loop system transfer function matrix \( H(s) \) from \( w \) to \( z \) can be found:

\[
Z(s) = H(s)W(s)
\]

(23)

The matrix \( H(s) \) can be written as a convex function of another matrix, \( R \):

\[
H(s) = P_{zw} + P_{zu}RP_{yw}
\]

(24)

where:

\[
R = K(I - P_{yu}K)^{-1}
\]

(25)

We see here that there is a one to one correspondence between matrix \( R \) and the controller \( K \).

If the output signal \( y \) is chosen to be the system error, with \( w \) and \( y \) being of the same dimension, then the framework can be represented as in Figure 3.3, below. This is the framework used in the design of the aircraft attitude controllers in later chapters.

### 3.4 MSS Controller Design: Problem Definition

The MSS controller design problem is defined as follows: given \( n \) desired convex closed-loop performance specifications, \( \phi_i(H) \leq \alpha_i, i = 1..n \), design an output feedback controller, \( K(s) \), such that the closed loop system satisfies all \( n \) performance specifications.
An important point to note is that the distinctive quality of the MSS controller design method is not its ability to produce a controller whose closed loop system simultaneously satisfies all \( n \) performance specifications, since this task can be achieved using many other design methods. Rather, its attractiveness is its ability to significantly simplify the way in which the controller \( K(s) \) can be designed.

### 3.5 MSS Controller Design Procedure

This section discusses the procedure and the mathematics of designing a controller, \( K(s) \) that solves the problem defined in Section 3.4. First, the material will be presented in detail, and then it will be summarized in a step-by-step.

#### 3.5.1 Sample Controllers

Consider the \( i^{th} \) desired closed loop performance specification. We define controller \( K_i(s) \) to be a sample controller if the closed loop system \( H_i(s) \) resulting from controller \( K_i(s) \) satisfies this specification:

\[
\phi_i(H_i) \leq \alpha_i
\]  

(26)

The system \( H_i(s) \) is then defined as the sample system corresponding to sample controller...
Recall that there are \( n \) desired closed loop performance specifications. Let the vector \( \psi \) contain all of these specifications: \( \psi = [\alpha_1 \ \alpha_2 \ \alpha_3 \ldots \alpha_n]^T \). The solution of the MSS design problem requires the designer to develop \( m \) sample controllers, \( K_i(s) \), \( i=1..m \), until each closed loop performance specification is satisfied by at least one controller. It is permissible to have multiple specifications satisfied by a single controller, so that \( m < n \).

### 3.5.2 Linear Programming and Convex Combination

Let \( \phi_{ij} \) denote the \( i^{th} \) specification resulting from closed loop system \( j \), so that there is a matrix \( \phi \):

\[
\phi = \{\phi_{ij}\} = \begin{bmatrix}
\phi_{11} & \ldots & \phi_{1m} \\
\vdots & \ddots & \vdots \\
\phi_{n1} & \ldots & \phi_{nm}
\end{bmatrix}
\]  

(27)

Using the information in the matrix \( \phi \) and vector \( \psi \), one can determine if a solution to the MSS controller design problem exists. However, in order to proceed with the solution algorithm, the matrix \( \phi \) must be square. As mentioned above, it may occur at this stage in the design that \( m < n \), i.e. there are fewer sample controllers than the number performance specifications. In this case, \( n - m \) columns must be added to \( \phi \), the elements of which are orders of magnitude larger than all other elements of \( \phi \), resulting in a new \( n \times n \) matrix. The additional columns simulate fictitious sample controllers that do not satisfy any specifications and that will not contribute to the final MSS control system. For ease of discussion and for differentiation between the number of sample systems and the number of performance specifications, \( \phi \) will still be described as an \( n \times m \) matrix; however, the reader must note that henceforth \( n = m \) because additional columns were added. The reason for this manipulation is due to the determinants that must be calculated as part of the solution.
process. This will become clear in the following discussion.

There are two conditions that must hold for the existence of a solution to the MSS controller design problem:

\[
F(\psi) = \frac{1}{(-1)^{n+1} \times \begin{vmatrix} \alpha_1 & \cdots & \alpha_n & 1 \\ \phi_{11} & \cdots & \phi_{n1} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \phi_{1m} & \cdots & \phi_{nm} & 1 \end{vmatrix}} = \frac{\begin{vmatrix} \alpha_1 & \cdots & \alpha_n & 1 \\ \phi^T & 1 \end{vmatrix}}{(-1)^{n+1} \times \text{det}(\phi)} \geq 0 \quad (28)
\]

and

\[
\alpha_i \geq \min(\phi_i(H_1), \phi_i(H_2), \phi_i(H_3), \ldots, \phi_i(H_m)) \quad (29)
\]

If both of these conditions hold, then a convex combination vector, \( \Lambda \), can be found through the solution of a linear programming problem:

\[
\phi \cdot \Lambda \leq \psi \quad (30)
\]

where \( \Lambda = [\lambda_1 \quad \lambda_2 \quad \lambda_3 \cdots \lambda_m]^T \), \( \lambda_i \in (0,1) \), \( \sum_{i=1}^{m} \lambda_i = 1 \). A closed loop system, \( H^* \), is then calculated as the convex combination of individual sample systems \( H_i(s) \) using the weighting of the elements in \( \Lambda \). The system \( H^* \) can then be shown to be a function of a matrix \( R^* \), just as \( H \) is a function of \( R \) in Eq. (24):

\[
H^* = \lambda_1 H_1 + \lambda_2 H_2 + \cdots \lambda_m H_m
\]

\[
H^* = \lambda_1 (P_{zw} + P_{zu} R_1 P_{yw}) + \lambda_2 (P_{zw} + P_{zu} R_2 P_{yw}) + \cdots \lambda_m (P_{zw} + P_{zu} R_m P_{yw}) \quad (31)
\]

\[
H^* = P_{zw} + P_{zu} R^* P_{yw}
\]

Using Eqs. (2), (16), and (17), one can verify that all closed loop specifications are then satisfied by \( H^* \). Consider one specification:

\footnote{For their derivation, the reader is referred to [28]}
Finally, it should be noted that the closed loop system in Eq. (31) is not the result of the direct convex combination of sample controllers $K_i$, rather it is a convex combination of matrices $R_i$ or sample systems $H_i(s)$.

$$R^* = \sum_{i=1}^{m} \lambda_i R_i \tag{33}$$

### 3.5.3 Extraction of MSS Controller

The MSS controller, $K^*$, i.e. the controller that satisfies all closed loop specifications, can be found from $R^*$ with some algebraic manipulation of Eq. (25):

$$K^* = (I + R^* P_{yu})^{-1} R^* \tag{34}$$

Therefore, once $\Lambda$ is calculated, $R^*$ can be found, and $K^*$ can be also determined.

The purpose of the vector $\Lambda$ is therefore to store the relative weighting of each of the sample systems in the overall solution. As mentioned in the previous section, the components of $\Lambda$ corresponding to the $n - m$ columns appended to the matrix $\phi$ will always be null since these columns represent fictional sample systems that have very poor closed loop performance. The optimization performed in the linear programming algorithm does not provide any weighting to these fictional sample systems.

Since $R^*$ is calculated with the summation in Eq. (33), if the model is of high order or if many sample controllers are used to satisfy the performance specifications, then $R^*$ will consequently be of high order. It may therefore be computationally demanding to perform the matrix inversions; furthermore, the controller $K^*$ that results may be of high order, and model
order reduction techniques may be required to reduce the order of $K^*$.

### 3.5.4 Summary of MSS Controller Design Procedure

In summary, the procedure to design MSS controller $K^*$ consists of the following:

1. Determine the open loop plant structure framework $P(s)$ in Eq. (21);
2. Design individual sample controllers, $K_i(s)$, $i=1..m$, until each closed loop design specification is satisfied by at least one controller;
3. Determine if a solution exists given the designed sample controllers by checking both conditions of Eqs. (28) and (29). If no solution exists, return to Step 2) and redesign controllers; otherwise continue to Step 4).
4. Solve the linear programming problem, Eq. (30), to determine the combination vector, $\Lambda$;
5. Calculate the matrix $R^*$ in Eq. (33);
6. Determine the MSS controller, $K^*$, from $R^*$ and $P_yu$ using Eq. (34).

### 3.6 Stability Analysis

The closed loop system $H^*(s)$ obtained through the convex combination of $m$ stable sample systems, $H_i(s)$, $i=1..m$, using the vector $\Lambda = [\lambda_1 \quad \lambda_2 \quad \lambda_3 \ldots \lambda_m]^T$ as described in Section 3.5.2, is also stable. As a proof, consider the state space models of the sample systems, $H_i(s)$:

$$H_i: (A_i, B_i, C_i, D_i) \quad A_i: N \times N, \quad B_i: N \times P, \quad C_i: Q \times N, \quad D_i: Q \times P \quad (35)$$

The transfer functions of the sample systems can be determined from the state space matrices:

$$H_i = C_i (sI_N - A_i)^{-1} B_i + D_i, \quad i = 1..m \quad (36)$$

It has been shown in Eq. (31) that the closed loop system $H^*$ is composed of the convex
combination of the individual sample systems:

\[ H^* = \sum_{i=1}^{m} \lambda_i H_i \]

\[ H^* = \sum_{i=1}^{m} \lambda_i (C_i(sI_N - A_i)^{-1}B_i + D_i) \]

\[ H^* = \sum_{i=1}^{m} \lambda_i (C_i(sI_N - A_i)^{-1}B_i) + \sum_{i=1}^{m} \lambda_i D_i \]

\[ H^* = [\lambda_1 C_1 \cdots \lambda_m C_m] \begin{bmatrix} sI_N - A_1 \\ \vdots \\ sI_N - A_m \end{bmatrix}^{-1} \begin{bmatrix} B_1 \\ \vdots \\ B_m \end{bmatrix} + [\lambda_1 D_1 \cdots \lambda_m D_m] \]

\[ H^* = [\lambda_1 C_1 \cdots \lambda_m C_m] \begin{bmatrix} sI_{mN} - \begin{pmatrix} A_1 \\ \vdots \\ A_m \end{pmatrix} \end{bmatrix}^{-1} \begin{bmatrix} B_1 \\ \vdots \\ B_m \end{bmatrix} + [\lambda_1 D_1 \cdots \lambda_m D_m] \]

\[ H^* = C^*(sI_{mN} - A^*)^{-1}B^* + D^* \]

The transfer function matrix \( H^* \) can therefore be represented by:

\[ H^* : A^*, B^*, C^*, D^* \rightarrow A^* : mN \times mN, \; B^* : mN \times P, \; C^* : Q \times mN, \; D^* : Q \times P \; \text{(38)} \]

It is important to note that while the sample systems \( H_i(s) \) and the final system \( H^*(s) \) have the same number of inputs, \( P \), and outputs, \( Q \), the number of states internal to \( H^* \) is \( mN \), i.e. the number of states multiplied by the number of states internal to the sample systems.

To complete the proof, we note that since the matrix \( A^* \) is diagonal, its eigenvalues consist of the union of the eigenvalues of the individual matrices \( A_i \), \( i = 1 \ldots m \):

\[ Eig(A^*) = Eig(A_1) \cup Eig(A_2) \ldots \cup Eig(A_m) \; \text{(39)} \]

Hence, if the sample systems \( H_i(s) \) are stable, i.e. the matrices \( A_i \) are Hurwitz, then the matrix
$A^*$ is Hurwitz, and $H^*$ is stable.

☐

3.7 Observations and Practical Notes on MSS Controller Design in MATLAB

Although the mathematics of the MSS controller design method is arguably quite elegant, there are some important points to note on its implementation in numerical software such as MATLAB. There are a number steps in the controller design process that require the multiplication, addition, or inversion of transfer function matrices. This can be very computationally expensive and may require a computer with a significant amount of memory. However, it was observed that computational time can be reduced by breaking down long operations into multiple, shorter steps. For example, the two most computationally-intensive operations in MATLAB are the calculations of the $R$ matrices of Eq. (25) and of the $K^*$ matrix in Eq. (34). The calculation of the $R$ matrices (recall that there is one $R$ matrix for every sample controller) in MATLAB can be performed in one line as follows:\(^3\):

$$r = \text{minreal}\left(\frac{K}{\text{eye}(5)-Pyu*K}\right);$$

However, it was observed that performing this calculation in one step causes instability in the resulting MSS controller. The exact reason for this is unknown and may be very difficult to determine, though it may be related to the way MATLAB inverts high order transfer function matrices. Once the operation is broken up into two steps, the computational time decreases significantly and the instabilities disappear:

$$\text{temp} = \text{minreal}\left(\text{inv(eye}(5)-Pyu*K)\right);$$
$$r = \text{minreal}(K*\text{temp});$$

\(^3\) The ‘minreal’ function is used to eliminate stable pole-zero cancellations.
For the same reasons, the calculation of matrix $K^*$ is broken into two steps from one step. The original code was:

\[
K_{\text{star}} = \text{minreal} \left( (1+R_{\text{star}}P_{yu}) \backslash R_{\text{star}} \right);
\]

and the modified version is:

\[
\text{temp} = \text{minreal}(1+R_{\text{star}}P_{yu}); \\
K_{\text{star}}=\text{minreal}(\text{temp} \backslash R_{\text{star}});
\]

One further point to note is that the MSS controller design method guarantees stability only for the strict mathematical procedure described in Section 3.5. However, it has been mentioned previously that the matrix $R^*$ is often of very high order since it is calculated via the summation of the individual matrices $R_i$. As a result, the transfer function matrix $K^*$ may also be of very high order, and model order reduction may be required in order to reduce the order to a level that is applicable for real time control. Unfortunately, stability is not guaranteed for a reduced-order model. As such, once the model order is reduced, stability must be verified by further simulation.

### 3.8 Summary

The Multiple Simultaneous Specification controller design method has been introduced and discussed in detail. The mathematical foundations and process for the method have been outlined and may hopefully serve as a reference or guide to those who wish to implement the process in the future. The stability of the controller design method has been proven. Finally, some practical observations and notes on the application of the controller design method in MATLAB are discussed.
CHAPTER 4

ATTITUDE CONTROLLER DESIGN AND SIMULATION

4.1 Introduction

The objective of this thesis is to design longitudinal (pitch) and lateral (roll) attitude controllers for the Zagi UAV based on the decoupled equations of motion. Since the longitudinal and lateral motions have been decoupled, the respective controllers can be designed individually. This is common practice in linear aircraft controller design, as in [19, 20]. Many specifications in aircraft control problems are convex in nature and therefore it is fitting to apply the MSS controller design method. The following sections describe the design of the controllers as well as provide continuous and discrete time linear model simulation results. The results demonstrate the effectiveness of the MSS controller design method.

With regards to notation, the following sections discuss the use of elevators and ailerons as the control surfaces for longitudinal and lateral motion, respectively. This notation is used simply to emphasize difference between lateral and longitudinal motions. On the Zagi aircraft, as discussed in Chapter 2, the there is only one set of control surfaces (in addition to the electric motor), referred to as elevons, which provide both lateral and longitudinal motion. Hence, for the discussions in this chapter, the control surfaces will be referred to as ailerons and elevators; however, they actuate the same physical control surface.
4.2 Selection of Closed Loop Performance Specifications

The choice of the number of specifications to place on the aircraft is subjective, and it is typically based on a trade-off of a number of factors. With more specifications placed on the closed loop system, it will naturally be more difficult to find a solution to the linear programming problem of Eq. (30). This may result in ‘looser’ overall specifications than a design problem that has fewer specifications. Furthermore, when more specifications are added and (presumably) more sample controllers are developed, the overall order of the resulting matrix $R^*$ as calculated in Eq. (33) increases significantly. This increases the required computational power to perform the controller design calculations and also requires more drastic model order reduction of the calculated MSS controller. With regards to this latter point, as discussed earlier, the closed loop performance is not guaranteed for a reduced-order controller. On the other hand, the beauty of the MSS controller design method is that if more specifications are added and the order of the resulting MSS controller can be reduced as required, then these specifications are welcome. For the each of the longitudinal and lateral attitude controllers, it has been compromised to place one envelope-type specification on the deflection of each of the control surfaces and then two additional specifications on the transient response of the aircraft in the respective manoeuvre. The longitudinal controller is therefore based around four specifications and the lateral controller is designed around three specifications.

4.3 Longitudinal Controller

The objective of the longitudinal attitude controller is to satisfy four specifications for a step input in pitch angle of +15 degrees. This is only one design problem in an overall longitudinal flight regime that includes takeoff, climb, cruise, descent, landing, etc., however
the pitch-up (or its counter-part, the pitch-down) manoeuver is critical in more complex manoeuvers and is therefore a good basis for controller design.

There are many convex specifications that can be chosen in the design problem, but this paper deals with four in particular. The first specification is that the five percent settling time of the pitch angle must be less than three seconds. Secondly, the overshoot during the manoeuver must be less than five percent. Additionally, the maximum deflection of the elevator during the manoeuver must be less than five degrees from the trim deflection angle. Finally, the maximum throttle above trim should be no more than +14 in order to prevent throttle saturation. For reference, full thrust is represented by a throttle value of +15.26. An explanation of these values and a detailed discussion on the scaling of the throttle values is provided in Chapter 5.

One must note that the total deflection of the elevators and throttle are calculated as their trim, or equilibrium, values plus an additional deflection:

\[
\delta_e = \delta_{e_{\text{trim}}} + \Delta \delta_e
\]
\[
\delta_t = \delta_{t_{\text{trim}}} + \Delta \delta_t
\]

The aforementioned specifications have been chosen based on the intuition gained from various flight tests and on the limits of the capabilities of the aircraft. The specifications are formally summarized as follows:

1) \( t_s \Delta \theta \leq 3 \) sec
2) Overshoot \( \Delta \theta \leq 5 \) %
3) Max \( |\Delta \delta_e| \leq 5 \) degrees
4) Max \( |\Delta \delta_t| < 14 \)

The design of the sample controllers and MSS controller follows the general MSS
design procedure described previously in Chapter 3. An extra integrator state, \( \Delta \theta_r = \int \Delta \theta_r \, dt \), has been added into the longitudinal equations in order to eliminate steady state pitch tracking error. The exogenous inputs of the system are therefore \( w = [\Delta u_r \ \Delta w_r \ \Delta q_r \ \Delta \theta_r \ \Delta \theta_r] \), the regulated outputs are \( z = [\Delta u \ \Delta w \ \Delta q \ \Delta \theta \ \Delta \theta] \), the output \( y = [\Delta u_e \ \Delta w_e \ \Delta q_e \ \Delta \theta_e \ \Delta \theta_e] \) is the system error, and the control signals are the elevator deflection and the throttle value: \( u = [\Delta \delta_e \ \Delta \delta_t] \). The output error feedback controller is therefore \( u = Ky \) where \( K \) is a 2x5 matrix. This fits into the framework of Boyd and Barrat in Figure 3.3. In order to determine the plant transfer function matrix \( P_{long}(s) \), we must first find the open loop 5x2 transfer function matrix \( G_{yu, long}(s) \) for the longitudinal system using the equation:

\[
G_{yu, long}(s) = D_{long} + C_{long}(sl - A_{long})^{-1}B_{long}
\]  

This transfer function matrix is of the form:

\[
G_{yu, long}(s) = \begin{bmatrix}
G_{u_{\delta e}} & G_{u_{\delta t}} \\
G_{w_{\delta e}} & G_{w_{\delta t}} \\
G_{q_{\delta e}} & G_{q_{\delta t}} \\
G_{\theta_{\delta e}} & G_{\theta_{\delta t}} \\
G_{\bar{\theta}_{\delta e}} & G_{\bar{\theta}_{\delta t}}
\end{bmatrix}
\]  

For brevity, the ten transfer functions are not included here. Once they are obtained, the plant \( P_{long}(s) \) can be determined:

\[
P_{long}(s) = \begin{bmatrix}
P_{zw}(s) & P_{zu}(s) \\
P_{yw}(s) & P_{yu}(s)
\end{bmatrix} = \begin{bmatrix}
0_{5x5} & G_{yu, long}(s) \\
I_{5x5} & -G_{yu, long}(s)
\end{bmatrix}
\]  

In this manner, we have satisfied the notation of Eq. (22).

Three controllers were designed using LQR techniques to satisfy the three convex specifications. The three controllers, \( K_{1, long} \) through \( K_{3, long} \), are as follows:
The performance of each closed loop sample system is charted in Table 4.1, below. The numbers in bold indicate that the corresponding sample system satisfies a closed loop specification. The section of the chart within the bolded outline represents the transpose of the matrix $\phi$ of Eq. (27).

Table 4.1: Performance of longitudinal sample systems

| System | $\phi_1$: $t_s \Delta \theta$ (sec) | $\phi_2$: Overshoot $\Delta \theta$ (%) | $\phi_3$: Max $|\Delta \delta_e|$ (degrees) | $\phi_4$: Max $|\Delta \delta_t|$ |
|--------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| $H_1$  | 4.83                           | 6.29                            | **1.98**                        | 17.08                           |
| $H_2$  | **2.47**                       | **0.92**                        | 5.24                            | **13.91**                       |
| $H_3$  | 3.67                           | 5.07                            | 5.08                            | **8.58**                        |
| Specification | 3            | 5                      | 5                      | 14          |

In this case, since there are more specifications than the number of sample systems, the matrix $\phi$ must be augmented with an extra column in order to be square. As discussed previously, this fourth column simulates a fourth sample system, $H_4$, which does not satisfy any specification and is therefore corresponds to a null weighting in the vector $\Lambda$. For clarity, the matrix $\phi$ is provided below:

$$\phi = \begin{bmatrix} 4.83 & 2.47 & 3.67 & 1 \times 10^{10} \\ 6.29 & 0.92 & 5.07 & 1 \times 10^{10} \\ 1.98 & 5.24 & 5.08 & 1 \times 10^{10} \\ 17.08 & 13.91 & 8.58 & 1 \times 10^{10} \end{bmatrix}$$ (45)

With these sample systems, the linear programming problem of Eq. (30) has a solution and the combination vector is found using the linear programming tools in MATLAB to be: $\Lambda = [0.11 \ 0.74 \ 0.15 \ 0]^T$. This combination vector indicates that

\[
K_{long}^1 = \begin{bmatrix} 4.83 & 2.47 & 3.67 & 1 \times 10^{10} \\ 6.29 & 0.92 & 5.07 & 1 \times 10^{10} \\ 1.98 & 5.24 & 5.08 & 1 \times 10^{10} \\ 17.08 & 13.91 & 8.58 & 1 \times 10^{10} \end{bmatrix}
\]
system $H_2$ is most heavily weighted in the convex combination, followed by systems $H_3$ and $H_1$, which are nearly equally weighted. This is likely due to the superior settling time and overshoot of system $H_2$ compared to the other two systems. Again, the fourth element of $\Lambda$ is zero because the hypothetical fourth system has very poor performance. The resulting MSS controller, $K_{long}^*$, is then found; however, it is not included here because it contains elements with terms of up to 72nd order. A controller of this order is not practical for real time implementation on the current autopilot hardware. Consequently, the model order reduction tools in MATLAB have been used to eliminate modes that are of least significance in the closed loop system. These tools have allowed for significant reduction of the controller order while maintaining a very similar closed loop response. The reduced-order MSS longitudinal controller is:

$$
K_{long}^* = \begin{bmatrix}
0.00 & -0.01 & -0.17 & -0.32 & -0.63 \\
1.32 & -0.95 & 16.34 & 49.90 & 11.02
\end{bmatrix}
$$

(46)

As described previously, the full order MSS controller is capable of satisfying all four performance specifications, but the performance of a reduced-order controller must be verified. Simulations were performed using Simulink, with a sample time of 10ms; these are considered the continuous time simulations. The discrete time simulations discussed later are performed with a sample time of 100ms. The result of the continuous time simulation of the closed loop longitudinal aircraft system using controller $K_{long}^*$ is shown in Figure 4.1. Three plots are shown: the first shows the pitch angle during the manoeuver, the second shows the elevator deflection, and the third shows the throttle. All four specifications have been simultaneously satisfied by the controller:

1) $t_s \Delta \theta = 2.61 \leq 3$ sec

2) Overshoot $\Delta \theta = 0.02\% \leq 5\%$
4.4 Lateral Controller

The purpose of the lateral attitude controller is to roll the aircraft to the desired bank angle. The controller has been designed for a step input of +15 degrees (banked turn to the right). The design of the lateral controller follows the same process as that of the longitudinal controller. There are three desired convex performance specifications for the lateral controller. The first is that the five percent settling time of the bank angle must be less than four seconds. Secondly, the maximum velocity in the $y$-axis, $\Delta v$, must be less than 4 ft/sec in order to reduce sideslip during the manoeuver. Finally, the third specification is a maximum aileron deflection of 1.5 degrees during the roll. The specifications on the elevator and
ailerons deflections in the longitudinal and lateral controllers have been placed in order to reduce the likelihood of saturation during complex manoeuvres. Stated formally, the three specifications are:

1) \( t_s \Delta \phi \leq 4 \text{ sec} \)

2) \( \text{Max } |\Delta v| \leq 4 \text{ ft/sec} \)

3) \( \text{Max } |\Delta \delta_a| \leq 1.5 \text{ degrees} \)

Note again that the lateral control signal is denoted as \( \Delta \delta_a \) in order to differentiate it from the longitudinal control signal \( \Delta \delta_e \), even though both signals represent actuations of the same control surfaces. The design of the sample controllers and MSS controllers continues as before. An extra integrator state is added here as well, \( \overline{\Delta \phi} = \int \Delta \phi dt \), to eliminate steady state roll tracking error. The exogenous inputs of the system are therefore \( w = [\Delta v_r \Delta p_r \Delta r_r \Delta \phi_r \overline{\Delta \phi}_r] \), the regulated outputs are \( z = [\Delta v \Delta p \Delta r \Delta \phi \overline{\Delta \phi}] \), the output \( y = [\Delta v_e \Delta p_e \Delta r_e \Delta \phi_e \overline{\Delta \phi}_e] \) is the system error, and the control signal is the aileron deflection, \( u = [\Delta \delta_a] \). The output error feedback controller is therefore \( u = Ky \) where \( K \) is a 1x5 matrix. The open loop transfer function matrix \( G_{yu, \text{lat}}(s) \) and the plant matrix \( P_{\text{lat}}(s) \) are similar in form to Eqs. (42) and (43).

As with the longitudinal controller, LQR methods were used to design sample controllers \( K_{1, \text{lat}} \) and \( K_{2, \text{lat}} \):

\[
K_{1, \text{lat}} = [0.01 \ 0.02 \ -0.58 \ -0.03 \ -0.18] \tag{47}
\]

\[
K_{2, \text{lat}} = [0.02 \ 0.01 \ -0.82 \ -0.12 \ -0.31]
\]

The performance of each closed loop sample system is charted in Table 4.2, below. The numbers in bold indicate that the corresponding sample system satisfies a closed loop specification. As with the longitudinal controller, since there are more
specifications than the number of sample systems, the matrix $\phi$ must be augmented with an extra column in order to be square. This third column simulates a third sample system, $H_3$, which does not satisfy any specification and is therefore given a null weighting in the vector $\Lambda$. The resulting matrix $\phi$ is provided below:

$$\phi = \begin{bmatrix} 4.11 & 3.28 & 1 \times 10^{10} \\ 3.21 & 4.19 & 1 \times 10^{10} \\ 0.94 & 1.78 & 1 \times 10^{10} \end{bmatrix} \quad (48)$$

| System | $\phi_1$: $t_s, \Delta \phi$ (sec) | $\phi_2$: Max $|\Delta v|$ (ft/sec) | $\phi_3$: Max $|\Delta \delta_a|$ (degrees) |
|--------|-----------------------------------|-----------------------------------|----------------------------------------|
| $H_1$  | 4.11                             | 3.21                              | 0.94                                   |
| $H_2$  | 3.28                             | 4.19                              | 1.78                                   |
| Specification | 4                  | 4                                   | 1.5                                    |

The linear programming problem has a solution and the weighting vector is $\Lambda = [0.51 \ 0.49 \ 0]^T$. The first two systems, $H_1$ and $H_2$, are nearly evenly weighted in the solution. Furthermore, only the first two terms are actually relevant, since the zero represents the weighting of the fictional system. After model order reduction in MATLAB, the MSS lateral controller, $K_{lat}^*$, is found:

$$K_{lat}^* = [0.01 \ 0.02 \ -0.66 \ -0.06 \ -0.22] \quad (49)$$

The results of the simulation using this controller are shown in Figure 4.2. The first plot shows the roll angle during the manoeuver, the second shows the y-axis velocity, and the third shows the aileron deflection angle. All three performance specifications have been met:

1) $t_s, \Delta \phi = 3.71 \leq 4$ sec
2) Max $|\Delta v| = 3.61 \leq 4$ ft/sec
3) Max $|\Delta \delta_a| = 1.03 \leq 1.5$ degrees
4.5 Discrete Time Simulation

The longitudinal and lateral MSS controllers have been applied in a simulation at a frequency of 10Hz while the system dynamics are calculated at 100Hz. In other words, the control laws are calculated one order of magnitude slower than the system dynamics, and hence this is a form of discrete time simulation. The 10Hz frequency is chosen since it is the rate at which control signals are generated during flight. Both controllers satisfy their respective performance specifications in the discrete time simulation. The longitudinal and lateral discrete time simulation results are shown in Figure 4.3 and Figure 4.4.

Figure 4.2: Simulation results for MSS lateral controller
Figure 4.3: Discrete time simulation results for MSS longitudinal controller

Figure 4.4: Discrete time simulation results for MSS lateral controller
Table 4.3 summarizes the performance of the MSS longitudinal and lateral controllers in both the continuous and discrete time simulations, demonstrating the success of the MSS controller design method.

Table 4.3: Continuous and discrete time MSS controller simulation results

<table>
<thead>
<tr>
<th>Controller</th>
<th>Specification</th>
<th>Continuous</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longitudinal</td>
<td>( t_s \Delta \theta \leq 3 \text{ sec} )</td>
<td>2.61 sec</td>
<td>2.55 sec</td>
</tr>
<tr>
<td></td>
<td>O.S. ( \Delta \theta \leq 5 % )</td>
<td>0.2 %</td>
<td>0 %</td>
</tr>
<tr>
<td></td>
<td>Max (</td>
<td>\Delta \delta_e</td>
<td>\leq 5 \text{ deg} )</td>
</tr>
<tr>
<td></td>
<td>Max (</td>
<td>\Delta \delta_i</td>
<td>\leq 14 )</td>
</tr>
<tr>
<td>Lateral</td>
<td>( t_s \Delta \phi \leq 4 \text{ sec} )</td>
<td>3.71 sec</td>
<td>3.19 sec</td>
</tr>
<tr>
<td></td>
<td>Max (</td>
<td>\Delta v</td>
<td>\leq 4 \text{ ft/sec} )</td>
</tr>
<tr>
<td></td>
<td>Max (</td>
<td>\Delta \delta_a</td>
<td>\leq 1.5 \text{ deg} )</td>
</tr>
</tbody>
</table>

It is important to note that although the MSS longitudinal and lateral controllers satisfy all performance specifications both in the continuous and discrete time simulations, there is no discrete time direct-design framework for the MSS controller design method. Unfortunately, as of the current time, the only method by which to verify the performance of an MSS controller in the discrete time is to discretize the system and check the simulation results.

4.6 Summary

Pitch and roll state-space attitude controllers have been designed for the Zagi aircraft using the MSS controller design method. The method by which a controller can be designed when there are more specifications than sample systems has been demonstrated for both the lateral and longitudinal controllers. Furthermore, both controllers satisfy their respective specifications in both the continuous and discrete time simulations using the linearized aircraft model. Until a discrete time formulation of the MSS controller design method is
developed, the only method of designing discrete time controllers is to design continuous
time controllers and then to verify their performance in a discrete time simulation.
CHAPTER 5

AUTOPILOT CONTROL HARDWARE & IMPLEMENTATION

5.1 Introduction

The autopilot on the Zagi UAV is implemented on two hardware systems. The first is the “Kestrel” autopilot from Procerus Technologies. The Kestrel autopilot can act as a standalone autopilot or serve as an interface between the outside world and the UAV by handling the communication protocols and interfacing with the aircraft control surfaces. The second hardware system for controlling the UAV is the Overo Fire Computer-on-Module (COM) from Gumstix Inc., which runs the MSS digital control system. These two systems are discussed in the following sections, as well as pertinent implementation details.

5.2 Kestrel Autopilot System

The Kestrel autopilot system from Procerus Technologies is shown in Figure 5.1, below. The autopilot provides autonomous flight control capabilities to a single UAV, including takeoff, GPS waypoint navigation, target following, and landing. It is mounted on the UAV and is connected to the control servos, batteries, and onboard sensors.

The real time autopilot software runs on the Kestrel’s 8-bit, 29 MHz microprocessor. The onboard sensor suite includes an inertial measurement unit, magnetometer, GPS, and differential an absolute pressure sensors. These sensors provide a complete inertial
navigation system for the UAV. The counterpart to the Kestrel autopilot is the ground station (typically a laptop computer) that is running the Procerus “Virtual Cockpit” software.

Figure 5.1: Kestrel autopilot [29]

The ground station also consists of a radio modem called the ‘Commbox’ that communicates with the onboard Maxstream radio modem over the 900 MHz frequency spectrum. The full platform, including the UAVs and the ground station, is shown in Figure 5.2, below.

In its typical operation, the autopilot onboard the UAV receives navigation commands from the Virtual Cockpit, and in return, sends telemetry data for real time mapping and post-flight analysis. A map of the flight zone can be downloaded from websites such as Google Maps [30] and loaded into Virtual Cockpit, allowing the UAV flight path to be superimposed on the map. Other telemetry data is also displayed within Virtual Cockpit, including navigation data, GPS satellite and radio communication strength, on-board battery voltage, motor speed, among many others. This provides the user with an excellent overall picture of the current “health” and flight status of the UAV.

After receiving high level navigation commands from the ground station, the autopilot executes various multi-level control loops that eventually actuate the control surfaces of the UAV. The details of these control loops are not of interest in this work, as the
The purpose of this research is to develop new, low level controllers, and therefore to operate the control surfaces directly and independently of Virtual Cockpit. The method by which this is accomplished is described in the following sections.

5.3 Serial Port Communication and Autopilot Communication Protocol

The Kestrel autopilot is equipped with multiple serial communication ports for integration with onboard computers or peripherals. Of particular interest is Serial Port A on the autopilot, which can be used to mirror (or duplicate) the modem communications port, thereby providing the ability for the autopilot to communicate with an external computer as if it were the ground station. The serial ports and sensors on the autopilot are shown in Figure 5.3. The four serial ports are located in the top-right corner.

Figure 5.2: UAV platform: aircraft and ground station
The autopilot communication protocol is implemented by a system of coordinated data transmission and acknowledgement packets. Data packets are assembled by Virtual Cockpit and/or an external computer, containing either control commands or requests for various types of information (telemetry, system status, etc.) Upon receiving a command-based packet from the ground station or the external computer, the autopilot responds with an acknowledgement (ACK) packet if the command is acceptable or with a negative acknowledgement (NACK) packet if the command has requested it to perform an action that is outside of its bounds. If the autopilot receives a request-based packet, it responds with a packet containing the requested data. The autopilot uses a checksum algorithm to determine if a packet has been corrupted during transmission, and corrupted packets are discarded accordingly. Through the use of communication data packets, full state information for the aircraft is available to the external computer, allowing for full state feedback for the MSS controllers.

5.4 The Gumstix Computer-on-Module

For this research, an Overo Fire Computer-on-Module (COM) board from Gumstix
Inc. shown in Figure 5.4, below, functions as the external control computer. It is mounted onboard the aircraft and runs the MSS controller. The term ‘Gumstix’ is appropriate as the brand name since the dimensions of the unit are on the scale of those of a piece of chewing gum. Its small size and light weight afford it the ability to be mounted to small aircraft such as the Zagi UAV. The Overo Fire COM is pre-loaded with the Linux operating system and has an ARM Cortex A-8 CPU and a C64x+ digital signal processor (DSP) core for 2D and 3D graphics acceleration. The CPU has a clock speed of 600MHz, and there is 256 MB of both RAM and on-board flash memory. The COM also has integrated 802.11g modem and Bluetooth wireless communication functionality. Many of these features are useful for single and multi-UAV research projects — for example, the graphics processing capabilities are useful for image-based control, and the wireless modem can be implemented in multi-UAV coordination and control applications.

The Gumstix COM connects to the ‘Tobi’ expansion board shown in Figure 5.5, below, in order to provide the Gumstix COM with access to serial communication ports, and a power supply. The Tobi expansion board also contains Ethernet, USB, audio and video ports. Consequently, Gumstix COMs, together with their expansion boards, are autonomous units and can function as complete computer systems with appropriate peripherals attached.
(such as a keyboard, mouse, and monitor). Additional software packages can be written and compiled on the Gumstix COM or on another computer and transferred to the COM for execution.

![Figure 5.5: Tobi expansion board [32]](image)

The current hardware implementation uses the Gumstix COM to execute the MSS controllers discussed in Chapter 4. Although any programming language could have been used, the MSS controllers have been written for the UAV in C++. The COM communicates with the Kestrel autopilot on Serial Port A and requests telemetry packets at a rate of 10Hz. These telemetry packets contain all pertinent state information for the digital controllers, including linear and rotational velocities, Euler angles, and other information such as GPS coordinates. After receiving the telemetry information, the controller outputs the desired control surface (elevon and motor) deflections to the autopilot through another packet. The autopilot then actuates the control surfaces accordingly. In this manner, the control system is partially independent of the ground station and the Virtual Cockpit software. A block diagram depicting this system of communication is shown in Figure 5.6, below. More detailed communication block diagrams are provided in the discussion on simulation in Chapter 6.
It is worthy to note that it is alternatively possible to use the ground station laptop as the external controller instead of the Gumstix COM. In this alternative scenario, the low level control surface commands come from the ground station and not from an on-board computer. Telemetry data would be transmitted to the ground station as usual; however, the controller, running in parallel with the Virtual Cockpit software on the ground station laptop, would transmit desired actuator deflections as packets to the UAV via the commbox. This is illustrated in Figure 5.7, below:
On the other hand, there are many advantages to using a Gumstix (or any onboard) computer to generate actuator commands as opposed to doing so on the ground station. The main advantage stems from the direct wired serial link between the Gumstix COM and the autopilot, allowing the COM to send and receive data packets directly to and from the autopilot. When transmitting over a wireless radio frequency to a ground station, there is a much greater risk of data corruption or loss [25]. Another disadvantage to using the ground station as the external controller is the fact that a relatively powerful computer must be used in order to simultaneously execute the Virtual Cockpit software and the controller software. The Overo Fire COM is inexpensive and is dedicated to the controller’s execution. Conversely, one advantage of a ground station-based controller is that it allows for centralized control of multiple UAVs. The control systems for all of the UAVs in a multi-UAV platform could be executed on the ground station, as opposed on individual Gumstix COMs on each individual aircraft. However, given that this research concentrates on the control of a single UAV, the decision has been made to use the Gumstix COM as the low level controller.

5.5 Basic On-board Electronic Connection Details

The major hardware components on-board the UAV during flight are the Gumstix COM with its Tobi expansion board, the Kestrel autopilot, two three-cell Lithium-Polymer (LiPo) batteries, and the electric motor.

The two 11.1V LiPo batteries operate in parallel and power all on-board electronics. The Gumstix COM and the autopilot each have voltage regulators to convert the incoming power to the required voltage levels. Power is also routed to the electric motor through an electronic speed control unit. Both the autopilot and the motor have power switches, allowing
the motor to be turned off while the autopilot is on (for safety) and further allowing the autopilot to be easily turned on or off without disconnecting the batteries. Figure 5.8, below, depicts all of the electronic hardware components on-board the UAV.

![Diagram of on-board main electronic components]

**Figure 5.8: On-board main electronic components**

## 5.6 Throttle Scaling and Trim

In Chapter 4, the design of the MSS attitude controllers is discussed, and simulations are performed using MATLAB and Simulink software packages. In the Simulink simulations, the value of the aircraft throttle, $\Delta \delta_T$, represents an offset from the equilibrium throttle value. It is imperative to calculate this equilibrium throttle value. Furthermore, the
throttle scale that is used in the simulations differs from the scale used on the autopilot hardware. On the autopilot hardware, the throttle is set on a scale of 0 to 1, representing a scale of 0 to 100 percent, where 100% is full throttle. In Simulink, the throttle scale is actually quite different, and it ranges from -1 to +15.26, as will be discussed shortly. Since the throttle value on the autopilot uses an absolute scale, it will be denoted as $\delta_T$. Note the slight change in notation from that of the throttle value in the Simulink simulation (i.e. the removal of the ‘Δ’ symbol). In order to implement the MSS controllers on the autopilot, the throttle values must be scaled appropriately from the Simulink values to the hardware values.

The equation for calculating thrust force from $\delta_T$ (where the value is represented on a scale from 0 to 1) using momentum theory is:

$$T = \frac{1}{2} \rho S_{prop} \left[ (K_m \delta_t)^2 - V^2 \right] Cx_{\delta_t}$$  

(50)

The following table summarizes each parameter and its value. The value of the motor constant, $K_m$ has been provided by the UAV manufacturer, Procerus Technologies.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>Total thrust force</td>
<td>Variable (lbf)</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Air density</td>
<td>0.0024 slug/ft$^3$</td>
</tr>
<tr>
<td>$S_{prop}$</td>
<td>Circular area covered by propeller ($\pi r^2$)</td>
<td>0.3484 ft$^2$</td>
</tr>
<tr>
<td>$K_m$</td>
<td>Motor constant</td>
<td>82.02 ft/s</td>
</tr>
<tr>
<td>$\delta_T$</td>
<td>Throttle value</td>
<td>Variable (dimensionless)</td>
</tr>
<tr>
<td>$V$</td>
<td>Airspeed</td>
<td>45.86 ft/s</td>
</tr>
<tr>
<td>$Cx_{\delta_t}$</td>
<td>Dimensionless X-force throttle coefficient</td>
<td>1(dimensionless)</td>
</tr>
</tbody>
</table>

The dimensionless throttle coefficient, $Cx_{\delta_t}$, is a scaling factor that represents the change in the X-force coefficient with the change in throttle value.

The software package by Perez that is used to determine the aircraft stability and control derivatives discussed in Chapter 2 calculates the equilibrium thrust value to be 0.1189
lbf (0.5289 N). Using Eq. (50), this corresponds to a throttle value of \( \delta_t = 0.596 \). Since this is the equilibrium throttle value, it corresponds to a throttle value of \( \Delta \delta_t = 0 \) in the Simulink simulations.

Furthermore, at full throttle (\( \delta_T = 1 \)), the thrust is 1.9331 lbf (8.599N). This value is intuitively sensible given the ability of the aircraft to attain large pitch angles and rapidly accelerate to speeds near 25 m/s (90 km/h). In order to calculate the throttle value \( \Delta \delta_T \) that corresponds to full throttle of \( \delta_t = 1 \), we approximate with the following equation:

\[
T = T_0 + X_{\delta_t} \Delta \delta_t
\]

(51)

The equation means that the total thrust force is equal to the equilibrium thrust force plus the change in the X force with change in throttle multiplied by the change in throttle from its equilibrium value. This equation assumes that all of the thrust is transmitted in the x-axis of the aircraft. The stability derivative \( X_{\delta_t} \) is calculated in Perez’s software to be 0.1189 lbf. One will note that this is the same value as the equilibrium thrust force itself. Accordingly, the throttle value \( \Delta \delta_t \) representing full throttle on the autopilot hardware is calculated as:

\[
\Delta \delta_t = \frac{1.9331 - 0.1189}{0.1189} = 15.26
\]

(52)

Finally, in order to generate no thrust from the propeller (at the reference flight speed) on the autopilot, we solve Eq. (50):

\[
0 = (82.02 \delta_t)^2 - 45.86^2
\]

(53)

\[
\delta_t = 0.559
\]

And in the simulation this corresponds to

\[
\Delta \delta_t = \frac{0 - 0.1189}{0.1189} = -1
\]

(54)
Based on these three data points, it can be seen that the mapping is nonlinear $\Delta \delta_T$ to $\delta_T$. This is summarized in Table 5.2:

<table>
<thead>
<tr>
<th>Thrust (lb)</th>
<th>Simulation Throttle Value ($\Delta \delta_T$)</th>
<th>Hardware Throttle Value ($\delta_T$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>0.559</td>
</tr>
<tr>
<td>0.1189</td>
<td>0</td>
<td>0.596</td>
</tr>
<tr>
<td>1.9331</td>
<td>15.26</td>
<td>1</td>
</tr>
</tbody>
</table>

In summary, when a throttle value is calculated by the MSS controller, the total thrust force generated by throttle value is calculated using Eq. (51). Then, Eq. (50) is solved for $\delta_t$ using this thrust force. The resulting $\delta_t$ value is the control signal used by the autopilot. One will note that this mapping varies with the reference flight speed and as such is only an approximation when applied throughout the flight regime.

### 5.7 Elevon Trim Deflection

The deflection of the elevator on an aircraft is often non-zero when the aircraft is in equilibrium, or trimmed flight. The elevator deflection during equilibrium flight is referred to as the trim deflection. It is not within the scope of this thesis to discuss the full details of the static stability of aircraft; however, the following basic development of the trim elevator (or elevon) deflection is appropriate.

The torque applied about the y-axis of an aircraft is calculated as:

$$M = \frac{1}{2} \rho V^2 S C_m$$

The parameter $C_m$ is the non-dimensional pitching moment coefficient and will be assumed to be a linear function of the angle of attack, the elevator deflection, and various airfoil properties [19]:

1.54
\[ C_m = C_{m_0} + C_{m_\alpha} \alpha + C_{m_{se}} \delta_e \]  

(56)

The non-dimensional stability and control derivatives \( C_{m_0} \), \( C_{m_\alpha} \), and \( C_{m_{se}} \) are calculated by Perez’s software to be 0, -0.2426, -0.0890, respectively. Here, \( C_{m_0} \) represents the contribution of the airfoil properties to \( C_m \). In equilibrium (trimmed flight), there is no pitching moment, and thus \( C_m \) is null. Eq. (56) can then be solved for \( \delta_e \) given that the trim angle of attack is known. Recall from Table 2.2 that the trim angle of attack is +1 degree (0.0175 rad):

\[ \delta_{e_{\text{trim}}} = \frac{(0.2426)(0.0175)}{-0.0890} = -0.0476 \]  

(57)

The trim elevator deflection is therefore -0.0476 radians or -2.73 degrees.

5.8 MSS Controller Software Implementation

5.8.1 Controller Structure

The MSS controllers are currently implemented on the Gumstix COM, which controls the UAV by sending aileron and elevator deflection as well as throttle value commands to the Kestrel autopilot. The autopilot mixes the aileron and elevator commands into elevon commands as discussed previously – see Eq. (14) in Chapter 2.

The Gumstix COM C++ code written for this research uses a multi-threaded software architecture. There are three threads running in parallel, and each performs a specific task or tasks:

1) The first thread is a low-level serial communications thread that handles bit-streams to and from the Gumstix COM;

2) The second thread requests for telemetry information from the autopilot at 10Hz, executes the MSS state feedback controllers, and sends the desired
control signals to the autopilot;

3) The third thread is used to set the reference command (step input in pitch, roll, or both) based on the value of a specific byte in the autopilot’s flash memory.

5.8.2 Controller Activation and Data Logging

To initiate the flight, the plane is launched with the Kestrel autopilot engaged and the Gumstix COM waiting in a standby mode. After launch, with the autopilot still engaged, the UAV begins to follow specified GPS-based waypoints. The low level MSS controllers do not initiate until commanded by the ground station.

There are various bytes of free space within the autopilot’s flash memory that can be used for debugging purposes, and these bytes can be set on the aircraft from within Virtual Cockpit while the UAV is airborne. A change in the value of the byte is sent wirelessly to the autopilot. The byte currently used to activate the controller is referred to on the autopilot as “Debug Byte 1.” The value of this byte is set to a default value of zero when the autopilot is powered on or its power is recycled. When the aircraft is in flight, this value can be set from within Virtual Cockpit to a specific value that corresponds to the desired manoeuvre, at which point the MSS controller is activated and the control signals calculated by the MSS controllers are sent to the autopilot and overwrite the control signals coming from the ground station.

The MSS controller software executes as a state machine. The Gumstix COM checks the debug byte every 0.1 seconds for changes to its value from the default of zero. When the ground station operator changes this byte to 1 from within Virtual Cockpit, the aircraft executes the pitch up manoeuvre using the MSS controllers. This interrupts the aircraft from flying along its designated flight path. After 15 seconds, or if the pitch angle decreases
significantly (indicating a probable crash), the controller execution finishes, the debug byte is set back to zero, and the regular UAV navigation continues as per the flight path set in Virtual Cockpit. A similar sequence occurs if the debug byte is set to 2; however, this initiates a roll manoeuvre. It is also important to note that both the lateral and longitudinal controllers execute simultaneously during flight once the debug byte is set to either 1 or 2, otherwise the aircraft may not be able to maintain a steady pitch angle during roll manoeuvres and vice versa. Flight telemetry data can then be recorded either on the Gumstix COM or on the ground station for post-flight analysis. A diagram of the state machine is shown below in Figure 5.9.

![Flight controller state machine diagram](image)

**Figure 5.9: Flight controller state machine diagram**

### 5.9 Summary

In this chapter, all of the electronic hardware components responsible for the autonomous control of the Zagi UAV were discussed. This includes the Kestrel Autopilot,
the Gumstix COM, and the Tobi expansion board. The throttle scaling from the MSS controller output value to the autopilot value is provided. The calculation of the elevator trim deflection is performed and an overview of the MSS controller software implementation on the Gumstix is provided.
CHAPTER 6

HARDWARE-IN-THE-LOOP SIMULATION

6.1 Introduction

It is imperative to simulate the behaviour of an engineered system such as an aircraft in as realistic and accurate as a manner as possible prior to performing experimental tests. This provides the engineer with a theoretical standard to which he or she can compare experimental results. Furthermore, assuming the system and environment have been properly modeled, accurate simulation allows the engineer to safely predict the behaviour of the system before any experiments are performed. This is especially valuable with airborne vehicles, as damage to aircraft such as the Zagi UAVs caused by crashes can be costly and time consuming to repair. Simulation is therefore a valuable tool for averting foreseeable failures.

Linear simulations of the MSS controller designs have been performed using Simulink and have been discussed in Chapter 4. However, prior to flight testing, it is desired to verify the behaviour of the closed loop system in a more realistic, nonlinear simulation environment.

Typically, two types of simulation methods are discussed in UAV literature: Software-in-the-loop (SIL) simulation and Hardware-in-the-loop (HIL) simulation [23, 33, 34]. The former is a method in which the aircraft and its environment are modeled in software; no physical hardware is required. HIL simulation, on the other hand, provides an
additional level of realism by making use of the aircraft hardware. It allows for testing of the aircraft actuators and control surfaces in order to check for correct operation, since these actuators physically move during the simulation. Another benefit of HIL simulation is that it allows for testing of communication protocols and often allows for testing of certain aspects of in-flight communication-based phenomena such as data loss, corruption, or transmission jitter. It also permits the use of external hardware during the simulation such as the Gumstix COMs. A high-level flow chart of the operation of a typical HIL simulation is shown in Figure 6.1:

![Flow Chart of HIL Simulation](image)

Figure 6.1: Basic operation of HIL simulators

This chapter discusses the topic of nonlinear HIL simulation of the Zagi UAV using the Kestrel autopilot system. It is expected that the exact results of a nonlinear simulation will vary from those of the linear simulations examined earlier; however, similar overall qualitative performance is expected.
6.2 Procerus UAV Simulation Software

There are two software packages that are used by the Procerus UAV system during for simulation. One is Virtual Cockpit (VC), which has been discussed repeatedly in previous chapters. The other is “Aviones,” which is the aircraft simulator. The aircraft dynamics, sensors, actuators, and flight environment are all simulated within Aviones during HIL simulation. Simulators such as Aviones may be referred to as ‘pseudo-nonlinear,’ since the simulated aircraft stability and control derivatives are constant yet its dynamics are calculated using the six nonlinear equations of motion.

Before discussing the SIL and HIL simulation modes in more detail, it is helpful to outline the operation of VC and Aviones during flight and simulation. Figure 6.2 shows the main VC window as it would appear during simulation. Table 6.1 lists and describes the basic elements of the VC window that are labelled in the figure.

Prior to the flight (real or simulated), the user uploads the desired waypoints for the flight to the autopilot from within VC. A ‘Takeoff’ flight mode is then selected (spiral, waypoint, or joystick), thereby engaging the throttle. The user then launches the aircraft. For real flights with the Zagi UAV, the aircraft is hand-launched (and Aviones is not used); in simulation mode, the aircraft is launched via the ‘UAV’ drop-down menu in Aviones. The aircraft will then climb, and once it reaches a safe altitude, it will automatically engage its ‘Navigation’ flight mode and fly along the desired flight path. The MSS controllers will engage only once a specific byte is set on the autopilot from within VC.
Figure 6.2: Virtual Cockpit main window

Table 6.1: Virtual Cockpit main window legend

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Flight mode selector</td>
</tr>
<tr>
<td>2</td>
<td>Attitude and telemetry information</td>
</tr>
<tr>
<td>3</td>
<td>Miscellaneous navigation data</td>
</tr>
<tr>
<td>4</td>
<td>Actual flight path</td>
</tr>
<tr>
<td>5</td>
<td>Aircraft</td>
</tr>
<tr>
<td>6</td>
<td>Desired flight path</td>
</tr>
<tr>
<td>7</td>
<td>Waypoint (2 of 3)</td>
</tr>
<tr>
<td>8</td>
<td>Warning message area</td>
</tr>
<tr>
<td>9</td>
<td>Ground Station GPS Position</td>
</tr>
</tbody>
</table>
The actual flight path, aircraft attitude, and telemetry data are all displayed in the VC screen as shown in the previous figure. Additional telemetry and real-time aircraft information are available from within various menus within VC. Figure 6.3, below, shows the Aviones simulator with the UAV depicted in black.

![Aviones aircraft simulator](image)

**Figure 6.3: Aviones aircraft simulator**

### 6.3 Simulation Modes

The Kestrel autopilot platform from Procerus Technologies allows for both SIL and HIL simulation. In SIL mode, VC and Aviones are run either on the same computer or two separate computers, and they communicate with one another using the TCP/IP protocol. The complete UAV system is simulated, including the Kestrel autopilot. Two initial conditions, namely latitude and longitude, are set in another Aviones configuration file. Unfortunately it is not possible to set an initial airspeed or altitude; the UAV must be re-launched from the
‘ground’ each time the simulation is restarted. While the aircraft is in simulated flight, VC provides it with high level commands, and the aircraft relays simulated telemetry data in return. Additional software can be written to communicate with VC and send externally-generated commands to the simulated aircraft. In real flight, these commands would be sent to the aircraft over the wireless modem via the commbox. This functionality is beyond the scope of this thesis research it requires the purchase of additional software packages from Procerus. Furthermore, as discussed previously, the current hardware implementation uses the Gumstix COM to send externally-generated commands to the autopilot.

The HIL simulation mode is more complex in its implementation than the SIL simulation mode, but it is more suitable for low level controller design where additional hardware such as the Gumstix COM is required. In HIL simulation, high level commands are generated by VC, initial conditions for the flight are set in Aviones, and the two programs communicate with each other over the TCP/IP connection as per the SIL simulation mode. However, the commands that are sent by VC to Aviones are also sent to the Kestrel autopilot over a serial connection. It is noteworthy that the physical autopilot is used here, unlike in the SIL simulation mode. Additionally, the low level control surface deflections generated by Aviones are physically actuated by the respective servos onboard the aircraft. During the HIL simulation, the Gumstix COM can also be used to communicate with the Autopilot, receive telemetry, and generate low level control commands. Telemetry information is then also relayed back through Aviones into VC and is displayed in the VC window. In this manner, during HIL simulation, the autopilot serves merely as a communications bridge between Aviones and the Gumstix COM.

To clarify these concepts, hardware communication block diagrams depicting both
the normal flight and the HIL simulation modes are shown in Figure 6.4 and Figure 6.5, respectively. Three important points are of note:

1) The autopilot in the block diagram is shown with two of its four communication ports: the modem port (labelled ‘M’) and Serial Port A (‘A’). These are the two main serial ports that are used for communication during flight and simulation.

2) The commbox can be used for HIL simulation; however, it is not required nor recommended. A wired serial connection between the ground station and the autopilot is preferred in HIL instead of the 900MHz wireless communication mode between the commbox and the Maxstream modem. This alternative configuration is recommended by Procerus because the wired serial connection can more robustly handle the substantial number of additional data packets that are transmitted between the autopilot and the ground station during simulation in addition to the standard telemetry packets that are transmitted during normal flight.

3) The SIL simulation mode (not shown) consists of only the ground station components of the HIL simulation block diagram since there is no communication between Aviones and the autopilot.

6.4 Running the HIL simulation

Appendix B contains step by step instructions for running the HIL simulation and serves as a reference for future students. These instructions serve only as a summary of the steps to initiate the simulation, and are not a substitute for the instruction manuals. Familiarity with both VC and Aviones is recommended before attempting to run the

75
simulations.

Figure 6.4: Hardware/communication block diagram: normal flight

Figure 6.5: Hardware/communication block diagram: HIL simulation mode

6.5 HIL Simulation Results

HIL simulations have been used to test the performance of the MSS longitudinal and lateral controllers. Upon performing initial simulations, a noticeable difference between the
linear (Simulink) simulations and the nonlinear (HIL) simulations had been noticed. This inconsistency was most likely due to differences between the dynamic model of the Zagi aircraft created by Procerus for Aviones and the dynamic model developed from Perez’s software (for which the MSS controllers were designed). The general inaccuracy in the Aviones model has been confirmed during discussions with an engineer at Procerus. Some slight modifications have since been made to the Aviones configuration files to improve the model accuracy; however, it is still expected that the in-flight behaviour of the UAV will be different than in the HIL simulation. Furthermore, by changing the model to be closer to the model developed using Perez’s software, the simulated Kestrel autopilot actually becomes unable to control the aircraft using the built-in navigation controllers. This is surprising because, as will be discussed in Chapter 7, the autopilot can adequately control the aircraft in real flight. In other words, the aircraft performance using the built-in autopilot is worse in the HIL simulation than in real flight. The inability for the autopilot to control the UAV in simulation made it extraordinarily difficult to set initial conditions for the execution of the MSS controllers.

Notwithstanding the aforementioned issues, HIL simulations were still performed with the hope that some of the qualitative behaviours of the UAV during pitch and roll manoeuvers could be observed.

The results of the pitch up manoever are shown in Figure 6.6, below. Three plots are presented for comparison with the corresponding plots of Figure 4.3. As can be seen from these results, the initial conditions for each of the parameters are non-zero, compared to the digital simulation results of Chapter 4. Furthermore, none of the closed loop performance specifications are met: there is significant pitch angle overshoot, the pitch angle does not
settle to within 5% of the +15 degree reference input during the controller’s execution, the elevator deflection exceeds 5 degrees away from trim, and the throttle saturates during the first second of the manoeuvre. However, as hypothesized above, the general behaviour of the UAV is as predicted by the linear simulations. The aircraft pitches up, the initial elevator deflection is negative and slowly increases, and the throttle initiates at a high value before decreasing to a near constant value. Upon comparison to Figure 4.3, overall trends within the plots are comparable to the discrete time simulation results.

The results of the lateral (roll) manoeuvre from the HIL simulation are shown below, in Figure 6.7. An additional note should be added here that, due to observed sensitivity of the aircraft to aileron deflection in high winds, a limit on the aileron deflection angle was set in the control software to +/- 2 degrees. Ideally, this would not affect the closed loop system performance, since there is a performance specification on the aileron deflection of a maximum deflection of +/- 1.5 degrees. However, from the figure it can be seen that in fact this saturation limit was reached.

As with the pitch up manoeuvre, initial conditions are not exactly as per the linear simulation. The manoeuvre begins at a positive roll angle of +20 degrees. The controller attempts to correct this initial roll angle error, however there is significant overshoot. The aircraft then rolls back in the positive direction and reaches a roll angle near +15 degrees. The y-axis velocity, \( \Delta v \), initially oscillates about 0 ft/sec with large amplitude, however it eventually oscillates with much smaller amplitude (approximately 2 ft/sec peak-to-peak). These small oscillations about a midpoint of 0 ft/sec are good results, indicating nearly zero sideslip during the manoeuvre. Finally the aileron deflection angle saturates at the limits of +/- 2 degrees.
Upon initial investigation, it appears that none of the closed loop performance specifications have been met. However, if one observes the system after 4 seconds, one of the three specifications is actually met: the maximum $y$-axis velocity is 2.193 ft/sec, i.e. less than the specified 4 ft/sec.

Figure 6.6: HIL simulation results - pitch up manoeuver (step applied at time $t=0^\circ$)
6.6 Summary

This chapter introduces the use of Software-in-the-Loop and Hardware-in-the-Loop simulation for the Zagi UAV. The benefits of Hardware-in-the-Loop simulation over Software-in-the-Loop and linear simulations are presented as well. The Aviones and Virtual Cockpit software packages that are provided by Procerus for simulation purposes are discussed in detail. Hardware-in-the-Loop simulations are performed to test the MSS
controller’s execution of pitch and roll manoeuvres. Although the HIL simulation results do not closely match those of the Simulink linear simulations, there are some benefits to the exercise. The simulated closed loop system performance is interesting to observe, there are some qualitative similarities to the linear simulations, and most importantly, the benefits of performing the simulation are still present: it has allowed the author to test the software that will be running on the Gumstix COM while the UAV is airborne. This includes testing the proper use of the data communication protocols and proper operation of the control surfaces.
CHAPTER 7

FLIGHT TESTING

7.1 Introduction

Flight testing of the Zagi UAV transpired from March until September 2010, once or twice every week. The flight test location was the Koffler Scientific Reserve at Joker’s Hill, located at 17000 Dufferin Street, in King City, Ontario, about 24 kilometers north of UTIAS. This section discusses the details of the flight testing process and the results of the flight test experiments using the MSS controllers.

7.2 Background: The Flight Test Experience

The flight tests were carried out, in most part, by the author and his colleagues at UTIAS, Mr. Mingfeng Zhang and Mr. Difu Shi. The ground station was set up on a hill overlooking the flight zone. A panoramic view of the flight zone as seen from the ground station is shown in Figure 7.1, and a bird’s eye view of the flight zone (outlined in red) and ground station location (red star) is shown in Figure 7.2.

Figure 7.1: Panoramic view of Joker's Hill flight zone
The ground station, as discussed previously, consists of a laptop computer, the commbox, and the RC transmitter. The commbox is connected to its own dedicated GPS receiver, in order to track the location of the ground station. This is useful for applications in which the ground station is moving. The RC transmitter is used to control the UAV manually when required. Due to the exceptional acceleration and flight speeds attainable by the Zagi aircraft as well as the open loop instabilities discussed in Chapter 2, controlling the UAV manually with the RC controller is a challenging task. Finally, the ground station is covered by a camping tent in order to provide shade for the flight testers and to reduce the brightness of the sun against the ground station laptop screen.

The flight tests require at least three people (flight operators) to be present. One of the
flight operators is responsible for UAV takeoff (hand-launching the aircraft) and controlling the aircraft with the RC controller when necessary – typically only in the case of an emergency. Since the RC is connected to the commbox at all times, this flight operator has limited mobility during the flight test. Another flight operator is required to operate Virtual Cockpit, log telemetry data, monitor the aircraft status, and update the flight path as required. This operator is therefore required to initiate execution of the pitch and roll manoeuvres by setting the proper debug byte, as discussed earlier. Finally, the third flight operator is required to monitor the aircraft visually at all times, take photos or videos, and assist with other duties as required.

The first three months of flight testing were spent on developing a familiarization with the Zagi UAV and its flight characteristics and with the ground station software, Virtual Cockpit. In addition, an operations protocol was developed, outlining the procedure for carrying out flight tests. The first months were difficult due to the learning curve required to properly fly the Zagi UAV as well as the repairs that were required after crashes. Some flight tests ended abruptly due to damage (from crashes) that could not be repaired in the field.

Following this initial stage, the next weeks were spent calibrating the aircraft sensors for flight. This was also a challenge, as the calibration procedure can only be performed with the aircraft under manual control with the RC transmitter. After the calibration period, the UAV was able to fly to the designated GPS waypoints as set within Virtual Cockpit. The level of accuracy with which the UAV could fly along its designated flight path was adequate for the purposes of this research. However, in strong winds (estimated at 15-30 km/h), the ability of the aircraft to track the correct heading and altitude diminished significantly. Moreover, some flight tests were cancelled due over-powering winds.
Once all of the preliminary sensor calibration and tuning was completed, development began on the integration of the MSS controllers on the Gumstix COMs. This was followed by HIL simulation (the results of which are in the previous chapter) and flight testing of the complete control system.

7.3 MSS Controller Flight Test Results

A number of flight tests were performed on the UAV using the MSS controllers. Both lateral (roll) and longitudinal (pitch-up) manoeuvres were attempted numerous times. The results varied between trials because of differences in wind conditions and differences in the initial conditions of the test manoeuvre. The results provided in this section are representative of the best results of the flight tests that were completed as of the date of the completion of this thesis.

The results of the pitch-up manoeuvre are shown below in Figure 7.3. A number of interesting observations can be made from these results. Firstly, as with the HIL simulations, it was very difficult to set the initial conditions for the manoeuvre. It can be seen from the figure that the initial pitch angle is in fact already approximately 15 degrees. The controller is able to maintain approximately this pitch angle for 2.5 seconds. At this point, a heavy gust actually caused the aircraft to pitch down and roll quickly to the right.

During the manoeuvre, the elevator is oscillating heavily. Presumably, due to the weather and wind conditions, the aircraft must use significant control effort to maintain a steady pitch angle. Another interesting observation is that the throttle is nearly saturated throughout the manoeuvre. This is likely caused by the thrust required by the aircraft to fly against the wind. From these results, it can be seen that none of the closed loop performance specifications have been met by the MSS controller. A discussion of possible reasons for this
result is provided in the next section.

The results of the roll manoeuver are provided in Figure 7.4, below. Unfortunately, due to hardware constraints during this flight test, the parameter $\Delta v$ could not be measured. The roll angle, however, can be seen to significantly overshoot the reference angle of 15
degrees. Initially, the aircraft overshoots the desired roll angle of 15 degrees, and then rolls back to correct this error. However, the aircraft then continues to roll to the right. Upon observation of the aileron control angle, one will see that the aileron deflection remains in the negative region. This may be the cause for the continued increase in roll angle. One will also note that there is no software-based saturation limit on the aileron deflection for this flight test as there was in the HIL simulation. The software-based limit was added after this flight test was completed; however, due to strong winds, the flight tests that were performed with the saturating limit in place actually had inferior results to this flight test. In summary, the three closed loop performance specifications on the lateral controller have not been satisfied as well during the flight testing phase.

Figure 7.4: Roll manoeuvre flight test results (step applied at time t=0°)
7.4 Discussion

The results of the previous section demonstrate that the closed loop performance specifications on the longitudinal and lateral manoeuvres have not been strictly met in-flight by the MSS controllers, even though they have been met in the discrete time linear simulations using Simulink. Furthermore, there are some distinct differences between the behaviour of the model in the HIL simulation and the flight tests. These differences can be attributed to a number of factors.

7.4.1 Initial Conditions

It has been mentioned a number of times that the proper initial conditions for the execution of the MSS controllers could not be obtained in the HIL simulation or during the flight tests. The actual initial condition for the manoeuvres is the trimmed (equilibrium) flight. This difference in initial conditions would have an effect on certain transient response characteristics of the aircraft (such as settling time and overshoot) once the manoeuvre is executed. The differences in initial conditions may also likely attribute to the differences in aircraft response between the HIL simulations and the flight tests.

7.4.2 Aircraft Dynamic Model

The development of the dynamic model of the UAV using Perez’s software was the first step of this research prior to the development of the MSS controllers. However, since then, some changes have occurred to the aircraft that have an effect on the dynamic model. Such changes include the addition of the Gumstix COM and Tobi expansion board that were recently on the aircraft. As well, there has been some damage to the aircraft due to crashes. This damage has been repaired, although the aerodynamic characteristics of the aircraft may have been impacted. Additionally, during the flight tests, the two LiPo batteries that powered
the electric motor and control hardware were mounted on the aircraft near the nose. However, the exact placement of the batteries may have been inconsistent from flight to flight and may shift during flight, even though they were mounted with Velcro. Any of these changes will have subsequently affected the state space model of the aircraft by modifying its geometric and inertial properties. It is impractical to update the dynamic model of the aircraft after every change to the airframe due to the length of time required for this process. As such, the original dynamic model was retained.

7.4.3 Wind Effects

The development of the dynamic model of the Zagi UAV provided in Chapter 2 is also based on the assumption that the effects of wind are negligible. Therefore, in the presence of strong winds, the accuracy of the state space model may be significantly reduced. Additionally, although the UAV can technically fly in winds up to 20-30 km/h, the pressure changes caused by the high-speed winds cause noise in altitude and velocity measurements [25], thus negatively affecting other aspects of the state space controllers that rely on these measurements.

7.4.4 Linear Thrust Model

Section 5.6 discusses the development of the linear thrust model for the Zagi aircraft and the algorithm for the conversion of the throttle value calculated by the controller into the throttle scale used by the autopilot. This throttle scale conversion algorithm depends on parameters that have been assumed constant, yet in actuality are time-varying and nonlinear with respect to the motion of the aircraft. This inaccuracy will have resulted in differences between actual thrust output and calculated thrust output and, consequently, caused
discrepancies between simulated and actual flight performance.

7.5 Summary

This chapter has provided a detailed description of the flight testing experience, including some of the difficulties that were encountered that prolonged the primary flight testing process. The MSS controllers were applied during the flight test and the results were discussed. A number of reasons were provided as to why none of the closed loop performance specifications were met by the aircraft during the flight tests.
CHAPTER 8

CONCLUSIONS, RECOMMENDATIONS, AND FUTURE WORK

8.1 Conclusions

This thesis presents the application of the MSS controller design method to aircraft control. Specifically, two MSS state space controllers are designed for a Zagi XS 48-inch wingspan flying-wing mini-UAV. Since the linear dynamic model of the aircraft is decoupled, separate longitudinal and lateral controllers are designed. Performance specifications are set for the lateral and longitudinal closed loop systems for roll and pitch-up manoeuvers, and these specifications are met by the MSS controllers in both continuous and discrete time linear simulations using Simulink. The MSS controller design method has therefore demonstrated the way in which it can be used to simplify a complex multiple-specification controller design problem.

The concept of SIL and HIL simulations are presented, along with the benefits of each method. HIL simulations are performed on the UAV and the results are compared qualitatively to the linear simulation results; however, none of the closed loop specifications are strictly satisfied.

A Gumstix COM has been used as an external on-board computer to execute the MSS controllers. It communicates with the Kestrel autopilot and sends it low level control commands. These commands overwrite the incoming commands from the ground station.
Experimental flight tests are performed and the results for the lateral and longitudinal manoeuvres are presented. The closed loop performance specifications are not strictly met by the MSS controllers in flight, however, it is interesting to compare the flight-test results to those of the linear and HIL simulations. A number of reasons have been suggested as to why the aircraft did not behave in flight as predicted by the linear simulations in Simulink and why the behaviour of the aircraft during flight tests differed from the HIL simulations.

8.2 Recommendations and Future Work

A number of recommendations can be suggested at this point with regard to future directions for this work:

8.2.1 Re-modeling of the UAV

The developing of a new dynamic model of the aircraft that incorporates the dynamic effects of the control hardware on-board would be useful for obtaining more accurate simulation results. Perez’s software could be used for the modeling process; however, the results from the software could be complemented by empirical data from actual flights. For example, in [23], the stability and control derivatives of a small UAV is determined through parameter identification from in-flight telemetry data.

Once the new model is developed, however, the MSS controllers would likely need to be redesigned.

8.2.2 Testing of Combined Lateral/Longitudinal Manoeuvres

The MSS controllers developed in this work were designed based on decoupled longitudinal and lateral state space equations. Additionally, only simple pitch and roll
manoeuvres were used as the standard reference commands. Agile aircraft such as the Zagi are capable of performing more complex manoeuvres, where both pitching and rolling occurs simultaneously. It would be advantageous to test the MSS controller performance under more complex manoeuvres.

8.2.3 New Flight Test Zone

As discussed in Chapter 7, the current flight test zone is the Koffler Scientific Reserve at Joker’s Hill. This flight test zone introduces a number of difficulties into the flight testing process. Firstly, it takes approximately 30-40 minutes to drive to Joker’s Hill from UTIAS. It takes a further 40-50 minutes to commute to UTIAS from the author’s home near the St. George Campus of the University of Toronto. As such, the total commute time is approximately 1.5 hours each way. During the summer months, flight tests were conducted once or twice per week, and as a result, a significant amount of time and money was spent simply on transportation to and from the flight tests. Furthermore, if the aircraft was ever damaged during flight tests to the extent that it had to be repaired with lab equipment, the flight operators returned to UTIAS and repaired the damage, but could not return to Joker’s Hill to continue with flight tests due to the amount of time that would be required to do so. Finally, and most importantly, Figure 7.2 shows that the actual flight area within Joker’s Hill is surrounded by trees, hills, and private farms, and that the section flight area within an unobstructed view is quite small: approximately 200m long by 100m wide. A much larger area of flat land would be ideal for the flight tests performed for this, as well as other UAV research, especially considering that the aircraft cruises at approximately 14 m/s.

It is therefore recommended that in the future, the UAV research group(s) at UTIAS seek out a larger flight zone that is close to UTIAS or the St. George Campus and that has a
flat, unobstructed view of at least 500 meters in each direction.

8.2.4 Design of MSS Controllers with Different Specifications

The results of presented in this thesis demonstrate the difficulties in achieving the closed loop performance specifications through flight tests, especially with low-weight aircraft such as the Zagi mini-UAV. This is mainly due to the difficulties in setting initial conditions and in rejecting external disturbances from sources such as wind. It may therefore be worth-while to design new longitudinal and lateral MSS controllers to satisfy a different set of specifications that are easier to test in-flight.

8.2.5 Outer Loop Controllers

The MSS controllers designed in this thesis are the inner-most controllers of a multi-loop aircraft control system because they directly actuate the control surfaces (the elevons and electric motor). Higher (outer) level controllers can then be designed to use these inner loop controllers. For example, an altitude-hold controller could calculate a desired pitch angle based on an altitude error signal. This desired pitch angle could then be fed to the MSS pitch angle controller, which could actuate the elevons to achieve the desired pitch angle.

8.2.6 Direct Digital Design Using the MSS Controller Design Method

The MSS controller design framework discussed in Chapter 2 is based around controller design in continuous time. To the best of the author’s knowledge, there is no mathematical framework for direct design of a discrete time controller using the MSS controller design method. Rather, the only way to design a discrete time controller is to design a controller in the continuous time framework and then test the closed loop system
using a discretized version of the controller; if the closed loop specifications are not met, then the controller must be redesigned. The development of a direct digital design methodology would therefore greatly simplify the work of digital control engineers.
REFERENCES


[34] Johnson, E. N., and Fontaine, S., “Use of Flight Simulation to Complement Flight
This appendix contains the lateral and longitudinal stability and control derivatives (in their dimensional form) for the Zagi UAV. The general notation for the dimensional derivatives is: $X_u = \frac{\partial x}{\partial u}, Z_u = \frac{\partial z}{\partial u}$, etc.

### Table A.1: Longitudinal Stability and Control Dimensional Derivatives

<table>
<thead>
<tr>
<th>Derivative</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_u$</td>
<td>-0.0051</td>
</tr>
<tr>
<td>$X_{Ta}$</td>
<td>0.0052</td>
</tr>
<tr>
<td>$X_w$</td>
<td>0.0195</td>
</tr>
<tr>
<td>$X_q$</td>
<td>0</td>
</tr>
<tr>
<td>$Y_u$</td>
<td>0</td>
</tr>
<tr>
<td>$Y_w$</td>
<td>0</td>
</tr>
<tr>
<td>$Y_q$</td>
<td>0</td>
</tr>
<tr>
<td>$Z_u$</td>
<td>-0.1003</td>
</tr>
<tr>
<td>$Z_w$</td>
<td>-0.6800</td>
</tr>
<tr>
<td>$Z_q$</td>
<td>-0.1491</td>
</tr>
<tr>
<td>$M_u$</td>
<td>-6.9670e-08</td>
</tr>
<tr>
<td>$M_{Ta}$</td>
<td>0</td>
</tr>
<tr>
<td>$M_w$</td>
<td>-0.0266</td>
</tr>
<tr>
<td>$M_q$</td>
<td>-0.0556</td>
</tr>
<tr>
<td>$M_{\alpha}$</td>
<td>0</td>
</tr>
<tr>
<td>$M_{\theta}$</td>
<td>0</td>
</tr>
<tr>
<td>$M_{\phi}$</td>
<td>0</td>
</tr>
<tr>
<td>$X_{\delta_e}$</td>
<td>0</td>
</tr>
<tr>
<td>$Z_{\delta_e}$</td>
<td>-11.4075</td>
</tr>
</tbody>
</table>
Table A.2: Lateral Stability and Control Dimensional Derivatives

<table>
<thead>
<tr>
<th>Derivative</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{\delta_e}$</td>
<td>-0.4483</td>
</tr>
<tr>
<td>$X_{\delta_T}$</td>
<td>0.1188</td>
</tr>
<tr>
<td>$Z_{\delta_T}$</td>
<td>0</td>
</tr>
<tr>
<td>$M_{\delta_T}$</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Derivative</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_{\psi}$</td>
<td>-0.0042</td>
</tr>
<tr>
<td>$Y_p$</td>
<td>-0.0425</td>
</tr>
<tr>
<td>$Y_r$</td>
<td>-8.5695e-04</td>
</tr>
<tr>
<td>$Y_{\phi}$</td>
<td>0</td>
</tr>
<tr>
<td>$Y_{\theta}$</td>
<td>0</td>
</tr>
<tr>
<td>$Y_{\theta}$</td>
<td>0</td>
</tr>
<tr>
<td>$L_{\psi}$</td>
<td>-0.0645</td>
</tr>
<tr>
<td>$L_p$</td>
<td>-0.4972</td>
</tr>
<tr>
<td>$L_r$</td>
<td>0.1395</td>
</tr>
<tr>
<td>$L_{\phi}$</td>
<td>0</td>
</tr>
<tr>
<td>$L_{\theta}$</td>
<td>0</td>
</tr>
<tr>
<td>$L_{\theta}$</td>
<td>0</td>
</tr>
<tr>
<td>$N_{\psi}$</td>
<td>8.9266e-04</td>
</tr>
<tr>
<td>$N_p$</td>
<td>-0.0408</td>
</tr>
<tr>
<td>$N_r$</td>
<td>-0.0048</td>
</tr>
<tr>
<td>$N_{\phi}$</td>
<td>0</td>
</tr>
<tr>
<td>$N_{\theta}$</td>
<td>0</td>
</tr>
<tr>
<td>$N_{\theta}$</td>
<td>0</td>
</tr>
<tr>
<td>$Y_{\delta_a}$</td>
<td>0</td>
</tr>
<tr>
<td>$L_{\delta_a}$</td>
<td>8.6292</td>
</tr>
<tr>
<td>$N_{\delta_a}$</td>
<td>-1.0373</td>
</tr>
</tbody>
</table>
APPENDIX B

HARDWARE-IN-THE-LOOP SIMULATION PROTOCOL

A USB to serial converter cable is required for HIL simulation if the computer that is running Aviones does not have a standard serial port. This is because the serial communication cable provided by Procerus is only compatible with 9-pin serial ports. Through trial and error it has been found that, for some unknown reason, only certain USB to serial converter cables can be used for this purpose. Although all of the converter cables are visibly identical, some fail to work for HIL simulation. This finding has been confirmed by. Consequently, if there is no communication between the ground station and the autopilot in Step 7, below, it is recommended that the user try to use a different USB-serial converter cable. The specific cable mentioned below (part number HL340) has proven to work and it is therefore recommended for this purpose.

The steps for executing the HIL simulation are as follows:

1) Disconnect the autopilot from the radio modem (the bottom two hardware boards).

2) Connect the HL340 USB-Serial converter cable to the PC and determine which communications port it uses (for example, use ‘Device Manager’ in Windows’ Control Panel).
3) Connect the DB9 serial connector of the HL340 to the DB9 connector of the programming cable, and connect the Molex connector of the programming cable to the autopilot’s modem port. Ensure that the 5th wire (the programming wire) of the programming cable is disconnected.

4) Open the Aviones configuration file, ‘autopilot1032_regs.txt’, in the Aviones folder and change the first line to match the communications port number of the HL340 (from Step 2). For example, if communications port number 4 is used, then the line must read:

```
HIL Comm Port,4.000000
```

5) Set the starting latitude and longitude in ‘autopilot1032_regs.txt’. If the file ever changes in the future, set these initial conditions back to their desired values.

6) Power on the Autopilot.

7) Start VC and open the communications port that the HL340 is connected to; Verify that communication occurs.

8) Run Aviones. There should not be an airplane visible yet on-screen.

9) In VC, go to Agents->Edit Agent List, and check the box ‘HIL Sim’ next to the UAV indicator; Click OK; The VC warning message area should display a message reading: “Warning: Simulation Mode Enabled – DO NOT FLY!”

10) The screen will refresh, an airplane will appear in Aviones, and the UAVs initial conditions will be set in VC as per Step 5.

11) Click ‘Zero Pressure,’ to calibrate the virtual pressure sensors.

12) Upload a flight plan.
13) Select a takeoff mode in VC.

14) In Aviones, from the ‘UAV’ drop-down menu, select ‘Launch If Necessary’ or ‘Relaunch Current UAV’ to launch the UAV. The aircraft will launch and movement will be visible both in VC and Aviones.