# TRANSITIONS BETWEEN HOVER AND LEVEL FLIGHT FOR A TAILSITTER UAV

by

Stephen R. Osborne

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

Brigham Young University

December 2007

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Stephen R. Osborne

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

| | |
|---|---|
| Date | Randal W. Beard, Chair |
| Date | Timothy W. McLain |
| Date | D.J. Lee |

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Stephen R. Osborne in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Randal W. Beard
Chair, Graduate Committee

Accepted for the Department

Michael A. Jensen
Chair

Accepted for the College

Alan R. Parkinson
Dean, Ira A. Fulton College of Engineering and Technology

## ABSTRACT

TRANSITIONS BETWEEN HOVER AND LEVEL FLIGHT FOR A TAILSITTER UAV

Stephen R. Osborne

Department of Electrical and Computer Engineering

Master of Science

Vertical Take-Off and Land (VTOL) Unmanned Air Vehicles (UAVs) possess several desirable characteristics, such as being able to hover and take-off or land in confined areas. One type of VTOL airframe, the tailsitter, has all of these advantages, as well as being able to fly in the more energy-efficient level flight mode. The tailsitter can track trajectories that successfully transition between hover and level flight modes. Three methods for performing transitions are described: a simple controller, a feedback linearization controller, and an adaptive controller. An autopilot navigational state machine with appropriate transitioning between level and hover waypoints is also presented. The simple controller is useful for performing a immediate transition. It is very quick to react and maintains altitude during the maneuver, but tracking is not performed in the lateral direction. The feedback linearization controller and adaptive controller both perform equally well at tracking transition trajectories in lateral and longitudinal directions, but the adaptive controller requires knowledge of far fewer parameters.

# ACKNOWLEDGEMENTS

I would first like to thank the members of my committee for their support and guidance in the creation of this thesis. Dr. Beard and Dr. McLain also deserve my gratitude for their leadership in the Magicc Lab, where I have worked for the past two years. Dr. Beard has also been an invaluable resource in directing my research and proofreading this thesis numerous times. Along with these professors, I also thank all my fellow Magicc Labbers who have helped me in any way, shape, or form during my stay here. Special thanks goes to Nate Knoebel for his assistance and expertise on tailsitter issues, as well as for many hours spent during flight testing in the summer heat.

# Table of Contents

**Acknowledgements** **xi**

**List of Tables** **xvii**

**List of Figures** **xix**

**1 Introduction** **1**

1.1 Background and Motivation . . . . . . . . . . . . . . . . . . . . . . . 1

1.2 Literature Review . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.3 Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

1.4 Document Organization . . . . . . . . . . . . . . . . . . . . . . . . . 5

**2 Tailsitter Physics** **7**

2.1 Three-dimensional Model . . . . . . . . . . . . . . . . . . . . . . . . 7

2.1.1 Quaternion Motivation and Definition . . . . . . . . . . . . . . 7

2.1.2 Quaternion/Euler Conversions . . . . . . . . . . . . . . . . . . 9

2.1.3 Navigation Equations . . . . . . . . . . . . . . . . . . . . . . . 9

2.1.4 Kinematic Equations . . . . . . . . . . . . . . . . . . . . . . . 10

2.1.5 Force Equations . . . . . . . . . . . . . . . . . . . . . . . . . . 11

2.2 Two-dimensional Model . . . . . . . . . . . . . . . . . . . . . . . . . 13

2.3 Chapter Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

**3 Experimental Platform** **19**

3.1 Simulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

3.2 Flight Test Setup . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

3.3 Quaternion Attitude Control . . . . . . . . . . . . . . . . . . . . . . 21

3.4 Hover Position Control . . . . . . . . . . . . . . . . . . . . . . . . 23

3.5 Level Flight Control . . . . . . . . . . . . . . . . . . . . . . . . . 26

3.6 Chapter Summary . . . . . . . . . . . . . . . . . . . . . . . . . . 28

**4 Trajectory Tracking Algorithms** **31**

4.1 Desired Trajectories . . . . . . . . . . . . . . . . . . . . . . . . . 31

4.2 Simple Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . 35

4.3 Feedback Linearization Controller . . . . . . . . . . . . . . . . . . . 37

4.4 Adaptive Controller . . . . . . . . . . . . . . . . . . . . . . . . . . 40

4.4.1 Lyapunov Stability . . . . . . . . . . . . . . . . . . . . . . . 40

4.4.2 Equations of Motion . . . . . . . . . . . . . . . . . . . . . . 41

4.4.3 Reference Model . . . . . . . . . . . . . . . . . . . . . . . . 41

4.4.4 Controller Derivation . . . . . . . . . . . . . . . . . . . . . . 42

4.5 Simulation Results . . . . . . . . . . . . . . . . . . . . . . . . . . 45

4.5.1 Simple Controller Simulations . . . . . . . . . . . . . . . . . . 46

4.5.2 Feedback Linearization Controller Simulations . . . . . . . . . 48

4.5.3 Adaptive Controller Simulations . . . . . . . . . . . . . . . . . 50

4.6 Flight Test Results . . . . . . . . . . . . . . . . . . . . . . . . . . 53

4.6.1 Simple Controller Flight Results . . . . . . . . . . . . . . . . . 53

4.6.2 Feedback Linearization Controller Flight Results . . . . . . . . 56

4.6.3 Adaptive Controller Flight Results . . . . . . . . . . . . . . . 59

4.7 Chapter Summary . . . . . . . . . . . . . . . . . . . . . . . . . . 62

**5 Autopilot State Machine** **63**

5.1 Autopilot Structure . . . . . . . . . . . . . . . . . . . . . . . . . . . 63

5.2 Autopilot Flight Results with Simple Controller . . . . . . . . . . . . 66

5.3 Autopilot Flight Results with Feedback Linearization Controller . . . 69

5.4 Autopilot Flight Results with Adaptive Controller . . . . . . . . . . . 71

5.5 Chapter Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . 73

**6 Conclusion** **75**

6.1 Summary of Results . . . . . . . . . . . . . . . . . . . . . . . . . . 75

6.2 Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75

**Bibliography** **77**

## List of Tables

5.1 Autopilot state machine state numbers . . . . . . . . . . . . . . . . . . . 67

# List of Figures

1.1 Convair XFY-1 Pogo, 1954 . . . . . . . . . . . . . . . . . . . . . . . . 2

1.2 Ryan X-13 Vertijet, 1956 . . . . . . . . . . . . . . . . . . . . . . . . 2

2.1 Lift and drag coefficients versus $\alpha$. . . . . . . . . . . . . . . . . . . . . . . . 13

2.2 Forces acting on tailsitter . . . . . . . . . . . . . . . . . . . . . . . . 14

3.1 The Magicc Lab Pogo airframe . . . . . . . . . . . . . . . . . . . . . . 20

3.2 Kestrel Autopilot . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

3.3 Quaternion attitude controller performance in flight test . . . . . . . 24

3.4 Hover position tracking flight data . . . . . . . . . . . . . . . . . . . . 27

3.5 Level waypoint tracking flight data . . . . . . . . . . . . . . . . . . . . 29

4.1 Hover-to-level trajectory goal . . . . . . . . . . . . . . . . . . . . . . 32

4.2 Level-to-hover trajectory goal . . . . . . . . . . . . . . . . . . . . . . 32

4.3 Desired $p$-trajectories . . . . . . . . . . . . . . . . . . . . . . . . . . 34

4.4 Desired $h$-trajectories . . . . . . . . . . . . . . . . . . . . . . . . . . 36

4.5 Simple controller hover-to-level trajectory tracking in simulation . . . 46

4.6 Simple controller level-to-hover trajectory tracking in simulation . . . 47

4.7 Feedback linearization controller hover-to-level trajectory tracking in simulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48

4.8 Feedback linearization controller level-to-hover trajectory tracking in simulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49

4.9 Adaptive controller hover-to-level trajectory tracking in simulation . . 51

4.10 Adaptive controller level-to-hover trajectory tracking in simulation . . 52

4.11 Simple controller hover-to-level transition flight data . . . . . . . . . 54

4.12 Simple controller level-to-hover transition flight data . . . . . . . . . 55

4.13 Feedback linearization controller hover-to-level trajectory tracking flight data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57

4.14 Feedback linearization controller level-to-hover trajectory tracking flight data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58

4.15 Adaptive controller hover-to-level trajectory tracking flight data . . . 60

4.16 Adaptive controller level-to-hover trajectory tracking flight data . . . 61

5.1 Autopilot state machine . . . . . . . . . . . . . . . . . . . . . . . . . 64

5.2 State machine flight results with simple controller . . . . . . . . . . . 68

5.3 State machine flight results with feedback linearization controller . . . 70

5.4 State machine flight results with adaptive controller . . . . . . . . . . 72

# Chapter 1

# Introduction

## 1.1 Background and Motivation

Throughout the history of flight, new airframes have constantly been developed in response to changing needs and new mission requirements. One unique airframe design in particular is the tailsitter. Tailsitters, as the name implies, sit on their tail when not in flight. They take off and land vertically, making them a member of the VTOL (Vertical Take-Off and Land) family of aircraft. Equipped with a powerful engine, tailsitters can utilize a "prop-hanging" technique to hover in place. Additionally, they can transition to a level flight mode and fly in a traditional fixed-wing mode, which is much more energy efficient than hover mode. Being able to transition between hover and level modes opens the door to a wide variety of possible missions unavailable to traditional aircraft or other VTOL airframes like helicopters, which stay in the energy-inefficient hover mode at all times.

In the years following World War II, the tailsitter design concept was explored in depth and even developed into a few experimental aircraft for flight testing. The primary research focus was to develop a short-range combat aircraft that could take off from a confined environment like the deck of a Navy destroyer. One such aircraft was the Convair XFY-1 Pogo, first flown in 1954 and pictured in Figure 1.1. The Ryan X-13 Vertijet in Figure 1.2 was another famous jet-powered tailsitter design that successfully flew in 1956. Although the tailsitter was aerodynamically sound and made sense on paper, in practice it proved difficult for the test pilots to fly. Controlling the tailsitter in hover mode, and especially landing while looking at the ground over one's shoulder was very tricky and dangerous. Ultimately, the tailsitter design was abandoned and largely forgotten.

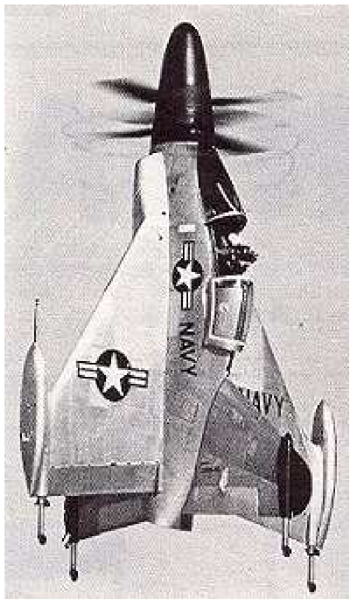


**Figure 1.1:** Convair XFY-1 Pogo, 1954

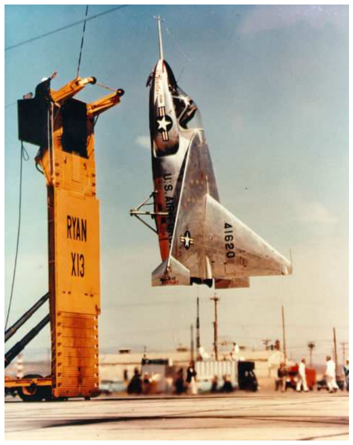


**Figure 1.2:** Ryan X-13 Vertijet, 1956

In recent years, advances in miniaturization and computation have enabled a rapid increase in research and development of Unmanned Air Vehicles (UAVs). Small UAVs in particular have proved very useful in both military and civilian applications, such as aerial surveillance, target tracking, forest fire monitoring, border patrol, search and rescue, and as links in communication networks. As UAV research has continued, the advantages of a VTOL UAV capable of hover flight have become clear. For instance, when monitoring a stationary or slow moving target, a UAV unable to hover must make several passes over the area, resulting in the target being unavailable for continual reconnaissance. With a hover-capable UAV, however, the target can remain under surveillance for as long as needed, constrained only by the battery life of the UAV.

The limitations of finite battery life quickly become significant with a typical hovering UAV. Hovering requires a great amount of energy because the airframe's entire weight is generally lifted by the force of the propulsion system alone, without any of the benefits that a lifting surface gives to a normal fixed-wing platform. Adding more batteries to a small UAV to increase flight time only increases the weight, often negating any desired improvement.

Primarily due to the problem of simultaneously desiring long flight time and the advantages of hover flight, the tailsitter airframe design has been renewed for use as a UAV platform. The main difficulties with the tailsitter design when first conceived and tested in the 1950's are largely solved with a computerized autopilot in command, rather than a human pilot. The BYU Magicc Lab has developed a tailsitter UAV autopilot that allows the UAV to hover or travel in level mode between waypoints. Of particular importance are the transitions between the two flight regimes. The primary focus of the research in this thesis is to develop a method to safely transition in a controlled manner between vertical and horizontal modes while tracking a desired trajectory.

## 1.2 Literature Review

Tailsitter research is still in its infancy. The BYU Magicc Lab has demonstrated flight results on UAVs that contribute to the work of this thesis. In this thesis, an adaptive control algorithm similar to [1] is described. An overview of preliminary tailsitter research and a description of BYU's tailsitter research platform is contained in [2].

In addition to the contributions of this thesis and the work of the BYU Magicc Lab, R. Hugh Stone of the University of Sydney, Australia has published papers detailing efforts to construct and fly an autonomous tailsitter. In [3], the preliminary airframe design of Stone's tailsitter is presented. A good overview of the project as well as a description of possible applications for tailsitters is given in [4]. The control and guidance architecture is presented in [5]. Of particular interest to this thesis is [6], which describes optimization methods of the tailsitter's stall-tumble maneuver transitions between hover and level flight.

Although not much literature is available detailing their efforts, a tailsitter UAV named SkyTote has been under development since 1998 by AeroVironment [7], a company specializing in unmanned aircraft systems. As discussed in [8], the SkyTote is being designed as a precision cargo delivery system, capable of delivering a 50 pound payload to within a 15 foot area up to 200 miles away from the mission start point, with a 1.5 hour max battery life. Mission parameters like these are ideal for a tailsitter which has the precision landing capability of a hovering UAV and the energy-efficiency of a fixed wing airframe. At this time, it is unknown what progress has been made on the SkyTote system.

Green and Oh at Drexel University have contributed several papers of interest to tailsitter research. Although the flight platform described is not a tailsitter, it can transition between level and hover flight. Much of the research focus is on developing a UAV that can fly in confined spaces, such as inside buildings. These efforts are described in [9], [10] and [11]. The same authors also use vision-based guidance systems to control small MAVs with similar hover characteristics to a miniature tailsitter in [12], [13], and [14].

Other authors contribute useful theoretical discussion applicable to tailsitter research. Methods for trajectory tracking with a VTOL aircraft are presented in [15], [16], and [17]. Costic, et. al., [18] describe a quaternion-based attitude tracking controller for spacecraft. Trajectory tracking for fixed-wing UAVs performing aggressive flight maneuvers is explored in [19]. An excellent description of attitude representations, including an extensive description of the quaternion representation, is presented in [20].

## 1.3 Contributions

The research described in this thesis presents three different control methods which may be used for transitioning a VTOL tailsitter UAV between hover and level flight modes. Simulation and flight test results for each of the modes are also presented. The tailsitter physics models used in the derivation as well as simulation of the control methods are also given, and will prove useful to others seeking to further develop tailsitter autopilot technology. Finally, a tailsitter navigational autopilot is presented which provides a framework for flying a flight plan composed of hover and level waypoints.

## 1.4 Document Organization

In Chapter 2, both 2D and 3D physics models are presented that will be used in later derivations of controllers as well as for testing the algorithms in simulation. Chapter 3 describes the simulation and hardware environments used in this research. Chapter 4 contains derivations for simple, feedback linearization, and adaptive controllers for negotiating transitions between flight modes for a tailsitter UAV. Simulation and flight test results for transitions are also included. In Chapter 5, a navigational autopilot for flying a flight path composed of mixed hover and level waypoints is described and flight results are given. Conclusions and recommendations for future work are given in Chapter 6.

# Chapter 2

# Tailsitter Physics

The starting point for developing trajectory tracking control is to develop an accurate physical model of the system. Two models are presented in this chapter, describing three-dimensional and two-dimensional dynamics. The three-dimensional, quaternion-based, high fidelity model is used in simulation testing of the tracking algorithms, but is too complex to be used as the basis for developing implementable control. Therefore, a two-dimensional model that captures the essential features of the system is used to develop the control algorithms.

## 2.1 Three-dimensional Model

The three-dimensional physics model uses a quaternion representation of tailsitter attitude. Quaternions, as described in this section, provide a unique description of attitude without the singularity introduced in a hover position by an Euler angle attitude representation.

### 2.1.1 Quaternion Motivation and Definition

Euler angles are traditionally used to represent aircraft attitude in aerospace literature. Starting from an aircraft orientation with the nose facing North (the world frame $x$ axis), the right wing facing East (the world frame $y$ axis) and the belly facing down (the world frame $z$ axis), each angle signifies a rotation about an individual axis. The order of rotations is non-commutative and so a standard order of rotations is used. First the aircraft is rotated about the $z$ axis by $\psi$, called the yaw angle. Then the aircraft is rotated about the newly created $y$ axis by $\theta$, the pitch angle. The final rotation is about the new $x$ axis and is called the roll angle, $\phi$.

Euler angles are an intuitive measure of the aircraft's attitude and thus are very useful in aircraft control. However, this Euler angle representation has singularities at $\theta = \pm\frac{\pi}{2}$. At these points when the pitch angle is pointed straight up or straight down, a situation called gimbal lock occurs where the body-fixed $x$ and inertial $z$ axes are now aligned. This situation is analogous to being at the North or South pole of the Earth where all longitudinal lines come together at a point, or singularity. At the North pole, for instance, all directions point south. For the aircraft Euler angle representation, no yaw information can be gathered at the singularities and attitude cannot be properly determined. Quaternions, fortunately, do not suffer from gimbal lock and provide a singularity-free version of attitude representation.

A quaternion contains four elements and may be thought of as the composition of an axis of rotation and an angle specifying the magnitude of rotation about that axis. A rotation of $\Theta$ radians about a three-dimensional vector $\vec{v}$ is represented as the quaternion

$$\eta = \begin{pmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \end{pmatrix} = \begin{pmatrix} \sin\frac{\Theta}{2} v_1 \\ \sin\frac{\Theta}{2} v_2 \\ \sin\frac{\Theta}{2} v_3 \\ \cos\frac{\Theta}{2} \end{pmatrix}. \tag{2.1}$$

For the tailsitter, attitude can be described with a single quaternion. This attitude quaternion represents the axis of rotation (defined in the world frame) and magnitude of rotation to achieve the current tailsitter orientation when starting from the initial attitude of nose facing North and right wing facing East.

### 2.1.2 Quaternion/Euler Conversions

A unit quaternion can be translated to traditional Euler angle representation by the transformation

$$\begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix} = \begin{pmatrix} \tan^{-1} \frac{2(\eta_2\eta_3+\eta_4\eta_1)}{1-2(\eta_1^2+\eta_2^2)} \\ \sin^{-1}(-2(\eta_1\eta_3 - \eta_4\eta_2)) \\ \tan^{-1} \frac{2(\eta_1\eta_2+\eta_4\eta_3)}{1-2(\eta_2^2+\eta_3^2)} \end{pmatrix} \tag{2.2}$$

and Euler angles are converted to a quaternion by

$$\begin{pmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \end{pmatrix} = \begin{pmatrix} \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \\ \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \end{pmatrix}. \tag{2.3}$$

Due to the singularity at $\theta = \pm\frac{\pi}{2}$, care must be taken when converting from a quaternion to Euler angles when the pitch of the aircraft is near these values, as $\psi$ will be indeterminate.

### 2.1.3 Navigation Equations

Body frame velocities are translated to the inertial frame by

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

or, with quaternions,

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} 1-2(\eta_2^2+\eta_3^2) & 2(\eta_1\eta_2 - \eta_4\eta_3) & 2(\eta_1\eta_3 + \eta_4\eta_2) \\ 2(\eta_1\eta_2 + \eta_4\eta_3) & 1-2(\eta_1^2+\eta_3^2) & 2(\eta_2\eta_3 - \eta_4\eta_1) \\ 2(\eta_1\eta_3 - \eta_4\eta_2) & 2(\eta_2\eta_3 + \eta_4\eta_1) & 1-2(\eta_1^2+\eta_2^2) \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}. \tag{2.4}$$

### 2.1.4 Kinematic Equations

We assume a quaternion-based attitude controller. The quaternion update equation is

$$\dot{\eta} = \frac{1}{2} A \Omega$$

with

$$A = \begin{pmatrix} \eta_4 & -\eta_3 & \eta_2 \\ \eta_3 & \eta_4 & -\eta_1 \\ -\eta_2 & \eta_1 & \eta_4 \\ -\eta_1 & -\eta_2 & -\eta_3 \end{pmatrix}$$

and

$$\Omega = \begin{pmatrix} p \\ q \\ r \end{pmatrix}$$

where $p$, $q$, and $r$ are angular velocities about the body frame $x$, $y$, and $z$ axes, respectively. The angular rates are updated by

$$\dot{\Omega} = k_\Omega(\Omega_c - \Omega)$$

where

$$\Omega_c = k_e \eta_e - k_d \Omega.$$

In these equations, $k_\Omega$, $k_e$, and $k_d$ are gains and $\eta_e$ is known as the error quaternion, given by

$$\eta_e = \begin{pmatrix} \eta_4 & \eta_3 & -\eta_2 & -\eta_1 \\ -\eta_3 & \eta_4 & \eta_1 & -\eta_2 \\ \eta_2 & -\eta_1 & \eta_4 & -\eta_3 \\ \eta_1 & \eta_2 & \eta_3 & \eta_4 \end{pmatrix} \eta_c.$$

The error quaternion is the error between the tailsitter's current attitude, given by $\eta$, and the desired attitude, $\eta_c$. The attitude controller adjusts the angular rates $p$, $q$, and $r$ to achieve the desired orientation. In practice, the angular rates are adjusted by controlling the aileron, elevator, and rudder control surfaces.

### 2.1.5 Force Equations

World frame accelerations are found by taking the derivative of (2.4):

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = N \begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} + \dot{N} \begin{pmatrix} u \\ v \\ w \end{pmatrix} \tag{2.5}$$

$$= N\dot{V} + \dot{N}V \tag{2.6}$$

where $N$ is the rotation matrix from (2.4) and

$$\dot{N} = \begin{pmatrix} -4\eta_3\dot{\eta}_3 - 4\eta_4\dot{\eta}_4 & 2\eta_2\dot{\eta}_3 + 2\dot{\eta}_2\eta_3 - 2\eta_1\dot{\eta}_4 - 2\dot{\eta}_1\eta_4 & 2\eta_2\dot{\eta}_4 + 2\dot{\eta}_2\eta_4 + 2\eta_1\dot{\eta}_3 + 2\dot{\eta}_1\eta_3 \\ 2\eta_2\dot{\eta}_3 + 2\dot{\eta}_2\eta_3 + 2\eta_1\dot{\eta}_4 + 2\dot{\eta}_1\eta_4 & -4\eta_2\dot{\eta}_2 - 4\eta_4\dot{\eta}_4 & 2\eta_3\dot{\eta}_4 + 2\dot{\eta}_3\eta_4 - 2\eta_1\dot{\eta}_2 - 2\dot{\eta}_1\eta_2 \\ 2\eta_2\dot{\eta}_4 + 2\dot{\eta}_2\eta_4 - 2\eta_1\dot{\eta}_3 - 2\dot{\eta}_1\eta_3 & 2\eta_3\dot{\eta}_4 + 2\dot{\eta}_3\eta_4 + 2\eta_1\dot{\eta}_2 + 2\dot{\eta}_1\eta_2 & -4\eta_2\dot{\eta}_2 - 4\eta_3\dot{\eta}_3 \end{pmatrix}$$

and

$$\dot{V} = (-\Omega \times V) + N^{-1}G + \frac{1}{m}T + \frac{1}{m}L + \frac{1}{m}D$$

where $G$ is the world frame gravity vector, $T$ is the thrust vector, and $L$ and $D$ are lift and drag vectors given by

$$L = \frac{\rho V^2 S}{2} \begin{pmatrix} \sin\alpha C_l(\alpha) \\ 0 \\ -\cos\alpha C_l(\alpha) \end{pmatrix}$$

and

$$D = \frac{\rho V^2 S}{2} \begin{pmatrix} -\cos\alpha C_d(\alpha) \\ 0 \\ -\sin\alpha C_d(\alpha) \end{pmatrix}$$

where $C_l(\alpha)$ and $C_d(\alpha)$ are the coefficients of lift and drag and are approximated by

$$C_l = \begin{cases} \sin 2\alpha + C_{l\alpha}\bar{\alpha}e^{-\lambda(\alpha-\bar{\alpha})}, & \alpha > \bar{\alpha} \\ \sin 2\alpha - C_{l\alpha}\bar{\alpha}e^{\lambda(\alpha+\bar{\alpha})}, & \alpha < -\bar{\alpha} \\ \sin 2\alpha + C_{l\alpha}\alpha, & \text{otherwise} \end{cases} \tag{2.7}$$

and

$$C_d = \begin{cases} C_{d2}(\pi/2)^2 + C_{d4}(\pi/2)^4, & \alpha > \pi/2 \\ C_{d2}(\pi/2)^2 + C_{d4}(\pi/2)^4, & \alpha < -\pi/2 \\ C_{d2}\alpha^2 + C_{d4}\alpha^4, & \text{otherwise.} \end{cases} \tag{2.8}$$

The approximations of $C_l$ and $C_d$ were obtained by an approximation to experimental data. Figure 2.1 shows the approximation using the formulae for $C_l$ and $C_d$ described above. A number of parameters help define the shape of the curves. For the approximation to closely match the experimental data curves, the values are $C_{l\alpha} = 4.0$, $\bar{\alpha} = 0.1676$, $\lambda = 20$, $C_{d2} = 1.0$, and $C_{d4} = -0.2$. These are the values used in simulations presented in this thesis.

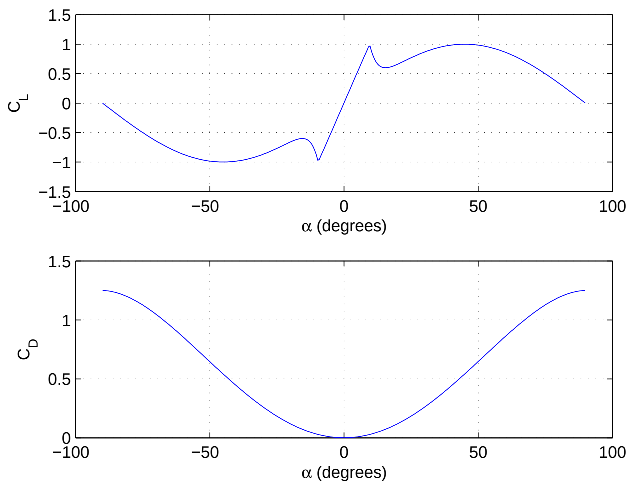


**Figure 2.1:** Lift and drag coefficients versus $\alpha$.

After substituting in the kinematic equation for $\dot{\eta}$, as well as enforcing the quaternion unit norm constraint, the force equations simplify to

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} (\frac{T}{m} + \frac{A_x}{m})(1 - 2\eta_3^2 - 2\eta_4^2) + \frac{A_z}{m}(2\eta_1\eta_3 + 2\eta_2\eta_4) \\ (\frac{T}{m} + \frac{A_x}{m})(2\eta_1\eta_4 + 2\eta_2\eta_3) + \frac{A_z}{m}(2\eta_3\eta_4 - 2\eta_1\eta_2) \\ (\frac{T}{m} + \frac{A_x}{m})(2\eta_2\eta_4 - 2\eta_1\eta_3) + \frac{A_z}{m}(1 - 2\eta_2^2 - 2\eta_3^2) + g \end{pmatrix} \tag{2.9}$$

where $A_x$ and $A_z$ are the combined aerodynamic lift and drag forces given by

$$A_x = \frac{\rho V^2 S}{2}(\sin\alpha C_l(\alpha) - \cos\alpha C_d(\alpha))$$

and

$$A_z = \frac{\rho V^2 S}{2}(-\cos\alpha C_l(\alpha) - \sin\alpha C_d(\alpha)).$$

## 2.2 Two-dimensional Model

In the course of this research, the three-dimensional tailsitter physics model proved difficult to work with in the development of a transition controller. Resulting controllers were overly complex due to the large number of control inputs ($T$, $\eta_1$, $\eta_2$, $\eta_3$, and $\eta_4$) and the nature of these inputs being interspersed throughout the force equations. For the scope and purpose of this thesis, it is convenient to derive a simpler, two-dimensional physics model and develop controllers based upon it. Since the transitions will be performed in the direction of current heading of the tailsitter, the extra dimension is not necessary in any case.

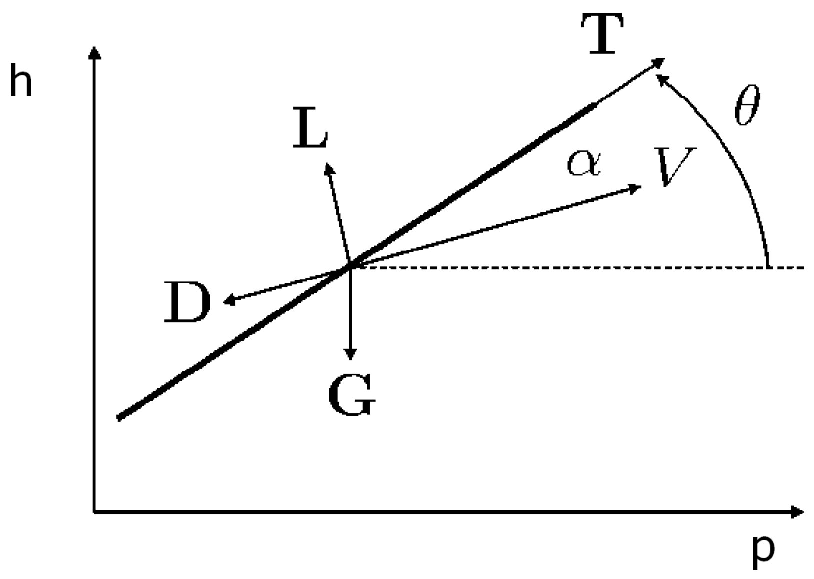


**Figure 2.2:** Forces acting on tailsitter

The forces acting at the tailsitter in a two-dimensional frame are shown in Figure 2.2. The forces are acting on the center of mass of the tailsitter that lies in a two-dimensional inertial coordinate frame. The $p$ dimension is in the direction of the tailsitter's heading in the horizontal plane and represents the distance travelled along the ground-track path. Transitions will always be performed along the $p$ axis, with the nose or belly of the tailsitter always aligned with this axis. The $h$ dimension is the altitude of the tailsitter. A pitch controller is assumed, which provides a moment about the axis perpendicular to both $p$ and $h$. By balancing forces acting on the tailsitter, the equations of motion are given by

$$m \begin{pmatrix} \ddot{p} \\ \ddot{h} \end{pmatrix} = \mathbf{G} + \mathbf{L} + \mathbf{D} + \mathbf{T}$$

and

$$\dot{\theta} = a(\theta^c - \theta)$$

where $\theta$ is the pitch angle, $\mathbf{G}$ is the force due to gravity, $\mathbf{L}$ is the lift force acting on the wing, $\mathbf{D}$ is the drag force acting on the wing, $\mathbf{T}$ is the thrust produced by the motor, and $\theta^c$ is the commanded pitch angle. It is assumed that a pitch controller is available with first-order characteristics described by the positive autopilot constant $a$.

In the inertial frame, the gravity vector is given by

$$\mathbf{G} = \begin{pmatrix} 0 \\ -mg \end{pmatrix}.$$

We will assume that the thrust vector is directed along the tailsitter's body frame $x$ axis. Therefore, in the inertial frame we have

$$\mathbf{T} = R(\theta) \begin{pmatrix} T \\ 0 \end{pmatrix}$$

where $R$ is the rotation matrix between body and inertial frames given by

$$R(\varphi) \triangleq \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}$$

and $T$ is the magnitude of thrust produced by the motor. We will assume that $T > 0$ is an input to the system.

Similarly, the lift and drag vectors are given by

$$\mathbf{L} = R(\theta - \alpha) \begin{pmatrix} 0 \\ L \end{pmatrix}$$

and

$$\mathbf{D} = R(\theta - \alpha) \begin{pmatrix} -D \\ 0 \end{pmatrix}$$

where

$$L = \frac{1}{2}\rho V^2 S C_l(\alpha)$$

is the magnitude of lift and

$$D = \frac{1}{2}\rho V^2 S C_d(\alpha)$$

is the magnitude of drag. Here $C_l(\alpha)$ and $C_d(\alpha)$ are given by Equation 2.7 and 2.8 and are shown in Figure 2.1. Combining the forces due to lift and drag gives

$$\mathbf{L} + \mathbf{D} = \frac{1}{2}\rho V^2 S R(\theta - \alpha) \begin{pmatrix} -C_d(\alpha) \\ C_l(\alpha) \end{pmatrix}.$$

Note that the airspeed is given by

$$V = \sqrt{\dot{p}^2 + \dot{h}^2}$$

and the angle of attack is given by

$$\alpha = \theta - \tan^{-1}\left(\frac{\dot{h}}{\dot{p}}\right).$$

Therefore $\theta - \alpha = \tan^{-1}\left(\frac{\dot{h}}{\dot{p}}\right)$ and

$$\begin{aligned} \cos(\theta - \alpha) &= \cos\left(\tan^{-1}\left(\frac{\dot{h}}{\dot{p}}\right)\right) \\ &= \frac{\dot{p}}{\sqrt{\dot{p}^2 + \dot{h}^2}} \\ &= \frac{\dot{p}}{V} \end{aligned}$$

and

$$\begin{aligned}\sin(\theta-\alpha) &= \sin\left(\tan^{-1}\left(\frac{\dot{h}}{\dot{p}}\right)\right)\\ &= \frac{\dot{h}}{\sqrt{\dot{p}^2+\dot{h}^2}}\\ &= \frac{\dot{h}}{V}.\end{aligned}$$

Therefore

$$\begin{aligned}\mathbf{L}+\mathbf{D} &= \frac{1}{2}\rho V^2 S\begin{pmatrix}\frac{\dot{p}}{V} & -\frac{\dot{h}}{V}\\ \frac{\dot{h}}{V} & \frac{\dot{p}}{V}\end{pmatrix}\begin{pmatrix}-C_d\\ C_l\end{pmatrix}\\ &= \frac{1}{2}\rho V S\begin{pmatrix}-\dot{p}C_d-\dot{h}C_l\\ -\dot{h}C_d+\dot{p}C_l\end{pmatrix}\end{aligned}$$

and the equations of motion are given by

$$\begin{pmatrix}\ddot{p}\\ \ddot{h}\end{pmatrix} = \begin{pmatrix}0\\ -g\end{pmatrix} + \frac{1}{2m}\rho V S\begin{pmatrix}-\dot{p}C_d-\dot{h}C_l\\ -\dot{h}C_d+\dot{p}C_l\end{pmatrix} + R(\theta)\begin{pmatrix}\frac{T}{m}\\ 0\end{pmatrix} \tag{2.10}$$

and

$$\dot{\theta} = a(\theta^c-\theta).$$

## 2.3 Chapter Summary

Chapter 2 has given an overview of the quaternion representation used to describe the tailsitter's attitude. The three-dimensional, quaternion-based dynamics model will be used for simulations of controllers that will be described in later chapters. A simpler, two-dimensional dynamics model was also developed. This model will be used in the derivation of transition controllers. In Chapter 3, description of the experimental setup and underlying navigational controllers will complete the prerequisite discussion necessary before the development of transition controllers in Chapter 4.

# Chapter 3

# Experimental Platform

This chapter describes the simulation and hardware platforms used to test the algorithms derived to track tailsitter transition trajectories. Also, other Magicc Lab research developed for tailsitter attitude control is described. Since the attitude controller is prerequisite to being able to control the tailsitter during transitions with the algorithms described in Chapter 4, it is included for completeness even though it is not the focus of this thesis. Similarly, the navigational state machine autopilot described in Chapter 5 for navigating between hover and level waypoints uses hover control, level control, as well as transition control between the two modes. The hover position controller and level flight controller are therefore briefly explained. Further description of underlying tailsitter attitude and position controllers is found in [21].

## 3.1 Simulation

Each of the transition algorithms were developed and tested first in Matlab with Simulink. The Matlab code was then converted to C and combined with the navigational state machine code to get simulation results of the whole system at work. The full flight path regime of hover and level waypoint following with transitions was included. Doing this allowed the entire simulation to be tested at once. The performance of the transition algorithms could also be seen and evaluated in the context of residing in a larger autopilot system.

## 3.2 Flight Test Setup

The tailsitter test vehicle used for experimentation was the model Pogo airframe shown in Figure 3.1. The Pogo is modeled after the Convair Pogo mentioned in the Introduction. The airframe is available commercially as a radio-controlled model airplane kit. In the original kit purchased by the Magicc Lab, the construction material was thin, tough styrofoam. In subsequent revisions and reworks of the Pogo, corrugated plastic has replaced styrofoam due to its superior durability.

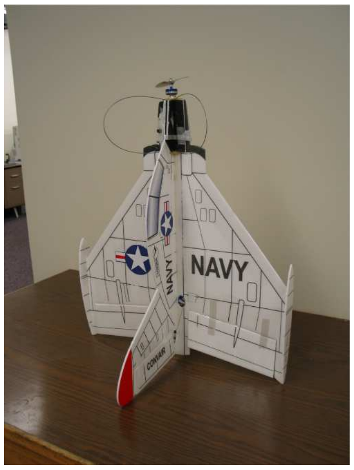


**Figure 3.1:** The Magicc Lab Pogo airframe

Prime characteristics of the Pogo airframe include very large control surfaces and a powerful motor with large propeller attached. The motor and propeller are able to generate a large air flow, or prop-wash over the control surfaces. In a typical fixed-wing aircraft, aerodynamic lift forces are primarily used to keep the vehicle in the air. When the Pogo is in hover flight, the only source of lift is generated by the motor, and therefore the resultant thrust must be very large to keep the aircraft aloft. The control surfaces, actuated by three electric servos typically used by radio-controlled model airplane builders, are very large to maximize potential control in the prop-wash region.

The Kestrel Autopilot version 2.2 shown in Figure 3.2, developed by Procerus Technologies [22], is the heart of the Pogo. The autopilot is lightweight and compact, measuring 5 x 10 x 1 centimeters and weighing 16 grams. The autopilot is equipped with a Rabbit 3100 29MHz microprocessor, on which all autopilot control code is programmed. The autopilot also has a variety of on-board sensors, including three-axis accelerometers and rate gyros, absolute and differential pressure sensors. Ports are available for attaching an external GPS receiver and three-axis magnetometer. With these sensor measurements, the autopilot is able to reasonably estimate the tailsitter's current state, including world frame position, altitude, attitude, and airspeed.

**Figure 3.2:** Kestrel Autopilot

The Kestrel Autopilot is also equipped with a communication link to the ground station control software, Virtual Cockpit, via a 900MHz modem. The Virtual Cockpit software, developed by the BYU Magicc Lab, displays heads-up information about the aircraft's current state. A satellite map of nearby terrain is also displayed, allowing waypoints to be placed at desired locations. The waypoints and other commands are uploaded and data from the aircraft can be downloaded and logged. A bread-crumb trail of the aircraft's trajectory is plotted.

## 3.3 Quaternion Attitude Control

The goal of quaternion attitude control is to adjust the tailsitter's control surfaces to achieve a desired attitude, as represented by the quaternion $\bar{\eta}_d$. To begin with, quaternion multiplication can be defined as

$$\bar{\eta}'' = \bar{\eta}' \otimes \bar{\eta} = \begin{pmatrix} \eta_4 \eta' + \eta_4' \eta - \eta' \times \eta \\ \eta_4' \eta_4 - \eta' \eta \end{pmatrix} \tag{3.1}$$

where $\bar{\eta}''$ is the result of two successive rotations represented by $\bar{\eta}$ and $\bar{\eta}'$ [20]. Equation (3.1) can also be written as

$$\bar{\eta}'' = \bar{\eta}' \otimes \bar{\eta} = \{\bar{\eta}\}_R \bar{\eta}' \tag{3.2}$$

where

$$\{\bar{\eta}\}_R = \begin{pmatrix} \eta_4 & -\eta_3 & \eta_2 & \eta_1 \\ \eta_3 & \eta_4 & -\eta_1 & \eta_2 \\ -\eta_2 & \eta_1 & \eta_4 & \eta_3 \\ -\eta_1 & -\eta_2 & -\eta_3 & \eta_4 \end{pmatrix}.$$

In

$$\bar{\eta}_d = \bar{\eta}_\epsilon \otimes \bar{\eta}_a = \{\bar{\eta}_a\}_R \bar{\eta}_\epsilon \tag{3.3}$$

$\bar{\eta}_a$ represents the actual attitude and $\bar{\eta}_\epsilon$ represents the error between the desired and actual quaternions expressed in the aircraft's body reference frame. Noting that $\{\bar{\eta}\}_R^T \{\bar{\eta}\}_R$ equals the identity matrix, the error quaternion can be written

$$\bar{\eta}_\epsilon = \{\bar{\eta}_a\}_R^T \bar{\eta}_d. \tag{3.4}$$

The error quaternion is conveniently expressed in the aircraft body reference frame. The aileron ($\delta_a$), elevator ($\delta_e$), and rudder ($\delta_r$) can be used to directly control $\eta_\epsilon$ and drive $\Theta_\epsilon$ to zero. Therefore, for the case of zero external disturbances, stable attitude control can be achieved by the PID-like strategy

$$\begin{pmatrix} \delta_a \\ \delta_e \\ \delta_r \end{pmatrix} = k_1 \eta_\epsilon - k_2 \Omega + k_3 \eta_{\epsilon i} \tag{3.5}$$

where $k_1$, $k_2$, and $k_3$ are diagonal gain matrices. In practice these values are gain scheduled by dividing by the current prop-wash, creating larger gains and therefore larger control surface deflections when the prop-wash is low. The $\Omega$ term represents the current angular rates in each direction and provides a dampening effect. The integral of quaternion error, $\eta_{\epsilon i}$, is used in the control to eliminate steady state error.

To provide smoother attitude tracking, the attitude control algorithm is enhanced by introducing a reference model quaternion. The model quaternion tracks the desired quaternion with first-order characteristics. The model quaternion is then used to formulate the error quaternion in Equation 3.4 rather than $\eta_d$. This provides a smoother shift between attitudes.

The transition controllers described in Chapter 4 depend on the existence of an underlying attitude controller. The transition controller algorithms generate desired pitch and heading angles. These values are transformed into a desired quaternion for input into the quaternion attitude controller. Figure 3.3 shows typical performance during flight testing of the quaternion attitude controller. The first four subfigures shows how each element of $\eta$ is tracked. The desired value, reference model value, and actual value for the quaternion parameters are shown. The quaternion error plot is also shown. For perfect tracking, the error quaternion would be

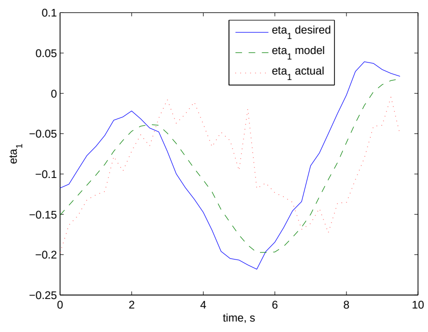
$$\eta_e = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Finally, the desired and actual pitch angle data is extracted from the quaternion information. This allows an easier visualization of the attitude controller's performance.

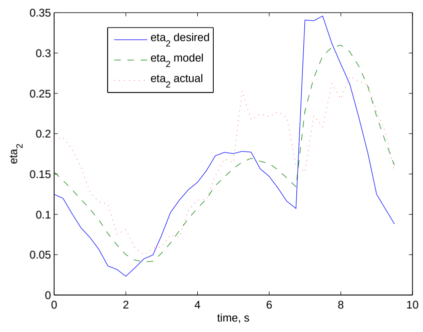
## 3.4 Hover Position Control

Position control is an outer loop of the quaternion attitude controller. A desired quaternion that will maneuver the tailsitter in the direction of the desired hover waypoint is given by

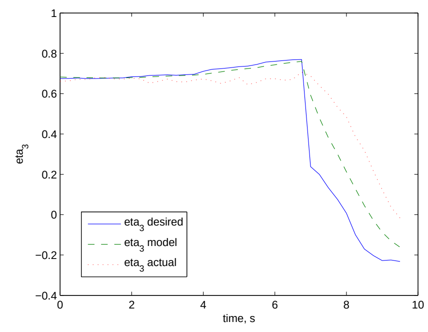
$$\bar{\eta}_d = \bar{\eta}_c \otimes \bar{\eta}_v. \tag{3.6}$$



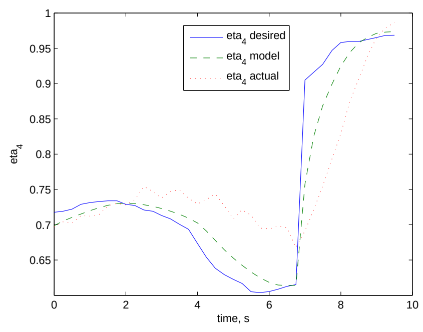
(a) $\eta_1$



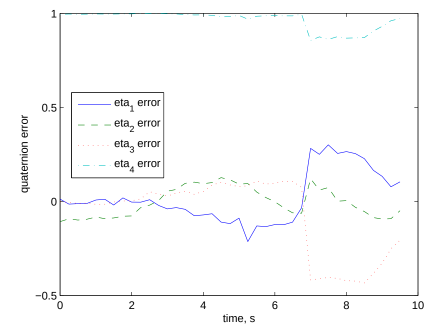
(b) $\eta_2$



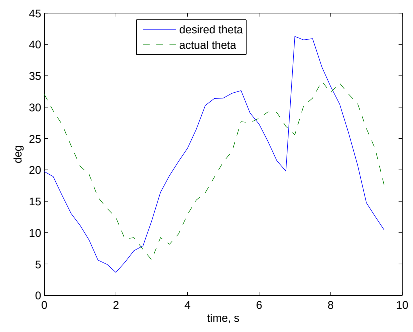
(c) $\eta_3$



(d) $\eta_4$



(e) Error quaternion



(f) $\theta$ tracking

**Figure 3.3:** Quaternion attitude controller performance in flight test

The term

$$\bar{\eta}_v = \begin{pmatrix} 0 \\ \sqrt{2}/2 \\ 0 \\ \sqrt{2}/2 \end{pmatrix}$$

is the vertical quaternion, representing the tailsitter's attitude with the nose of the tailsitter pointing straight up and the belly facing North. In terms of the previously described axes, the nose points along the negative $z$ axis and the belly faces the positive $x$ axis.

The term $\bar{\eta}_c$ is a correction quaternion that describes the rotation needed to tilt the nose of the aircraft in the proper direction for $x$-$y$ position tracking and is given by

$$\bar{\eta}_c = \bar{\eta}_{cv} \otimes \bar{\eta}_{cp} \tag{3.7}$$

where $\eta_{cp}$ is based on position error and $\eta_{cv}$ provides dampening based on body frame velocities. The quaternion parameters $\hat{\eta}_{cp}$ and $\Theta_{cp}$ are given by

$$\hat{\eta}_{cp} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} (x - x_d)/||e_p|| \\ (y - y_d)/||e_p|| \\ 0 \end{pmatrix}$$

and

$$\Theta_{cp} = k_3 ||e_p||$$

where $k_3$ is a gain, $x_d$ and $y_d$ refer to the aircraft desired position, and $||e_p||$ is the norm of position error

$$||e_p|| = \sqrt{(x - x_d)^2 + (y - y_d)^2}.$$

The quaternion components $\hat{\eta}_{cv}$ and $\Theta_{cv}$ are given by

$$\hat{\eta}_{cv} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} \frac{w}{\sqrt{v^2+w^2}} \\ \frac{v}{\sqrt{v^2+w^2}} \\ 0 \end{pmatrix}$$

and

$$\Theta_{cv} = k_4 \sqrt{v^2 + w^2}$$

where $k_4$ is a gain.

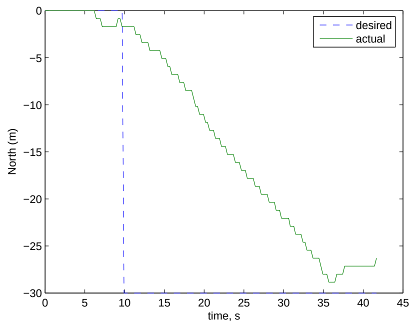
Along with generating a desired quaternion for waypoint tracking, an altitude controller also exists to allow the tailsitter to obtain and hold a given altitude while in a hover position. Since the nose of the tailsitter generally points up in hover flight mode, altitude can be adjusted with the throttle. First, a desired thrust command is generated that would allow the tailsitter to descend or ascend to its commanded altitude. A control loop that adjusts the throttle to match a desired thrust is then used. This same throttle from thrust loop will be used by the transition controllers in Chapter 4, which rely on being able to command and achieve a desired thrust.

In practice, descending is difficult for the tailsitter due to air flowing in the opposing direction over the control surfaces and the decrease in prop-wash due to a decreased throttle setting. It is therefore recommended to descend slowly, in order to keep the throttle setting above some minimum value required to give good control authority. With such considerations in mind, it is possible to control the tailsitter's altitude.

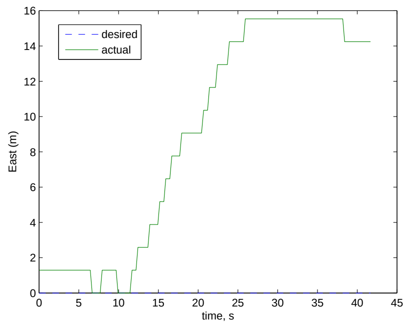
Typical performance of the hover position controller is shown in Figure 3.4. In this experiment, the tailsitter hovers at (0,0) and then is commanded to hover to a point 30 meters to the south. Also, the altitude is controlled by throttle and is set to a desired value of 10 meters. The tailsitter begins on the ground and takes off at about $t = 10$ seconds. It is seen that the tailsitter does move south 30 meters, but also drifts to the east a considerable distance, which is caused by wind. Due its large wing area and lightweight construction materials, the tailsitter is very susceptible to wind, especially while in a hover position. The hover controller can compensate for some wind by tilting the nose of the tailsitter into the wind, but for this flight test this was not enough to prevent drifting in the direction of wind.

## 3.5 Level Flight Control

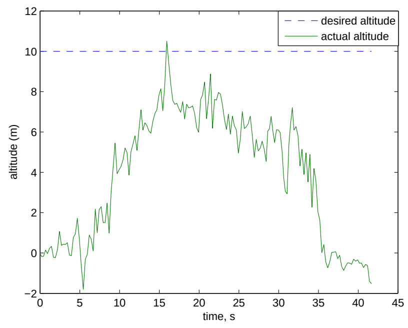
The level flight controller also has the quaternion attitude controller as an inner loop. Desired Euler angles $\theta_c$ and $\psi_c$ are generated and converted to a command quaternion by Equation 2.3. The roll angle, $\phi_c$ is left at zero for level flight control.



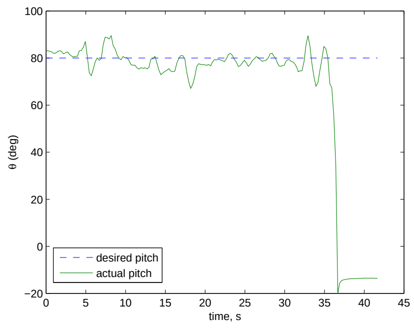
(a) North



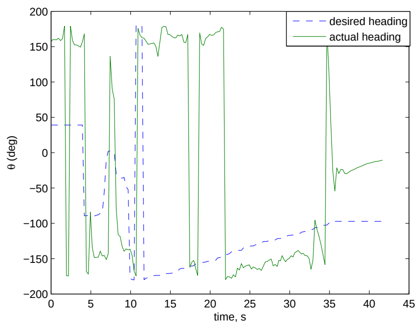
(b) East



(c) Altitude



(d) Pitch



(e) Heading

**Figure 3.4:** Hover position tracking flight data

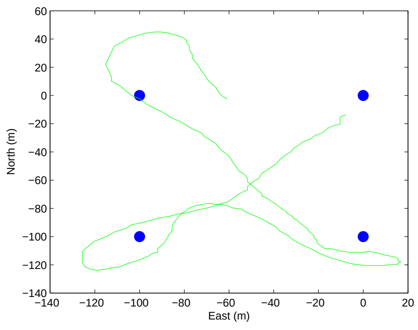
The desired heading angle, $\psi_c$, is generated from vector field path following approach as discussed in [23]. Given the tailsitter's current position, and a desired straight path to follow between two waypoints, a heading angle is generated that will simultaneously return the tailsitter to the path (if deviation has occurred) and point it in the direction of the current waypoint.

Pitch angle, $\theta_c$, and the throttle setting are generated to track desired altitude. When within a window of the desired altitude, the pitch angle is controlled with a feed-forward loop using altitude error, and the throttle is used to maintain airspeed. Above the altitude window, pitch is controlled to maintain airspeed and throttle is set to a designated low setting. Below the altitude window, the pitch-from-airspeed loop is also in effect, but throttle is turned on to full. With this scheme, good altitude tracking in level flight is possible.

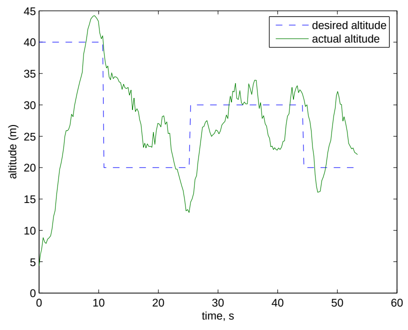
Flight results showing typical performance of the level flight controller is shown in Figure 3.5. In this experiment, an X-shaped path of waypoints was flown. Looking at the North-East GPS waypoint track plot, the tailsitter travels from the Northeast point to the Southwest point, then to the Southeast point and finally the Northwest point. Different altitudes are commanded for each waypoint to demonstrate altitude tracking ability. The tailsitter's altitude controller uses throttle to control airspeed and adjusts the pitch angle to try to achieve the desired altitude.

## 3.6 Chapter Summary

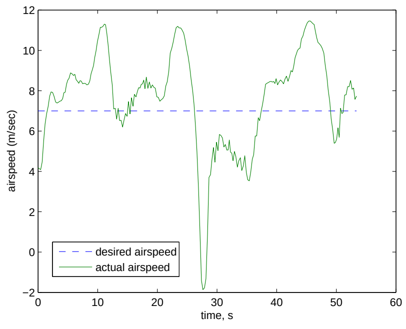
In this chapter, the method for simulation and the experimental setup for flight testing were described. Several underlying controllers were also described. The stage is now set for the derivation of transition algorithms in the next chapter.



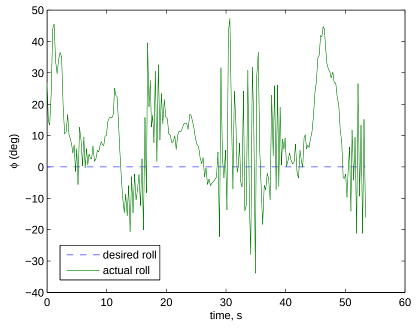
(a) N-E waypoints



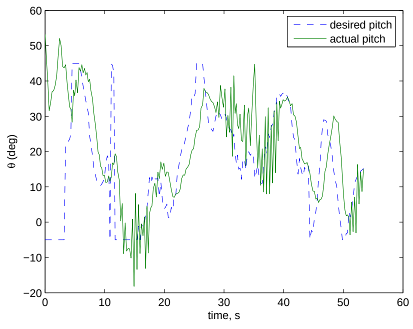
(b) Altitude



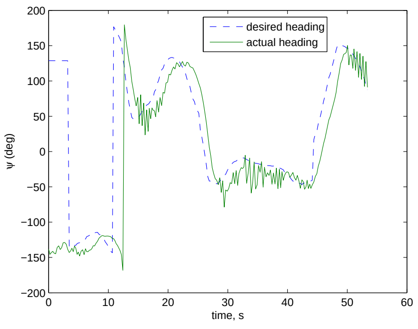
(c) Airspeed



(d) $\phi$



(e) $\theta$



(f) $\psi$

**Figure 3.5:** Level waypoint tracking flight data

# Chapter 4

# Trajectory Tracking Algorithms

In this chapter, a method for creating desired two-dimensional trajectories for tailsitter transitions is presented. Also, three methods for following that trajectory through each type of maneuver (hover-to-level and level-to-hover) are given. Simulation results and actual flight test results are also given for each of the three methods.

## 4.1 Desired Trajectories

The goal of trajectory design is to develop trajectories that will be followed during each type of transition maneuver. The trajectories should be simple and easy to follow. Trajectory design is performed in a two-dimensional world frame. The parameter $p$ represents the distance travelled in a line along the current heading. Altitude is referred to as $h$. The trajectories are all time based, with $t$ being the current time and $t = 0$ at the start of the transition.

In practice, a transition will occur between a level waypoint and a hover waypoint, or vice versa. The straight line path between the two waypoints is the heading along which the maneuver is performed. The point along that path that the maneuver begins is dictated by a higher level command module that is described in Chapter 5.

A desirable transition will guide the tailsitter in between an initial position, $(p_0, h_0)$, and a final desired position, $(p_f, h_f)$. For a hover-to-level transition, the tailsitter will be in a hover position at $(p_0, h_0)$ and be flying level with a constant velocity at $(p_f, h_f)$, as shown in Figure 4.1. For a level-to-hover transition, the tailsitter will be flying level with some initial velocity at $(p_0, h_0)$ and hovering at $(p_f, h_f)$, as shown in Figure 4.2.

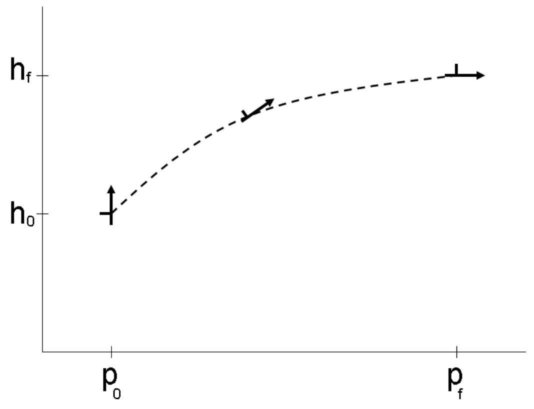


**Figure 4.1:** Hover-to-level trajectory goal

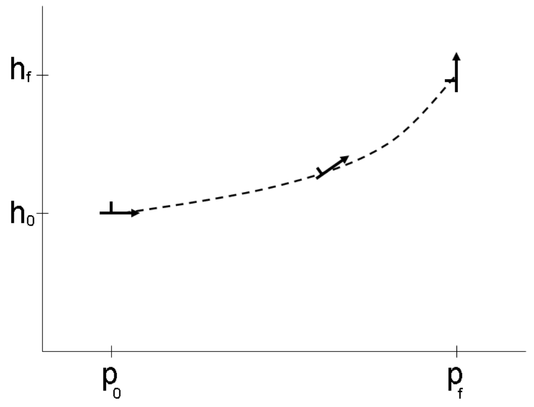


**Figure 4.2:** Level-to-hover trajectory goal

The trajectory generation algorithm generates the quantities $p$, $\dot{p}$, $\ddot{p}$, $h$, $\dot{h}$ and $\ddot{h}$ for accomplishing a desired transition. The inputs to the algorithm are the initial position $(p_0, h_0)$, the desired final position $(p_f, h_f)$, and either the initial velocity $V_0$ for level to hover transitions or the final desired velocity $V_f$ for hover to level transitions. The maneuver time $t_m$, the length of time the maneuver will take, is computed from the other parameters.

Trajectory design is treated independently for both dimensions. The $p$-trajectory is velocity based. For a hover-to-level transition, the tailsitter's velocity will initially be zero and will need to increase to $V_f$ when it is in level flight. For a level-to-hover transition, the velocity will initially be $V_0$ and will then go to zero as the tailsitter assumes a hover position. Trajectories in the $p$ direction are developed with this goal in mind and are given by

$$\ddot{p}_d = \begin{cases} \frac{V_f}{t_m}, & t \leq t_m \\ 0, & \text{otherwise} \end{cases}$$

$$\dot{p}_d = \begin{cases} \frac{V_f}{t_m}t, & t \leq t_m \\ V_f, & \text{otherwise} \end{cases}$$

$$p_d = \begin{cases} \frac{V_f}{2t_m}t^2 + p_0, & t \leq t_m \\ V_f(t - t_m) + V_f\frac{t_m}{2}, & \text{otherwise} \end{cases}$$

where

$$t_m = \frac{2(p_f - p_0)}{V_f}$$

for a hover-to-level transition and

$$\ddot{p}_d = \begin{cases} -\frac{V_0}{t_m}, & t \leq t_m \\ 0, & \text{otherwise} \end{cases}$$

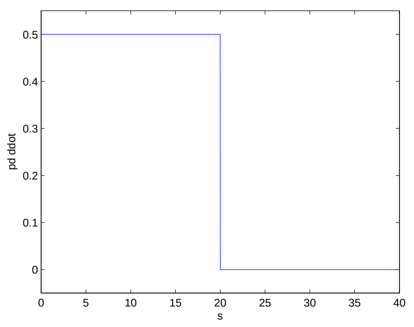
$$\dot{p}_d = \begin{cases} -\frac{V_0}{t_m}t + V_0, & t \leq t_m \\ 0, & \text{otherwise} \end{cases}$$

$$p_d = \begin{cases} -\frac{V_0}{2t_m}t^2 + V_0 t + p_0, & t \leq t_m \\ -\frac{V_0}{2t_m}t_m^2 + V_0 t_m + p_0, & \text{otherwise} \end{cases}$$

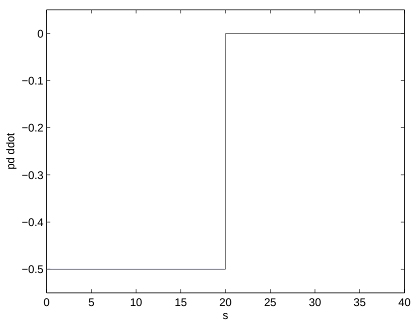
where

$$t_m = \frac{2(p_f - p_0)}{V_0}$$

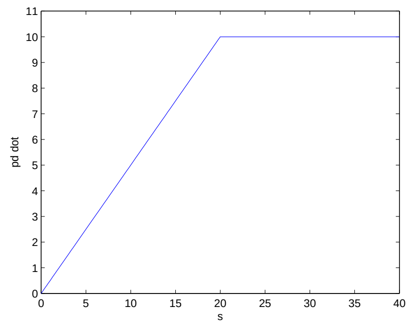
for a level-to-hover transition.



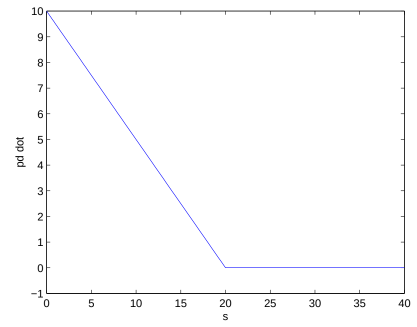
(a) Desired $\ddot{p}$ for hover to level



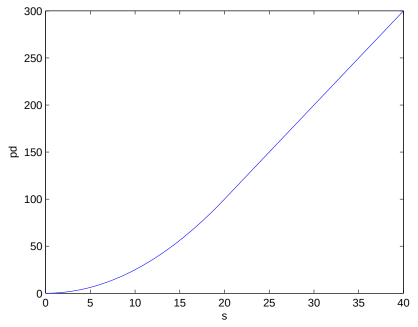
(b) Desired $\ddot{p}$ for level to hover



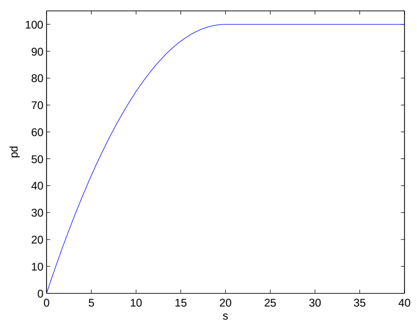
(c) Desired $\dot{p}$ for hover to level



(d) Desired $\dot{p}$ for level to hover



(e) Desired $p$ for hover to level



(f) Desired $p$ for level to hover

**Figure 4.3:** Desired $p$-trajectories

For desired trajectories in the $h$ dimension a different approach is desired. For both transition types, we desire a smooth shift from a constant altitude to another constant altitude. This can be achieved with the use of sigmoid functions. For both hover to level and level to hover transitions, the desired trajectories are given by

$$\begin{aligned} h_d &= \frac{h_f - h_0}{1 + e^{-k(t - \frac{t_m}{2})}} + h_0, \\ \dot{h}_d &= k(h_f - h_0)\frac{e^{kt + (\frac{t_m}{2})k}}{(e^{kt} + e^{t_m \frac{k}{2}})^2}, \end{aligned}$$

and

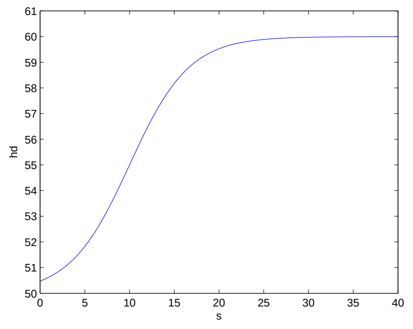
$$\ddot{h}_d = \frac{-k^2(h_f - h_0)(e^{kt} - e^{t_m \frac{k}{2}})e^{kt + t_m \frac{k}{2}}}{(e^{kt} + e^{t_m \frac{k}{2}})^3}.$$

In these equations, $k$ is a constant that determines how quickly the desired altitude trajectory curves arrive at their final value. The length of time of the maneuver is determined by the distance along the path, from $p_0$ to $p_f$, and is calculated in the discussion on desired $p$ trajectories. Once the value of $t_m$ is determined, it is used in sigmoid functions to develop smooth $h$ trajectories.

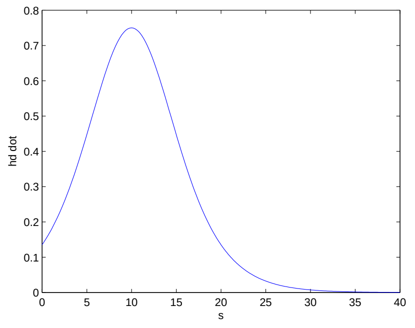
The desired $p$-trajectories are shown in Figure 4.3. Desired $h$-trajectories are shown in Figure 4.4. For both maneuvers, the input parameters are $(p_0, h_0) = (0, 50)$ and $(p_f, h_f) = (100, 60)$.

## 4.2 Simple Controller

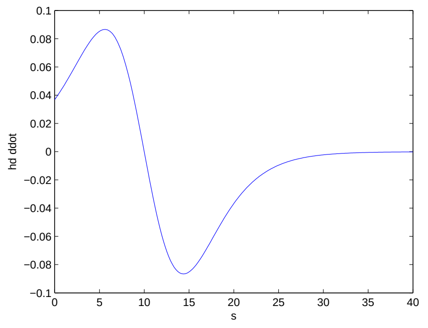
It is desirable to develop a controller that will successfully follow the trajectories generated in Section 4.1. Sections 4.3 and 4.4 both describe methods that will follow the trajectories. The controller described in this section, however, will successfully perform a transition, but only follows the desired $h$ trajectory. This type of transition is useful when the lateral distance travelled is not as important as per-



(a) Desired $h$ for both transition types



(b) Desired $\dot{h}$ for both transition types



(c) Desired $\ddot{h}$ for both transition types

**Figure 4.4:** Desired $h$-trajectories

forming a transition with little altitude deviation. For instance, if the tailsitter were equipped with some sensor that could detect a nearby wall directly in the level flight path, a quick level-to-hover transition could be commanded. Furthermore, the development of a simple controller will provide a baseline for judging performance and other issues with the other two, more complex controllers.

When the maneuver begins with the simple controller, a desired quaternion is generated from Equation 2.3 with $\phi = 0$, $\psi$ being the heading between the previous and current waypoint, and $\theta$ being zero for a hover-to-level transition or ninety degrees for a level-to-hover transition. This is used as the desired quaternion in the quaternion attitude controller described in Section 3.3 and in [21]. While the tailsitter is making the transition, the throttle from altitude control loop is enabled, allowing desired altitude to be tracked. This control loop generates a desired thrust to follow to achieve a desired altitude. The desired thrust is tracked by adjusting the throttle.

## 4.3 Feedback Linearization Controller

Feedback linearization is a technique of controlling nonlinear systems by transforming them into an equivalent linear system [24]. Fortunately for the case of tailsitter dynamics, Equation 2.10 is already in normal form, with the control inputs $T$ and $\theta$ separated from the nonlinearities. The feedback linearization controller relies on knowledge of the aerodynamic model to derive a controller to track both $p$ and $h$ trajectories through transition maneuvers. First, define position error as

$$\tilde{p} = p - p_d$$

and

$$\tilde{h} = h - h_d.$$

Then acceleration error is

$$\begin{aligned}\begin{pmatrix} \ddot{\tilde{p}} \\ \ddot{\tilde{h}} \end{pmatrix} &= \begin{pmatrix} \ddot{p} - \ddot{p}_d \\ \ddot{h} - \ddot{h}_d \end{pmatrix} \\ &= \begin{pmatrix} \frac{\rho S}{2m}\sqrt{\dot{p}^2 + \dot{h}^2}(-\dot{p}C_d - \dot{h}C_l) + \frac{T}{m}\cos\theta - \ddot{p}_d \\ -g + \frac{\rho S}{2m}\sqrt{\dot{p}^2 + \dot{h}^2}(-\dot{h}C_d + \dot{p}C_l) + \frac{T}{m}\sin\theta - \ddot{h}_d \end{pmatrix}.\end{aligned}$$

The control input we will define as

$$U \triangleq \begin{pmatrix} \frac{T}{m}\cos\theta \\ \frac{T}{m}\sin\theta \end{pmatrix} = U_1 + U_2$$

where $U_1$ will be used to cancel nonlinearities from the system and $U_2$ is a pseudo-input that will be selected to drive tracking error to zero. By selecting $U_1$ as

$$U_1 = \begin{pmatrix} -\frac{\rho S}{2m}\sqrt{\dot{p}^2+\dot{h}^2}(-\dot{p}C_d - \dot{h}C_l) \\ g - \frac{\rho S}{2m}\sqrt{\dot{p}^2+\dot{h}^2}(-\dot{h}C_d + \dot{p}C_l) \end{pmatrix}$$

the system is linearized and now becomes

$$\begin{pmatrix} \ddot{\tilde{p}} \\ \ddot{\tilde{h}} \end{pmatrix} = U_2 - \begin{pmatrix} \ddot{p}_d \\ \ddot{h}_d \end{pmatrix}.$$

To track the desired $p$ and $h$ trajectories with second order characteristics, it is desirable for

$$\ddot{\tilde{p}} = -k_{dp}\dot{\tilde{p}} - k_{pp}\tilde{p}$$

and

$$\ddot{\tilde{h}} = -k_{dh}\dot{\tilde{h}} - k_{ph}\tilde{h}$$

where $k_{dp}$, $k_{pp}$, $k_{dh}$, and $k_{ph}$ are tunable gains.

To achieve this, let

$$U_2 = \begin{pmatrix} -k_{dp}\dot{\tilde{p}} - k_{pp}\tilde{p} + \ddot{p}_d \\ -k_{dh}\dot{\tilde{h}} - k_{ph}\tilde{h} + \ddot{h}_d \end{pmatrix}.$$

The control input is then

$$\begin{pmatrix} \frac{T}{m}\cos\theta \\ \frac{T}{m}\sin\theta \end{pmatrix} = U_1 + U_2 = \begin{pmatrix} -\frac{\rho S}{2m}\sqrt{\dot{p}^2+\dot{h}^2}(-\dot{p}C_d - \dot{h}C_l) - k_{dp}\dot{\tilde{p}} - k_{pp}\tilde{p} + \ddot{p}_d \\ g - \frac{\rho S}{2m}\sqrt{\dot{p}^2+\dot{h}^2}(-\dot{h}C_d + \dot{p}C_l) - k_{dh}\dot{\tilde{h}} - k_{ph}\tilde{h} + \ddot{h}_d \end{pmatrix}.$$

To be able to eventually command thrust values and pitch commands, it is necessary get independent expressions for $T$ and $\theta$. We find it convenient to redefine the rows of the input vector as

$$F_1 \triangleq -\frac{\rho S}{2m}\sqrt{\dot{p}^2 + \dot{h}^2}(-\dot{p}C_d - \dot{h}C_l) - k_{dp}\dot{\tilde{p}} - k_{pp}\tilde{p} + \ddot{p}_d$$

and

$$F_2 \triangleq g - \frac{\rho S}{2m}\sqrt{\dot{p}^2 + \dot{h}^2}(-\dot{h}C_d + \dot{p}C_l) - k_{dh}\dot{\tilde{h}} - k_{ph}\tilde{h} + \ddot{h}_d$$

resulting in the expressions

$$\frac{T}{m}\cos\theta = F_1 \tag{4.1}$$

and

$$\frac{T}{m}\sin\theta = F_2. \tag{4.2}$$

By squaring both of these equations and adding the results together, we have

$$\frac{T^2}{m^2} = F_1^2 + F_2^2$$

and therefore the value for thrust is

$$T = m\sqrt{F_1^2 + F_2^2}.$$

If we divide Equation 4.2 by Equation 4.1,

$$\frac{\frac{T}{m}\sin\theta}{\frac{T}{m}\cos\theta} = \frac{F_2}{F_1}$$

and

$$\tan\theta = \frac{F_2}{F_1}$$

so

$$\theta = \tan^{-1}\frac{F_2}{F_1}.$$

We now have command values for thrust and $\theta$. In practice, the commanded pitch value along with the heading angle along which the maneuver is to be performed is converted to a quaternion using Equation 2.3. This quaternion is then used as the desired quaternion in the quaternion attitude controller from Section 3.3. The throttle command $T$ is fed to the throttle from the thrust feedback control loop, which adjusts the throttle setting to effect a desired thrust.

The feedback linearization controller requires knowledge of several parameters that are not known or measured accurately on the tailsitter. These parameters primarily include $C_l$ and $C_d$, which are given by Equations 2.7 and 2.8. This controller's performance is therefore expected to be greatly affected by how well the true values of these parameters are known.

## 4.4 Adaptive Controller

In the feedback linearization example, several parameters that are not typically known were made available to the controller. In this section, a model reference adaptive controller (MRAC) is described which requires knowledge of only $m$, $g$, state information $p$, $\dot{p}$, $h$ and $\dot{h}$, and desired trajectories for $p_d$ and $h_d$. We will then eliminate the need to require knowledge of $C_l$ and $C_d$ in order to successfully track a desired trajectory.

### 4.4.1 Lyapunov Stability

Derivation of the adaptive control method relies on Lyapunov stability theory, discussed in [24], which provides conditions to prove a system's stability. First a Lyapunov function $V(x)$ is chosen, where $x$ denotes the state variables that need to be driven to zero. In the case to be described below, $x$ is the trajectory tracking error, that we would like to drive to zero. The choice of $V(x)$ must adhere to the following rules:

1. $V(0) = 0$,

2. $V(x) > 0$, or in other words, $V(x)$ is positive definite,

3. $V(x)$ is continuously differentiable.

If $V(x)$ meets these criteria and $\dot{V}(x) < 0$ for $x \neq 0$, or in other words, $\dot{V}(x)$ is negative definite, then $x \to 0$ asymptotically.

The adaptive control method described in this section will develop a Lyapunov function based on the error in tracking desired trajectories. Then, with the addition of a proper parameter estimation scheme, it will be shown that the derivative of the Lyapunov function is negative definite and therefore the error in trajectory tracking will go to zero.

### 4.4.2 Equations of Motion

We rewrite the two-dimensional tailsitter equations of motion from Equation 2.10 as

$$\begin{pmatrix} \ddot{p} \\ \ddot{h} \end{pmatrix} = \upsilon \begin{pmatrix} -A & -B \\ B & -A \end{pmatrix} \begin{pmatrix} \dot{p} \\ \dot{h} \end{pmatrix} + \begin{pmatrix} 0 \\ -g \end{pmatrix} + \begin{pmatrix} \frac{T}{m}\cos\theta \\ \frac{T}{m}\sin\theta \end{pmatrix}$$

or

$$\ddot{X} = \upsilon \Upsilon \dot{X} + G + U \tag{4.3}$$

where $\upsilon = \sqrt{\dot{p}^2 + \dot{h}^2}$ is airspeed. The true values of parameters that will be estimated are

$$A \triangleq \frac{\rho S}{2m} C_d$$

and

$$B \triangleq \frac{\rho S}{2m} C_l.$$

### 4.4.3 Reference Model

The reference model is a second order model that tracks the commanded position values. The reference model evolves according to

$$\begin{pmatrix} \ddot{p}_m \\ \ddot{h}_m \end{pmatrix} = - \begin{pmatrix} \alpha_{1p} & 0 \\ 0 & \alpha_{1h} \end{pmatrix} \begin{pmatrix} \dot{p}_m \\ \dot{h}_m \end{pmatrix} - \begin{pmatrix} \alpha_{2p} & 0 \\ 0 & \alpha_{2h} \end{pmatrix} \begin{pmatrix} p_m \\ h_m \end{pmatrix} + \begin{pmatrix} \alpha_{2p} & 0 \\ 0 & \alpha_{2h} \end{pmatrix} \begin{pmatrix} p_d \\ h_d \end{pmatrix}$$

or

$$\ddot{R} = -\alpha_1 \dot{R} - \alpha_2 R + \alpha_2 R_c \tag{4.4}$$

where $\alpha$ values are tunable gains and $p_d$, $h_d$ are desired trajectory values.

### 4.4.4 Controller Derivation

The adaptive controller is given in Theorem 4.1. The proof is used to show that with adaptive parameter estimation, the position error

$$\tilde{X} \overset{\triangle}{=} X - R$$

and the velocity error

$$\dot{\tilde{X}} = \dot{X} - \dot{R}$$

are both asymptotically stable.

**Theorem 4.1** *If the estimates of $A$ and $B$, called $\hat{A}$ and $\hat{B}$, are updated according to*

$$\dot{\hat{A}} = \gamma_1 \sqrt{\dot{p}^2 + \dot{h}^2}(-\dot{p}\dot{\tilde{p}} - \dot{h}\dot{\tilde{h}})$$

*and*

$$\dot{\hat{B}} = \gamma_2 \sqrt{\dot{p}^2 + \dot{h}^2}(-\dot{h}\dot{\tilde{p}} + \dot{p}\dot{\tilde{h}})$$

*where $\gamma_1$ and $\gamma_2$ are gains, and the control input is given as*

$$U = -k\dot{\tilde{X}} - \tilde{X} - \upsilon\hat{\Upsilon}\dot{X} - G - \alpha_1 \dot{R} - \alpha_2 R + \alpha_2 R_c$$

*where the reference model propagates according to Equation 4.4, then*

$$\dot{\tilde{X}} \to 0$$

*and*

$$\tilde{X} \to 0$$

*asymptotically.*

*Proof:* Consider the Lyapunov function candidate

$$V = \frac{1}{2}\tilde{X}^T\tilde{X} + \frac{1}{2}\dot{\tilde{X}}^T\dot{\tilde{X}} + \frac{1}{2\gamma}\tilde{\xi}^T\tilde{\xi} \tag{4.5}$$

where

$$\tilde{\xi} = \xi - \hat{\xi}$$

is the difference between actual and estimated $A$ and $B$ parameters and is given by

$$\tilde{\xi} = \begin{pmatrix} \tilde{A} \\ \tilde{B} \end{pmatrix} = \begin{pmatrix} A \\ B \end{pmatrix} - \begin{pmatrix} \hat{A} \\ \hat{B} \end{pmatrix}$$

and $\gamma$ is the vector of adaptive control gains

$$\gamma = \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix}.$$

The time derivative of (4.5) can be shown to be

$$\dot{V} = \dot{\tilde{X}}^T\tilde{X} + \dot{\tilde{X}}^T\ddot{\tilde{X}} + \frac{1}{\gamma}\dot{\tilde{\xi}}^T\tilde{\xi} \tag{4.6}$$

where

$$\ddot{\tilde{X}} = \ddot{X} - \ddot{R}.$$

Replacing $\ddot{\tilde{X}}$ in Equation 4.6 with $\ddot{X}$ from Equation 4.3 and $\ddot{R}$ from Equation 4.4 results in

$$\dot{V} = \dot{\tilde{X}}^T(\tilde{X} + \upsilon\Upsilon\dot{X} + G + U + \alpha_1\dot{R} + \alpha_2 R - \alpha_2 R_c) + \frac{1}{\gamma}\dot{\tilde{\xi}}^T\tilde{\xi}.$$

Substituting $U$ with the expression given in the statement of Theorem 4.1 gives

$$\dot{V} = -k\dot{\tilde{X}}^T\dot{\tilde{X}} + \upsilon\dot{\tilde{X}}^T\tilde{\Upsilon}\dot{X} + \frac{1}{\gamma}\dot{\tilde{\xi}}^T\dot{\tilde{\xi}} \tag{4.7}$$

where

$$\tilde{\Upsilon} \triangleq \Upsilon - \hat{\Upsilon}$$

is the difference between actual and estimated parameters $A$ and $B$. Equation 4.7 may be rewritten as

$$\dot{V} = -k\dot{\tilde{X}}^T\dot{\tilde{X}} + \upsilon\dot{\tilde{X}}^T Z\tilde{\xi} + \frac{1}{\gamma}\dot{\tilde{\xi}}^T\tilde{\xi}$$

where

$$Z \triangleq \begin{pmatrix} -\dot{p} & -\dot{h} \\ -\dot{h} & \dot{p} \end{pmatrix}$$

and

$$\dot{\tilde{\xi}} = \dot{\xi} - \dot{\hat{\xi}}.$$

We assume that $\xi$ is slowing varying enough that $\dot{\xi}$ may be treated as zero. Then the Lyapunov function derivative becomes

$$\dot{V} = -k\dot{\tilde{X}}^T\dot{\tilde{X}} + \upsilon\dot{\tilde{X}}^T Z\tilde{\xi} + \frac{1}{\gamma}\dot{\hat{\xi}}^T\tilde{\xi}.$$

By choosing the adaptive parameter update law as given in the statement of Theorem 4.1, or in other terms

$$\dot{\hat{\xi}}^T = \gamma\upsilon\dot{\tilde{X}}^T Z$$

then

$$\dot{V} = -k\dot{\tilde{X}}^T\dot{\tilde{X}}. \tag{4.8}$$

Since $\dot{V}$ is negative definite, $\dot{\tilde{X}} \to 0$ by the theory of Lyapunov. LaSalle's Invariance Principle [24] will be used to show that $\tilde{X} \to 0$.

Let the set $E$ be defined as

$$E \triangleq \left\{ \begin{pmatrix} \tilde{X} \\ \dot{\tilde{X}} \\ \tilde{\xi} \end{pmatrix} : \dot{V} = 0 \right\}.$$

The derivative of the Lyapunov function is only 0 where $\dot{\tilde{X}} = 0$, so

$$E = \left\{ \begin{pmatrix} \tilde{X} \\ \dot{\tilde{X}} \\ \tilde{\xi} \end{pmatrix} : \dot{\tilde{X}} = 0 \right\}.$$

Let $M$ be the largest invariant set in $E$. Then

$$\begin{pmatrix} \tilde{X} \\ \dot{\tilde{X}} \\ \tilde{\xi} \end{pmatrix} \in M \Rightarrow \dot{\tilde{X}}(t) = 0 \quad \forall\ t.$$

Therefore if $\tilde{X}(0) = 0$, or in other words if the reference model begins as the current value of $X$, then

$$\tilde{X}(t) = 0 \quad \forall\ t$$

and $\tilde{X}$ is asymptotically stable. ■

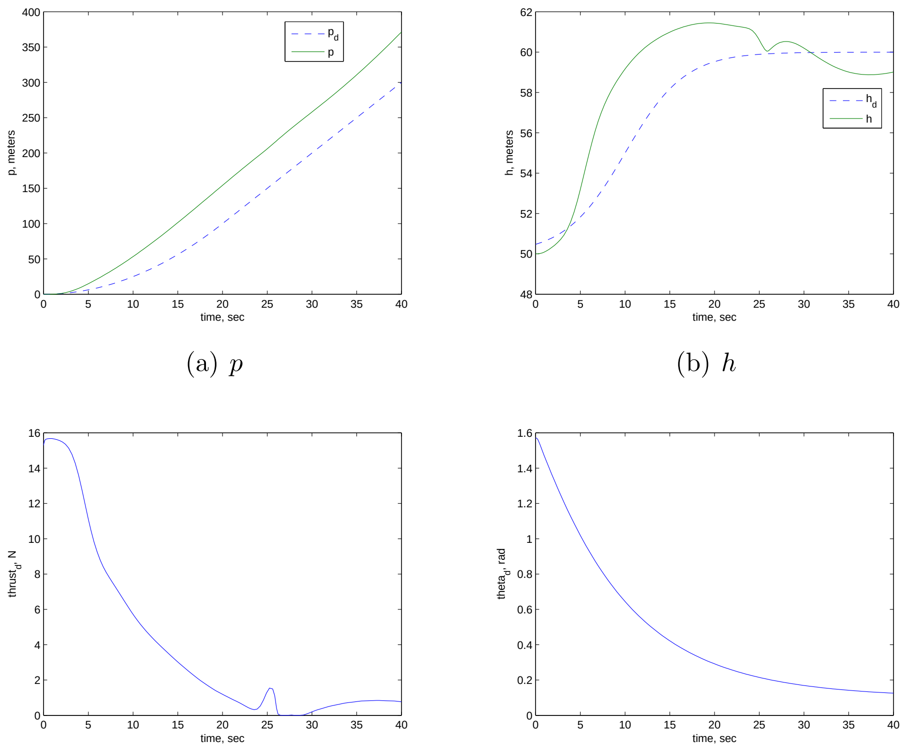
Although using an adaptive controller is beneficial in that knowledge of system parameters is not required, a host of new gains used in the algorithm are introduced which must be then be tuned primarily by trial and error. In general, each of these gains has a specific effect. The gain $\alpha$ determines how closely the reference model tracks the desired trajectory, $\gamma$ controls how fast the estimated parameters adapt, and $k$ determines how fast the actual values converge to the reference model.

## 4.5 Simulation Results

Each of the controllers was simulated in Matlab Simulink for both a hover-to-level transition and a level-to-hover transition. The physics model used in the simulations is described in Section 2.1.

### 4.5.1 Simple Controller Simulations

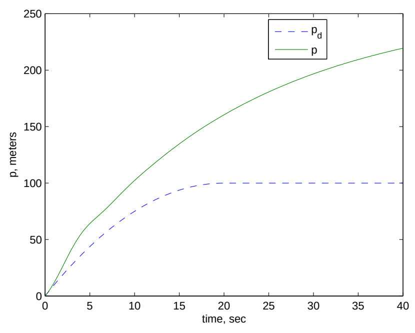
The results from the simple controller simulation are seen in Figure 4.5 for hover-to-level and Figure 4.6 for level-to- hover. As described in Section 4.2, no effort is being made to track the $p$ trajectory. Although there are some deviations in tracking the $h$ trajectory, the controller approaches the final desired altitude for both transition types. This controller's primary benefit is its simplicity. When a transition is desired but trajectory tracking is not a high priority, then the simple controller should perform nicely.



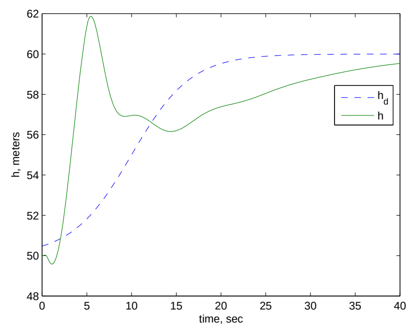
(a) $p$ (b) $h$

(c) thrust (d) $\theta$

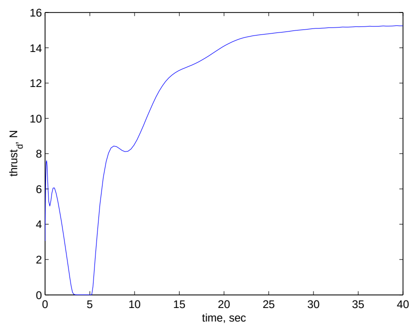
**Figure 4.5:** Simple controller hover-to-level trajectory tracking in simulation



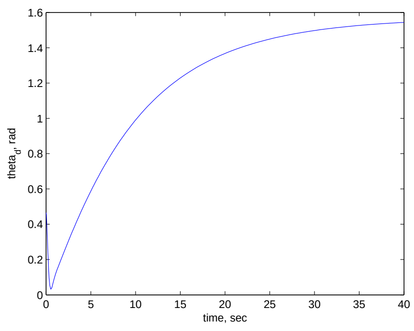
(a) $p$



(b) $h$



(c) thrust

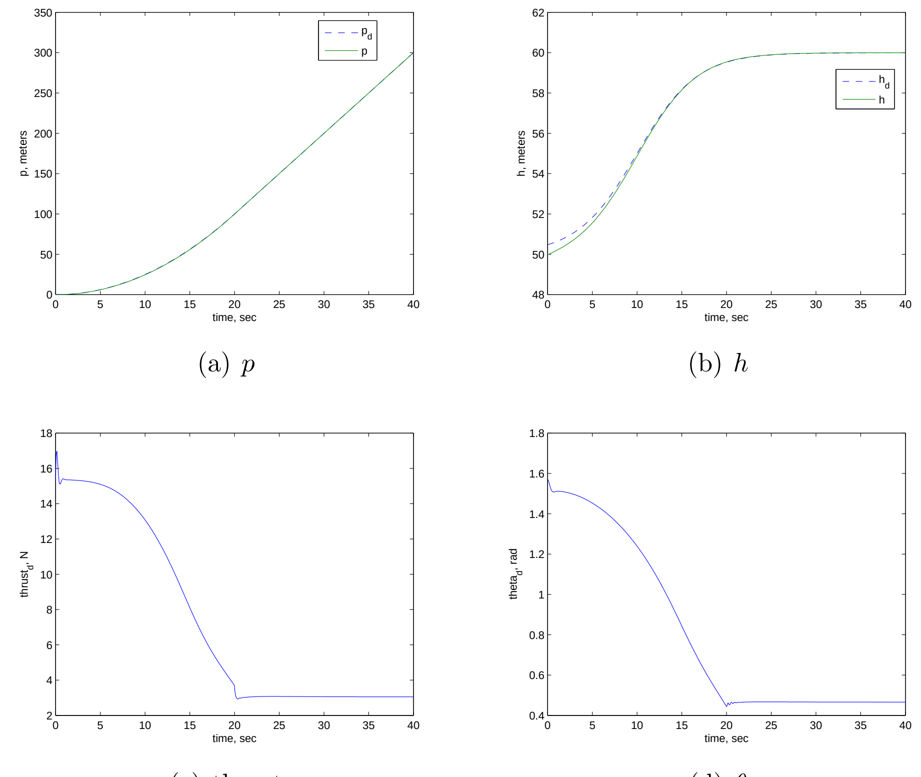


(d) $\theta$

**Figure 4.6:** Simple controller level-to-hover trajectory tracking in simulation

### 4.5.2 Feedback Linearization Controller Simulations

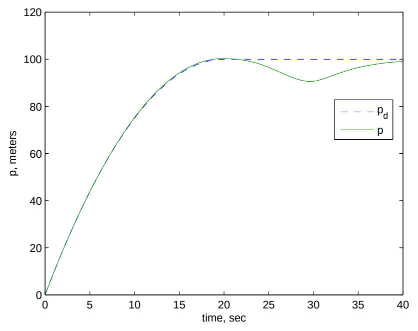
Figure 4.7 shows the results of the feedback linearization controller simulation for a hover-to-level transition. Level-to-hover transition results are shown in Figure 4.8. As is seen, the feedback linearization controller works very well for tracking both trajectories during the entire course of the maneuver. During the level-to-hover transition, it is seen that around $t = 30$ seconds a slight deviation in $p$ tracking occurs. To restore the tailsitter to the proper trajectory, a corresponding dip in pitch can be seen. The controller pitches down the tailsitter in order to gain velocity in the $p$ direction and correct the deviation. When the tailsitter is once again tracking, the tailsitter pitches back up to a vertical position.



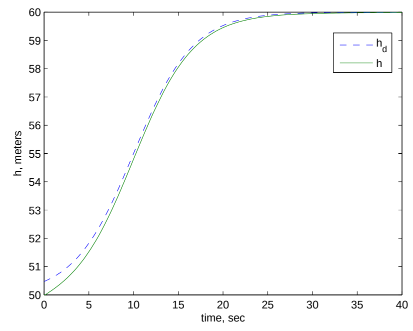
(a) $p$ (b) $h$

(c) thrust (d) $\theta$

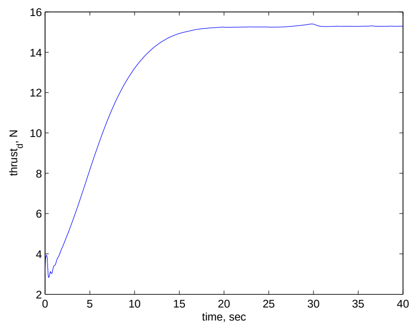
**Figure 4.7:** Feedback linearization controller hover-to-level trajectory tracking in simulation



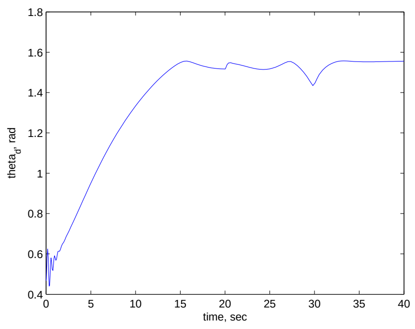
(a) $p$



(b) $h$



(c) thrust



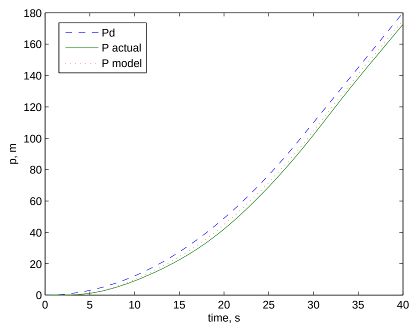
(d) $\theta$

**Figure 4.8:** Feedback linearization controller level-to-hover trajectory tracking in simulation

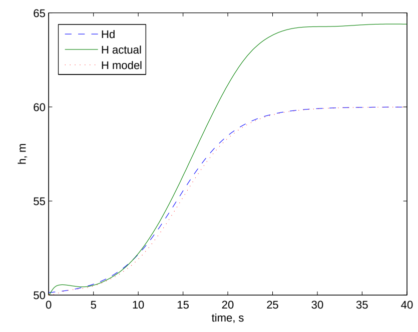
### 4.5.3 Adaptive Controller Simulations

Adaptive controller simulation results are seen in Figure 4.9 and Figure 4.10 for hover-to-level transition results and level-to-hover transition results, respectively. Simulation results show favorable trajectory tracking for both $p$ and $h$ dimensions with a maximum deviation of about 7 $m$ in $p$ and 4 $m$ in $h$. The adaptively estimated parameters $\hat{A}$ and $\hat{B}$ are also plotted and can be seen to adapt to changing conditions as the tailsitter flies through its transitions. The true values of $A$ and $B$ are also plotted. The estimated parameters are not guaranteed to track the actual values of these parameters and are not seen to do so, although the estimates do generally move in the same direction as the actual values.

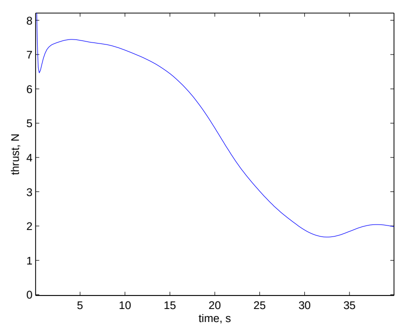
During the first moments of the level to hover transition, there is poor tracking as the estimated parameters change rapidly. The result is a large spike in both desired thrust and desired pitch angle. Once the adaptive algorithm has been running for a few iterations, the parameters stabilize resulting in a smoother transition in thrust and pitch angle. This effect will be minimized in actual implementation by choosing gains appropriately.



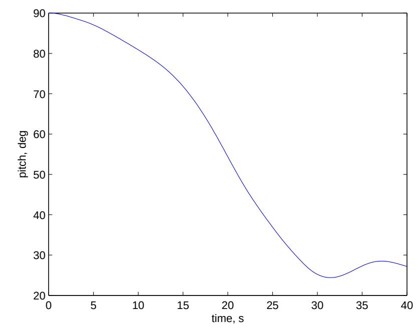
(a) $p$



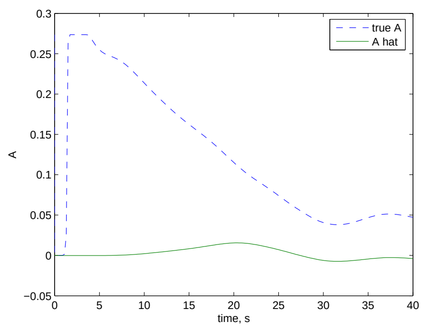
(b) $h$



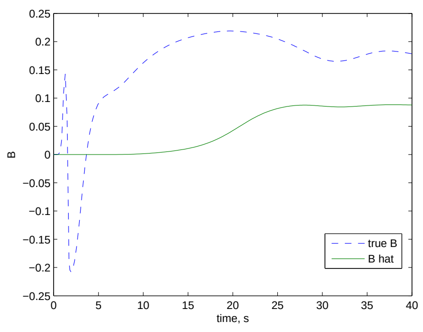
(c) thrust



(d) $\theta$

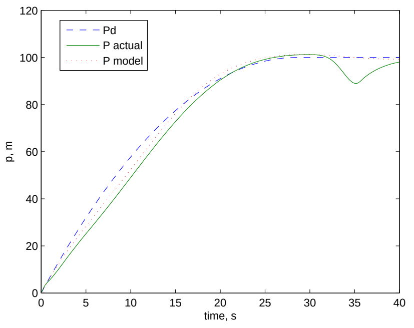


(e) $\hat{A}$

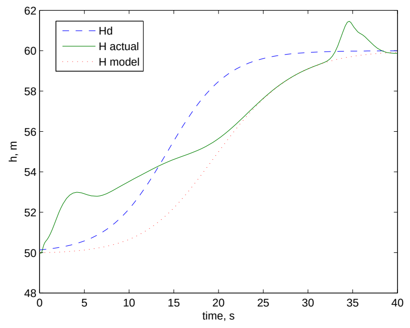


(f) $\hat{B}$

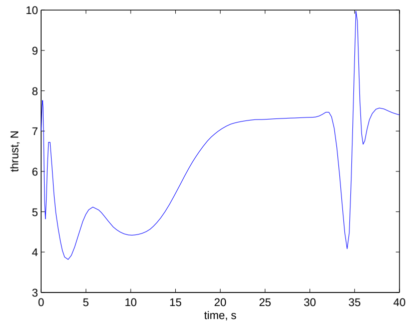
**Figure 4.9:** Adaptive controller hover-to-level trajectory tracking in simulation



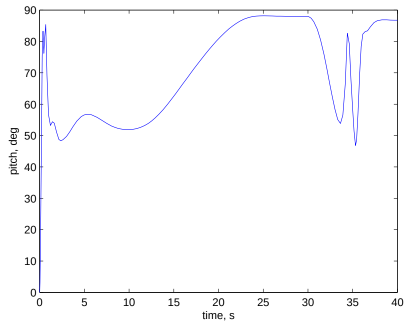
(a) $p$



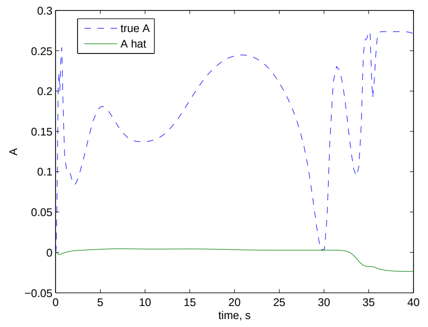
(b) $h$



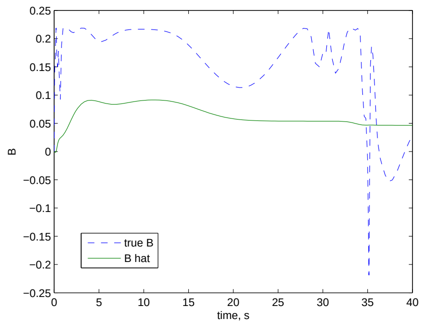
(c) thrust



(d) $\theta$



(e) $\hat{A}$



(f) $\hat{B}$

**Figure 4.10:** Adaptive controller level-to-hover trajectory tracking in simulation

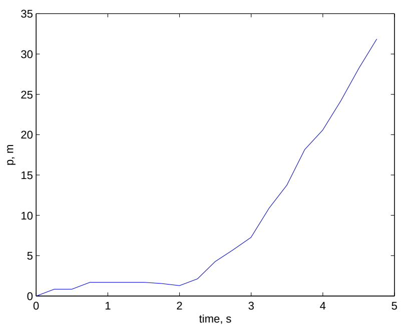
## 4.6 Flight Test Results

Results from flight testing show the algorithms working, but performance is only satisfactory. Trajectory tracking is not as good as simulation results. This is attributable to a number of factors. Primarily, the reality of sensor noise and lag hurts performance the most. In particular, GPS, which is used to measure position and velocities, is known to be inaccurate as well as time delayed. Also, GPS updates occur only a few times per second. Performance would undoubtedly be improved if GPS readings were improved and the frequency of updates increased.

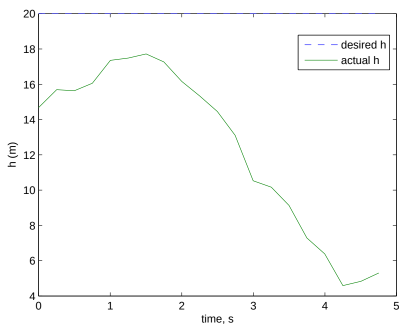
Along with sub-par sensors, the other factors contributing to deviations in the flight results are the characteristics of the underlying controllers described in Chapter 3. Both feedback linearization and adaptive controllers depend on being able to command a desired pitch angle and desired thrust command. The quaternion attitude controller and thrust controller are not able to perfectly match the desired commands. Improvements to trajectory tracking through transitions will be seen as these controllers are improved.

### 4.6.1 Simple Controller Flight Results

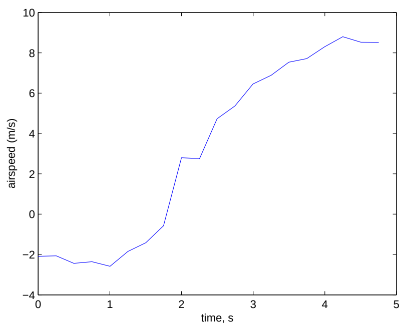
In actual flight on the tailsitter, the simple controller is able to complete the desired transitions very quickly. For both types of maneuvers, however, altitude deviation is very large, approaching 10 meters or more. For the hover-to-level transition, shown in Figure 4.11, a drop in altitude is seen while for the level-to-hover transition, shown in Figure 4.12, altitude increases by about this same amount. During the transitions, the altitude hold loop is enabled, which uses the throttle to achieve a thrust that should balance the weight of the tailsitter while in a hover position. For the time the tailsitter is taken out of a hover position by the simple controller, the altitude controller has difficulty maintaining altitude with just the throttle. Also, limitations of the thrust tracking controller must also be taken into account to explain the evident altitude deviations.



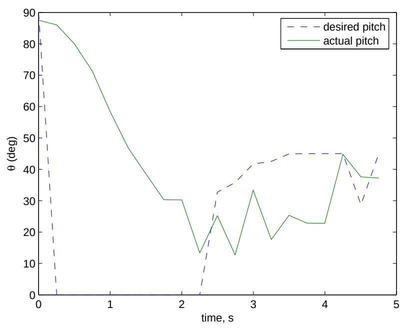
(a) $p$



(b) $h$

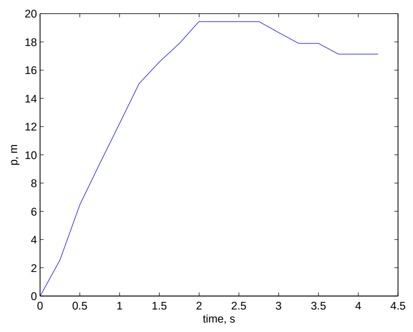


(c) Airspeed

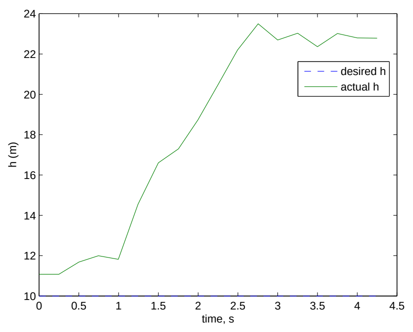


(d) $\theta$

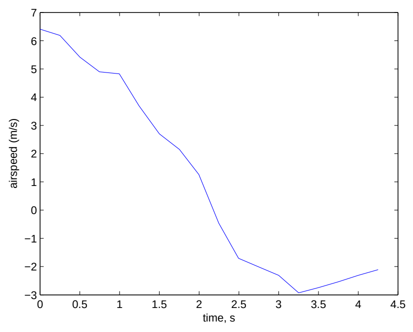
**Figure 4.11:** Simple controller hover-to-level transition flight data



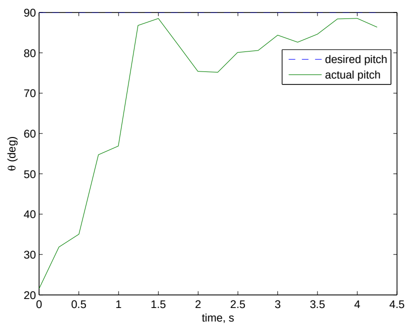
(a) $p$



(b) $h$



(c) Airspeed



(d) $\theta$

**Figure 4.12:** Simple controller level-to-hover transition flight data

### 4.6.2 Feedback Linearization Controller Flight Results

As discussed in the derivation of the feedback linearization controller in Section 4.3, values for the coefficients of lift and drag are required for generating desired inputs for trajectory tracking. In the implementation of this controller, values for $C_l$ and $C_d$ are obtained from Equations 2.7 and 2.8, both of which are functions of $\alpha$, the angle of attack. The method for measuring the current angle of attack of the tailsitter is

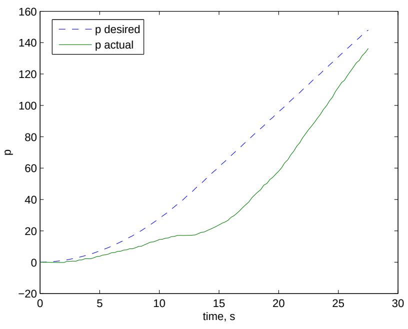
$$\alpha = \tan^{-1}(\dot{h}, \dot{p})$$

where $\dot{h}$ and $\dot{p}$, the velocities in the vertical and forward directions, are obtained from GPS sensors. As mentioned previously, the quality and frequency of GPS derived information are unfortunately not accurate or timely enough to obtain reliable measurements. Thus, the formulation for $\alpha$ and therefore $C_l$ and $C_d$ are adversely affected by the quality of GPS measurements. No satisfactory results were obtained for the feedback linearization controller using the $\alpha$ derived values of $C_l$ and $C_d$ in the algorithm. Instead, the values for $C_l$ and $C_d$ used by the controller were fixed with average values of each parameter seen while performing the transitions in simulation. Making this modification allowed satisfactory trajectory tracking results to be obtained.

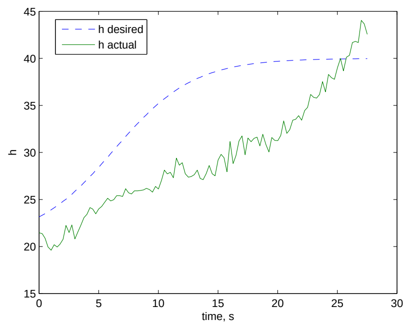
Flight results for the feedback linearization controller performing a hover-to-level transition are shown in Figure 4.13. There is significant deviation in trajectory tracking of up to 20 meters in the $p$ direction and 10 meters in the $h$ direction. However, it is possible to see the controller working to maintain tracking through the transition by observing the desired thrust and desired pitch angle plots. If the thrust matching controller and attitude controller were better able to match these desired values, better results for trajectory tracking would be evident.

Flight results for the feedback linearization controller performing a level-to-hover transition are shown in Figure 4.14. Tracking in the $p$ direction has about 15 meters of error at the greatest point. The $h$ trajectory tracking is off by 5 to 7 meters at its greatest point. It is interesting to note the drop in pitch angle at about $t = 3$.

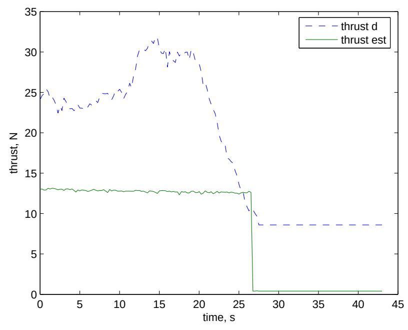
At this point, the tailsitter is deviating from the desired $p$ trajectory. To compensate for this, the tailsitter pitches down to move in the forward $p$ direction and try to get back on track.



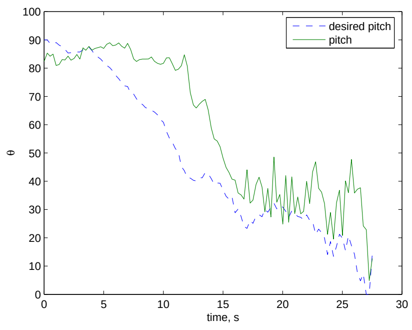
(a) $p$



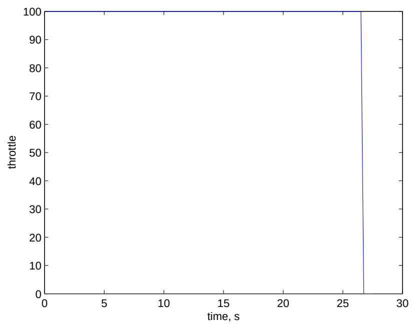
(b) $h$



(c) desired thrust

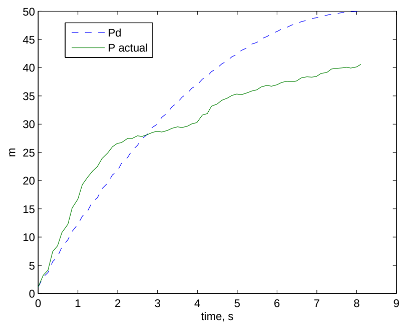


(d) $\theta$

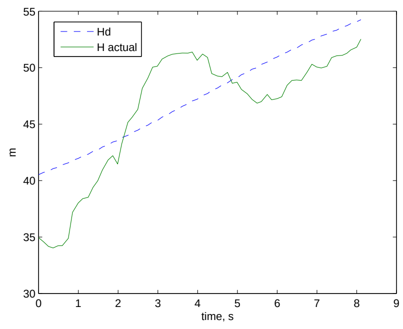


(e) throttle

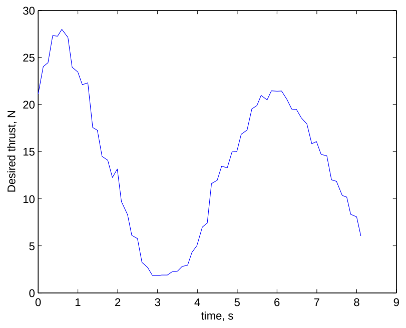
**Figure 4.13:** Feedback linearization controller hover-to-level trajectory tracking flight data



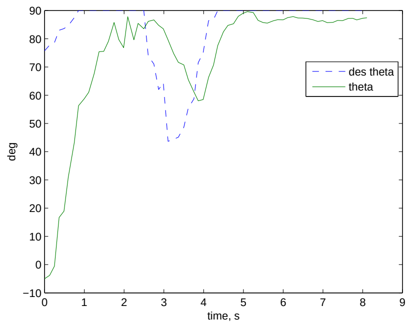
(a) $p$



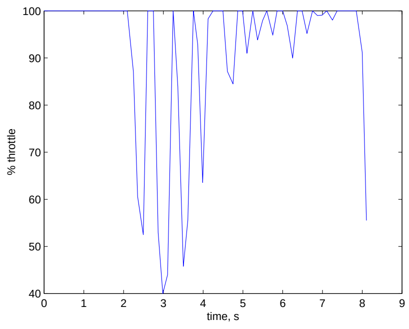
(b) $h$



(c) desired thrust



(d) $\theta$



(e) throttle

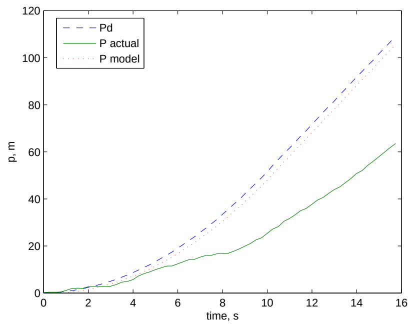
**Figure 4.14:** Feedback linearization controller level-to-hover trajectory tracking flight data

### 4.6.3 Adaptive Controller Flight Results

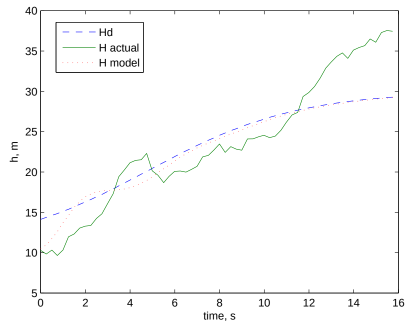
Flight results of a hover to level transition using the adaptive controller are shown in Figure 4.15. Tracking in the $p$ direction is good at the start of the maneuver but grows to nearly 40 meters at the end. Altitude tracking is much better, with deviations of less than 3 meters until the end of the transition, when error reaches about 10 meters. These errors are likely due to limitations in the attitude and thrust controllers. Also, the adaptive parameters $\hat{A}$ and $\hat{B}$ also change rapidly at the end of the maneuver. The adaptive controller will have a difficult time tracking a trajectory while the estimated parameters are still being adjusted by the algorithm. This problem might be lessened by increasing the $\gamma$ gains somewhat.

The thrust plot for the hover-to-level transition shows interesting behavior. The desired thrust is plotted alongside the estimated thrust, which is measured by accelerometers. Besides seeing the error in the thrust controller, the mechanism for performing the transition is evident. Thrust drops rapidly in the beginning of the maneuver, initiating a stall-tumble event causing the tailsitter's pitch angle to drop out of the hover position. Then thrust increases to steady the tailsitter and minimize altitude tracking error. Finally thrust decrease once again as steady level flight is assumed.

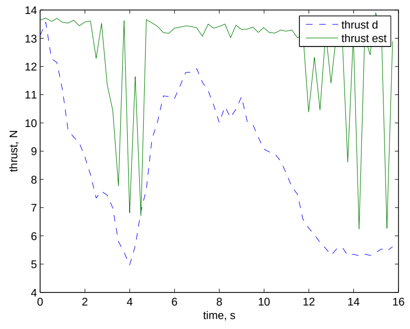
The results of a level-to-hover transition using the adaptive controller are shown in Figure 4.16. Tracking error in $p$ is up to 15 meters and is about the same amount in $h$. However, some desirable characteristics are seen in these plots. Observing the data in the $p$ plot, the tailsitter makes the transition too early and is hovering several meters off of the desired forward trajectory. To rectify this situation, there is a large momentary drop in pitch angle, which gives the tailsitter some forward airspeed and allows it to better approach the desired $p$ trajectory. Plots for $\hat{A}$ and $\hat{B}$ show that the adaptive parameters reach steady state values by the time the maneuver is finished.



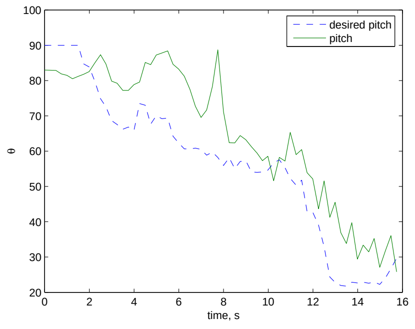
(a) $p$



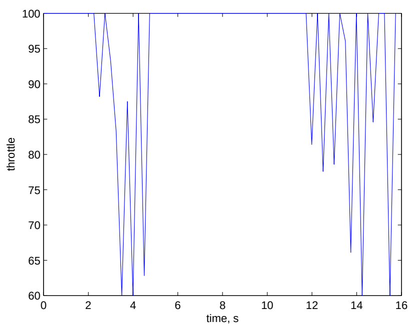
(b) $h$



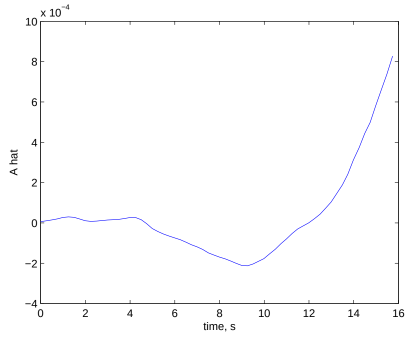
(c) thrust



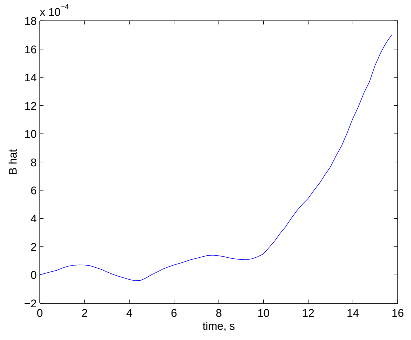
(d) $\theta$



(e) throttle

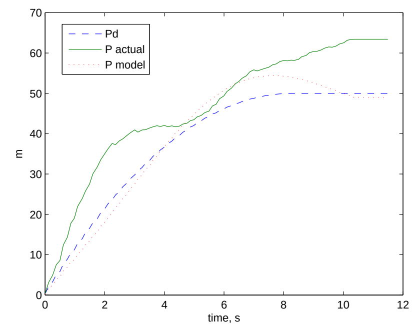


(f) $\hat{A}$

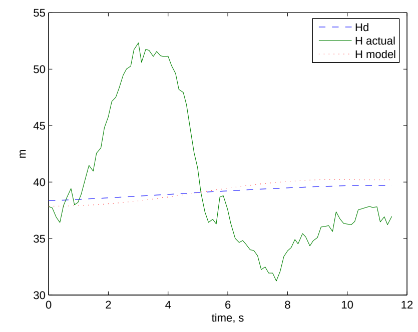


(g) $\hat{B}$

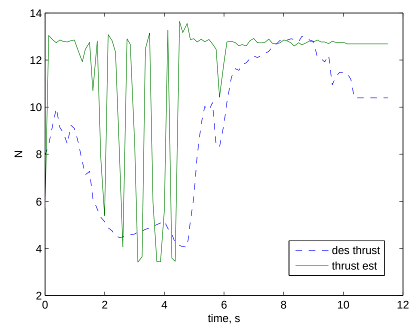
**Figure 4.15:** Adaptive controller hover-to-level trajectory tracking flight data



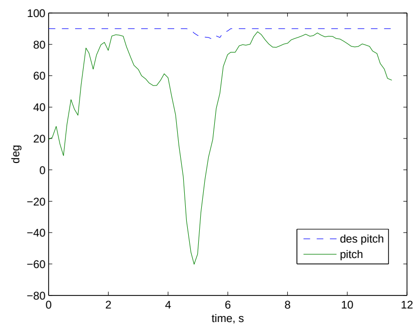
(a) $p$



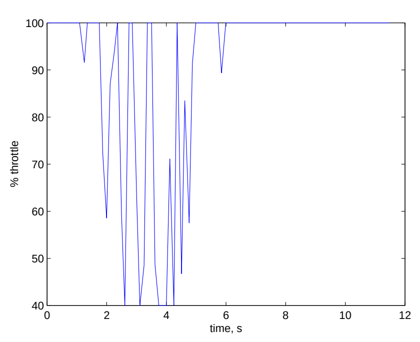
(b) $h$



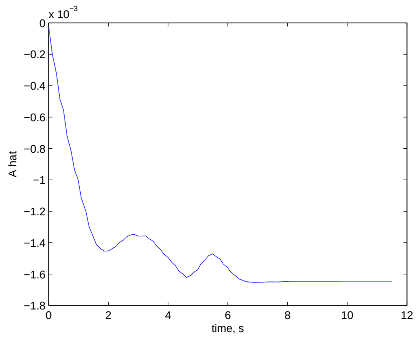
(c) thrust



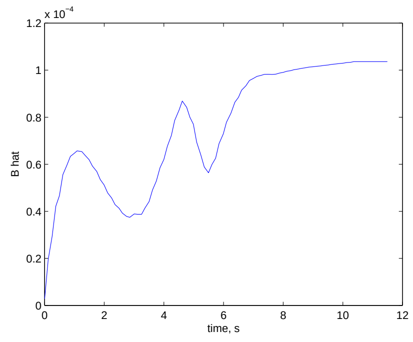
(d) $\theta$



(e) throttle



(f) $\hat{A}$



(g) $\hat{B}$

**Figure 4.16:** Adaptive controller level-to-hover trajectory tracking flight data

## 4.7 Chapter Summary

After discussing the creation of the desired trajectory generation goals and process, three algorithms for tracking those trajectories through both types of tail-sitter transitions were presented in this chapter. First, the simple controller was discussed, followed by the feedback linearization controller and the adaptive controller. Simulation results and flight test results were presented. In the next chapter, the autopilot state machine will be discussed. The autopilot state machine will use the hover position controller and level waypoint controller described in Chapter 3 along with the transition controllers presented in this chapter to fly a waypoint path consisting of a combination of level and hover waypoints.

# Chapter 5

# Autopilot State Machine

The highest level of autopilot control code is the state machine. It contains logic necessary to navigate a waypoint path consisting of both hover and level waypoints. The structure of the autopilot state machine is described in this chapter. Flight results for a sample waypoint path will also be shown.

## 5.1 Autopilot Structure

The autopilot state machine shown in Figure 5.1 retrieves waypoints uploaded by the Virtual Cockpit ground station and sets appropriate navigational control, throttle, and attitude control modes to achieve desired functionality. Possible flight commands in this autopilot are hover waypoints and level waypoints. Various flight modes include hover flight mode, level flight mode, and modes for transitioning between these modes. The autopilot state machine code is run in an autopilot function running at about 5 $Hz$. For each execution of the loop, UAV states and timers are checked to see if a move to another state is in order. A description of each autopilot state will be given.

It should be noted that in the autopilot state machine, hover-to-level transitions are performed immediately but level-to-hover transitions are performed only when close enough to the desired hover waypoint. This is because of the desire to stay in the more energy efficient level flight mode for as long as possible.

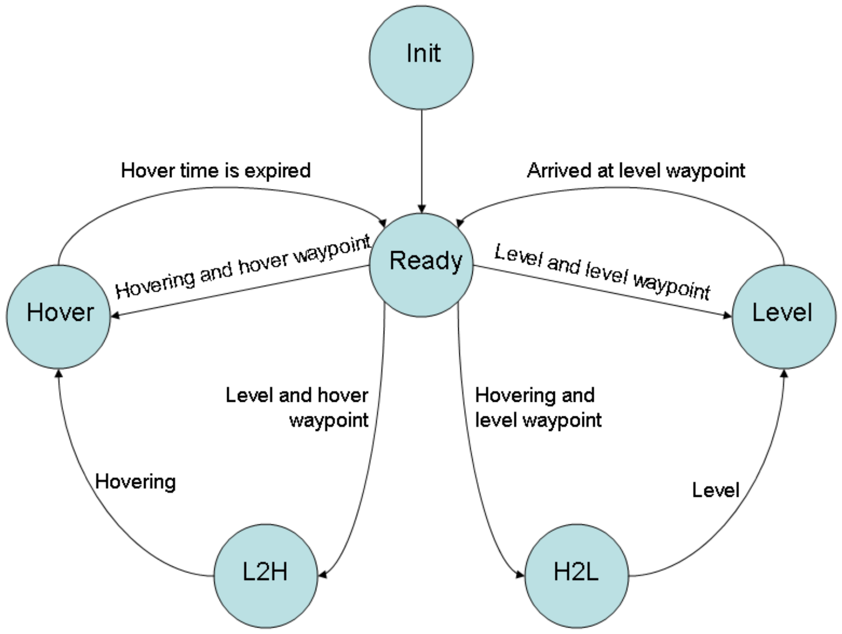


**Figure 5.1:** Autopilot state machine

### Initialization State

On start-up, the autopilot is in the Initialization state. In this state, any necessary initialization is performed. The autopilot remains in the Initialization state for just one cycle and then moves to the Ready state.

### Ready State

The Ready state retrieves the next waypoint command from the waypoint stack uploaded from Virtual Cockpit, described in Section 3.2. Depending on the current orientation of the tailsitter and the type of waypoint received, it then moves to the next appropriate state. If the tailsitter is hovering and the waypoint is a hover waypoint, the next state is the Hover state. If hovering but the next waypoint is a level waypoint, then the state machine moves to the Hover-to-Level state. If the tailsitter is level and a level waypoint is received, the state machine enters the Level state. Otherwise, if level and the next waypoint is a hover waypoint, the Level-Level-to-Hover state is entered.

### Hover State

In this state, the hover position controller described in Section 3.4 is turned on. Also, the throttle from altitude loop is enabled, which uses the throttle to control the tailsitter's altitude. When the tailsitter has arrived at the desired hover waypoint, the state machine moves to the Maintain Hover state.

### Maintain Hover State

Upon entering the Maintain Hover state a timer is initialized. Each hover waypoint has a hover time associated with it. When this time has expired, the state machine will move to the Ready state.

### Level State

When flying between level waypoints the level flight controller described in Section 3.5 is used. Throttle is controlled along with pitch angle to maintain a desired altitude. To control tailsitter attitude, the quaternion attitude controller is used as in the hover flight mode, although different gains are loaded.

### Rotate-to-Heading State

The Rotate to Heading state is entered before a hover to level transition is performed. Since the transition is to be performed along a straight line, it is necessary to first rotate the tailsitter in the direction of the new level waypoint. When the tailsitter is within a certain angle threshold of the desired heading, the Hover to Level state is entered.

**Hover-to-Level State**

While in the Hover-to-Level state, the desired trajectory is created and one of the transition path following algorithms described in Chapter 4 is executed. Which algorithm to perform is selectable by the user. Once the tailsitter achieves a level flight attitude, the Level state is entered, which will take the tailsitter to its waypoint.

**Level-Level-to-Hover State**

This state functions just like the Level state, but will switch to the Level to Hover state when the tailsitter arrives within a certain distance threshold of the hover waypoint. This is to allow the tailsitter to fly in level flight mode for as long as possible to conserve battery life.

**Level-to-Hover State**

The desired trajectory is created and one of the transition path following algorithms is executed. Once the tailsitter reaches a near-hover position, the Hover state is entered.

## 5.2 Autopilot Flight Results with Simple Controller

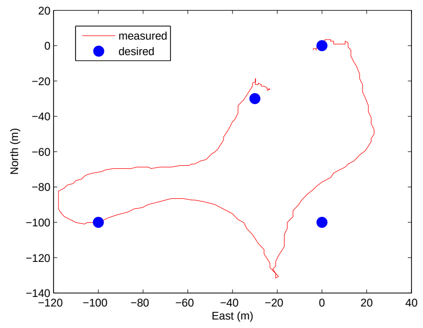
Flight results are shown in Figure 5.2 for a flight path composed of two level waypoints and two hover waypoints. The transitions in and out of the hover waypoint are done with the simple controller described in Section 4.2. In this experiment, the waypoints at (0,0) and (-30,-30), the first and last points of the path, are both hover waypoints while the remaining two points are level waypoints. The autopilot state is numbered, with the corresponding state names listed in Table 5.1. It can be seen that the state machine successfully navigates the tailsitter in level and hover flight modes as well as through transitions. It should be noted that the desired airspeed has a positive value for the hover waypoints which should be corrected to read zero.

An examination of the GPS waypoint path reveals poor tracking on first inspection, but much of this is due to the confined flight area in which flight testing was done. The tailsitter takes very wide turns in these results. If a larger flight test area were available and longer flight path legs were flown, then the turns would not be as noticeable and better waypoint tracking would be seen. Waypoint tracking for the tailsitter will be much better given a long, straight level flight path rather than a flight path with almost constant turns as is shown in the results.

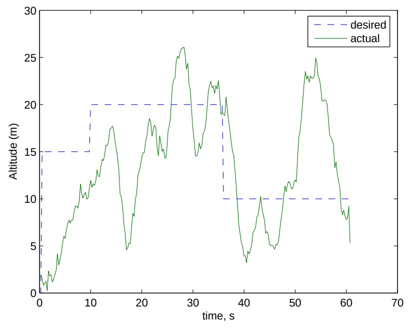
The tailsitter hovers on takeoff up to its first waypoint. The tailsitter comes within a threshold value of desired altitude at about $t = 12$ seconds. At that time, the next waypoint is interpreted. Since it is a level waypoint, an immediate hover-to-level transition is commanded, as can be seen in the plot of pitch angle. The tailsitter switches to the level flight controller and flies to the next waypoint. During the time the tailsitter is in level flight mode, pitch and throttle are adjusted to try to achieve desired altitude and airspeed. At $t = 50$ seconds, the tailsitter is close enough to the final hover waypoint to perform a level-to-hover transition. The tailsitter then descends to the ground.

**Table 5.1:** Autopilot state machine state numbers

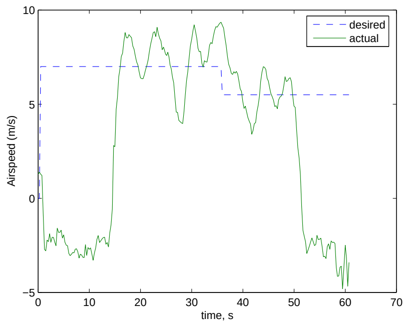
| Number | State |
|---|---|
| 0 | Initialization |
| 1 | Ready |
| 2 | Hover |
| 3 | Maintain Hover |
| 4 | Init Maintain Hover |
| 5 | Level |
| 6 | Rotate to Heading |
| 7 | Hover to Level |
| 8 | Init Hover to Level |
| 9 | Level to Hover |
| 10 | Init level to Hover |
| 11 | Level-Level to Hover |



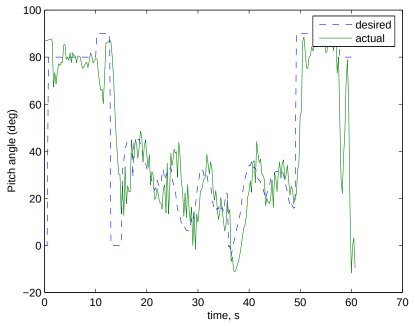
(a) N-E GPS path



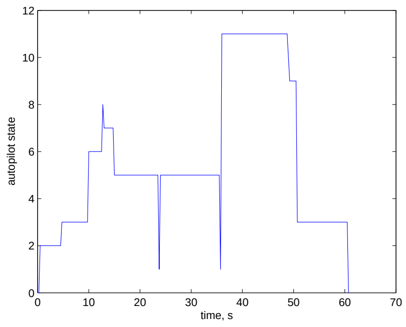
(b) Altitude



(c) Airspeed



(d) Pitch angle



(e) Autopilot state

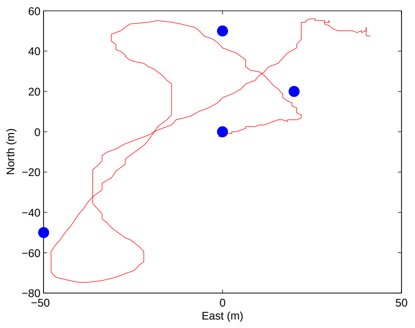
**Figure 5.2:** State machine flight results with simple controller

## 5.3 Autopilot Flight Results with Feedback Linearization Controller

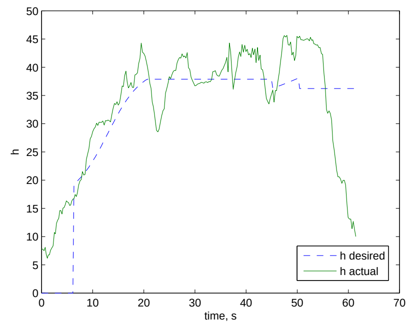
Flight results of the tailsitter flying a waypoint path composed of level and hover waypoints with the transitions handled by the feedback linearization controller is shown in Figure 5.3. The path begins with a hover waypoint at (0,0). Then the tailsitter transitions to level flight and flies through two level waypoints. Finally the tailsitter hovers at (20,20). For the transitions, the $p$ and $h$ trajectories are plotted. It should be noted, however, that the desired values for $p$ and $h$ do not change once the transition is complete. Only the portions of the desired trajectories that occur during each transition state should be considered.

For this flight, the GPS track plot shows large turns after reaching the desired level waypoints. If larger flight path legs could be flown, these large deviations would not be as noticeable. Once the tailsitter completes the turn and is heading for the next waypoint, it is able to fly in a straight line.

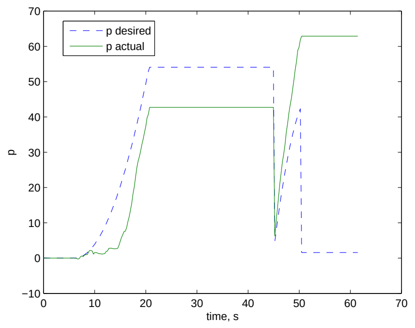
As seen in the altitude and pitch plots, the tailsitter takes off from ground level in a hover position. At just before $t = 10$ seconds, the tailsitter begins a hover-to-level transition. This maneuver can be seen in the $p$ and $h$ plots from $t = 10$ until $t = 20$. Both of these plots show good tracking during the maneuver. Once the tailsitter is level, the level flight controller is engaged and level waypoints are flown, with altitude and airspeed being controlled by pitch angle and throttle. At $t = 45$ seconds the tailsitter is approaching its final waypoint, which is designated as a hover waypoint. The transition is performed and the tailsitter descends to the ground.



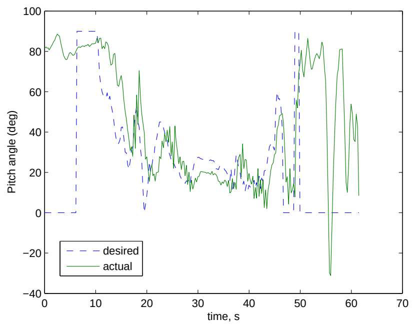
(a) N-E GPS path



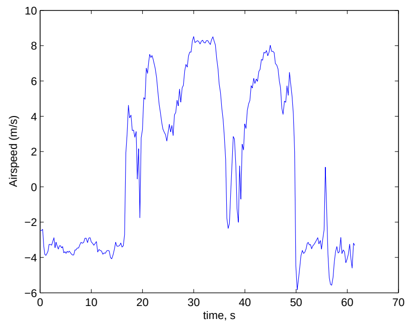
(b) Altitude



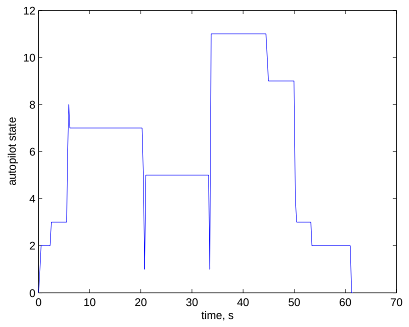
(c) $p$ trajectory



(d) Pitch angle



(e) Airspeed



(f) Autopilot state

**Figure 5.3:** State machine flight results with feedback linearization controller

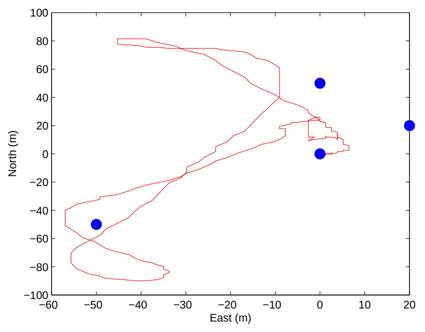
## 5.4 Autopilot Flight Results with Adaptive Controller

The adaptive controller is used to negotiate transitions for the flight path plotted in Figure 5.4. The desired flight path is the same as that used in Section 5.3, with an initial hover waypoint followed by two level waypoints and a final hover waypoint. Like the desired trajectories for $p$ and $h$, the adaptive parameters $\hat{A}$ and $\hat{B}$ only change during the transition state.

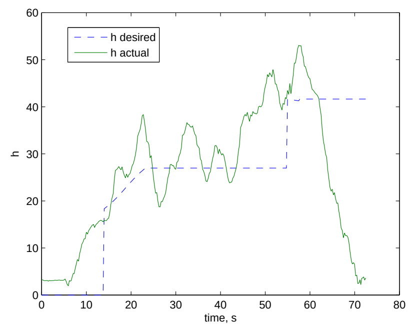
The GPS tracking plot looks considerably worse than the plots shown for the simple and feedback linearization controllers. This is due to the same effects from having short flight path legs as well as from an increase in wind during this flight. In any case, however, the autopilot state machine as well as the adaptive controller-managed transitions can be seen to work properly despite the poor performance of the level and hover controllers themselves.

The tailsitter begins on the ground and hovers up to a desired altitude, whereupon a hover-to-level transition is commanded. The pitch angle plot shows that the transition is completed successfully, but the $p$ plot shows that tracking in the forward direction exhibited error of about 20 $m$. Tracking is off by about 10 $m$ in the $h$ direction during the transition, which lasts from about $t = 15$ seconds to $t = 25$ seconds. Once in a level orientation, the level flight controller is engaged and level waypoints are flown. At about $t = 55$ seconds a level-to-hover transitions is performed. Tracking during this transition is much better in $p$ but about the same in $h$.

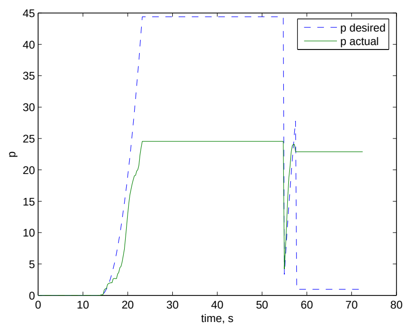
The plots for $\hat{A}$ and $\hat{B}$ show the adaptive parameters changing during each transition type. The values of these parameters are not changed during times when the tailsitter is not in a transition state. These plots should only be considered for the times when a transition is in progress. It can be seen that both parameters approach much different values for the two different transitions, which is indicative of the changing flight conditions when each is encountered.



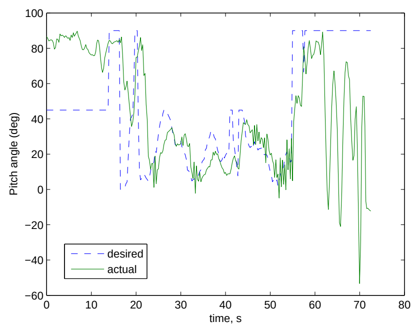
(a) N-E GPS path



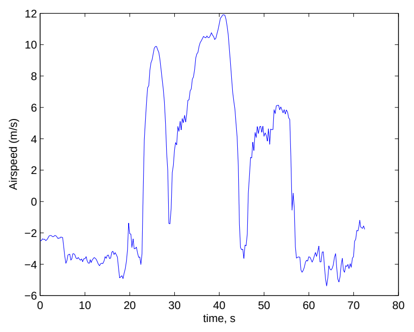
(b) Altitude



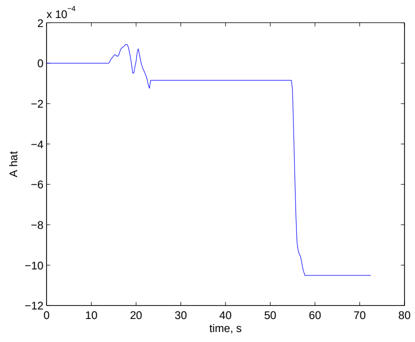
(c) $p$ trajectory



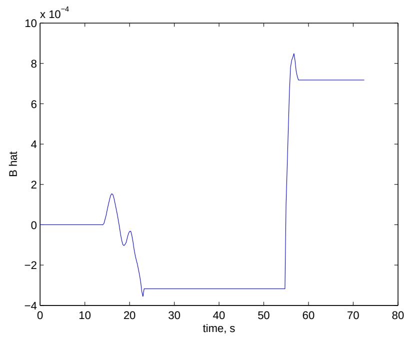
(d) Pitch angle



(e) Airspeed



(f) $\hat{A}$



(g) $\hat{B}$

**Figure 5.4:** State machine flight results with adaptive controller

## 5.5 Chapter Summary

The autopilot state machine receives waypoint commands and engages appropriate flight modes to navigate the desired path. The state machine receives level or hover waypoints and appropriately commands either the level flight controller, hover position controller, or one of the transition controllers, depending on what is currently required. Flight results were given for flying a waypoint path with each of the transition controllers described in Chapter 4.

# Chapter 6

# Conclusion

## 6.1 Summary of Results

Flight test results show the three transition trajectory tracking algorithms working satisfactorily. The simple controller can be used in cases where a specific trajectory does not require tracking and only a flight mode transition is desired. When further control over the tailsitter's path is desired, the feedback linearization controller or adaptive controller can be used. These two controllers exhibit similar performance, but the adaptive controller requires knowledge of fewer parameters and therefore is the more useful of the two.

There is much room for improvement in the actual flight results. The difficulties in tracking trajectories well are attributable to a number of factors. Tailsitter position and velocity measurements are required for both feedback linearization and adaptive controllers. Both of these measurements are obtained by GPS readings, but these readings are subject to inaccuracies and time delays. A faster, more precise measure of position and velocity would be needed to track the transition trajectories well. Along with imperfect sensors, limitations in the attitude and thrust controllers also lead to tracking error in the transition trajectory tracking algorithms. Finally, wind of almost any strength at all is enough to disturb the tailsitter away from its desired flight trajectory. If these limitations were addressed, the algorithms presented in this thesis should perform much better.

## 6.2 Future Work

In this thesis the trajectories to track were limited to transition maneuvers performed with the tailsitter travelling straight, with the nose/belly always facing the direction of travel. However, the tailsitter is a very agile and flexible aircraft. Tailsitter maneuverability is much greater than a fixed wing aircraft due to its ability to hover. With a quaternion attitude representation and controller, it is theoretically possible to command the tailsitter to any conceivable attitude. Of course, some attitudes will be very hard to maintain for an extended period of time if they are outside of a near hover position or in the stall region. However, future research could focus on creating and tracking trajectories for any number of flight paths, such as spirals, curved paths, or even aerobatic flight maneuvers.

Furthermore, each type of maneuver, like the transitions described in this thesis or other possible maneuvers that could be developed for the tailsitter, has many possibilities for execution. For example, a hover-to-hover maneuver with a climb in altitude could be performed by climbing in hover mode to the correct altitude and pitching slightly in hover mode to transition to the desired position. Or, the same maneuver could be accomplished by transitioning to level flight, spiraling up to the appropriate altitude, getting close to the final desired hover waypoint, and transitioning back to hover mode. It would be a beneficial study to see what maneuver types are more energy efficient or otherwise beneficial in different situations.

It has been suggested that the tailsitter could benefit greatly by the addition of a downward pointing optic flow or other vision sensor. With such a sensor, hover stabilization techniques would be possible by adjusting the control surfaces to cancel out optic flow movements in all directions caused by wind or other factors. For example, the tailsitter could use vision processing algorithms to identify a landing platform and land in the proper orientation to recharge batteries. With the ability to take-off from the ground, the tailsitter equipped with autonomous landing technology could fly several consecutive missions without a need for human interaction.

# Bibliography

[1] N. B. Knoebel, S. R. Osborne, J. S. Matthews, A. M. Eldredge, and R. W. Beard, "Computationally simple model reference adaptive control for miniature air vehicles," *Proceedings of the 2006 American Control Conference*, pp. 5978–5983, 2006. 3

[2] N. B. Knoebel, S. R. Osborne, D. O. Snyder, T. W. McLain, R. W. Beard, and A. M. Eldredge, "Preliminary modeling, control, and trajectory design for miniature autonomous tailsitters," *AIAA Conference on Guidance, Navigation, and Control*, 2006. 3

[3] H. Stone and K. C. Wong, "Preliminary design of a tandem-wing tail-sitter UAV using multi-disciplinary design optimisation," *International Aerospace Congress*, pp. 707–720, 2004. 3

[4] H. Stone and G. Clarke, "The t-wing: A VTOL UAV for defense and civilian applications," *UAV Australia Conference*, 2001. 4

[5] R. H. Stone, "Control architecture for a tail-sitter unmanned air vehicle," *Proceedings of the 5th Asian Control Conference*, pp. 736–744, 2004. 4

[6] H. Stone and G. Clarke, "Optimization of transition maneuvers for a tail-sitter unmanned air vehicle (UAV)," *Australian International Aerospace Congress*, 2001. 4

[7] http://www.aerovironment.com/. 4

[8] T. J. Cord and S. Newbern, "Unmanned air vehicles: New challenges in design," *Proceedings of the 2001 IEEE Aerospace Conference*, vol. 6, pp. 2699–2704, 2001. 4

[9] W. E. Green and P. Y. Oh, "A MAV that flies like an airplane and hovers like a helicopter," *Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 693–698, 2005. 4

[10] ——, "A fixed-wing aircraft for hovering in caves, tunnels, and buildings," *Proceedings of the 2006 American Control Conference*, pp. 1092–1097, 2006. 4

[11] ——, "Autonomous hovering of a fixed-wing micro air vehicle," *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pp. 2164–2169, 2006. 4

[12] ——, “Optic flow based collision avoidance on a hybrid MAV,” *IEEE Robotics and Automation Magazine*, in press. 4

[13] ——, “Flying insect inspired vision for autonomous aerial robot maneuvers in near-earth environments,” *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pp. 2347–2352, 2004. 4

[14] P. Y. Oh and W. E. Green, “CQAR: Closed quarter aerial robot design for reconnaissance, surveillance and target acquisition tasks in urban areas,” *International Journal of Computational Intelligence*, vol. 1, no. 4, pp. 353–360, 2004. 4

[15] A. Ailon, “Control of a VTOL aircraft: Motion planning and trajectory tracking,” *Proceedings of the 13th Mediterranean Conference on Control and Automation*, pp. 1493–1498, 2005. 4

[16] P. Martin, S. Devasia, and B. Paden, “A different look at output tracking: control of a VTOL aircraft,” *Proceedings of the 33rd Conference on Decision and Control*, pp. 2376–2381, 1994. 4

[17] P. Setlur, D. Dawson, Y. Fang, and B. Costic, “Nonlinear tracking control of the VTOL aircraft,” *Proceedings of the 40th IEEE Conference on Decision and Control*, pp. 4592–4597, 2001. 4

[18] B. T. Costic, D. M. Dawson, M. S. de Queiroz, and V. Kapila, “Quaternion-based adaptive attitude tracking controller without velocity measurements,” *AIAA Journal of Guidance, Control, and Dynamics*, vol. 24, no. 6, pp. 1214–1222, 2001. 4

[19] J. Hauser and R. Hindman, “Aggressive flight maneuvers,” *Proceedings of the 36th Conference on Decision and Control*, pp. 4186–4191, 1997. 4

[20] M. D. Shuster, “A survey of attitude representations,” *The Journal of the Astronautical Sciences*, vol. 41, no. 4, pp. 439–517, 1993. 4, 22

[21] N. B. Knoebel, *Master’s Thesis*. Brigham Young University, 2007. 19, 37

[22] http://www.procerusuav.com/. 20

[23] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, “Vector field path following for small unmanned air vehicles,” *Proceedings of the 2006 American Control Conference*, pp. 5788–5794, 2006. 27

[24] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, New Jersey: Prentice Hall, 2002. 37, 40, 44