Modeling and Control of a Tailsitter with a Ducted Fan

Matthew Elliott Argyle
Brigham Young University - Provo, matt.argyle@gmail.com

Follow this and additional works at: https://scholarsarchive.byu.edu/studentpub
Part of the Electrical and Computer Engineering Commons

BYU ScholarsArchive Citation
https://scholarsarchive.byu.edu/studentpub/180

This Peer-Reviewed Article is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Student Publications by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Modeling and Control of a Tailsitter with a Ducted Fan

Matthew Elliott Argyle

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Randal W. Beard, Chair
Timothy W. McLain
Brian D. Jeffs
Mark B. Colton
James K. Archibald

Department of Electrical and Computer Engineering
Brigham Young University
June 2016

Copyright © 2016 Matthew Elliott Argyle
All Rights Reserved
ABSTRACT

Modeling and Control of a Tailsitter with a Ducted Fan

Matthew Elliott Argyle
Department of Electrical and Computer Engineering, BYU
Doctor of Philosophy

There are two traditional aircraft categories: fixed-wing which have a long endurance and a high cruise airspeed and rotorcraft which can take-off and land vertically. The tailsitter is a type of aircraft that has the strengths of both platforms, with no additional mechanical complexity, because it takes off and lands vertically on its tail and can transition the entire aircraft horizontally into high-speed flight. In this dissertation, we develop the entire control system for a tailsitter with a ducted fan.

The standard method to compute the quaternion-based attitude error does not generate ideal trajectories for a hovering tailsitter for some situations. In addition, the only approach in the literature to mitigate this breaks down for large attitude errors. We develop an alternative quaternion-based error method which generates better trajectories than the standard approach and can handle large errors. We also derive a hybrid backstepping controller with almost global asymptotic stability based on this error method.

Many common altitude and airspeed control schemes for a fixed-wing airplane assume that the altitude and airspeed dynamics are decoupled which leads to errors. The Total Energy Control System (TECS) is an approach that controls the altitude and airspeed by manipulating the total energy rate and energy distribution rate, of the aircraft, in a manner which accounts for the dynamic coupling. In this dissertation, a nonlinear controller, which can handle inaccurate thrust and drag models, based on the TECS principles is derived. Simulation results show that the nonlinear controller has better performance than the standard PI TECS control schemes.

Most constant altitude transitions are accomplished by generating an optimal trajectory, and potentially actuator inputs, based on a high fidelity model of the aircraft. While there are several approaches to mitigate the effects of modeling errors, these do not fully remove the accurate model requirement. In this dissertation, we develop two different approaches that can achieve near constant altitude transitions for some types of aircraft. The first method, based on multiple LQR controllers, requires a high fidelity model of the aircraft. However, the second method, based on the energy along the body axes, requires almost no aerodynamic information.

Keywords: Attitude Control, Attitude Error, Guidance and Control, Tailsitter, TECS, VTOL
# Table of Contents

List of Tables ........................................................................................................ vi

List of Figures .......................................................................................................... vii

Nomenclature ........................................................................................................... ix

Abbreviations ........................................................................................................... xi

Chapter 1 Introduction ............................................................................................. 1
  1.1 Control Architecture ....................................................................................... 5
    1.1.1 Vertical Bat Control Architecture ......................................................... 7
  1.2 Contributions .................................................................................................... 8
  1.3 Organization ..................................................................................................... 9

Chapter 2 Coordinate Frames and Attitude Representations .................................... 10
  2.1 Coordinate Frames .......................................................................................... 10
  2.2 Attitude Representations ................................................................................ 10
    2.2.1 Euler Angles .......................................................................................... 11
    2.2.2 Quaternions ........................................................................................... 12

Chapter 3 Vertical Bat Kinematics and Dynamics ..................................................... 16
  3.1 Kinematics ...................................................................................................... 18
  3.2 Dynamics ........................................................................................................ 19
    3.2.1 Gravity .................................................................................................. 19
    3.2.2 Propeller ................................................................................................. 20
    3.2.3 Duct ....................................................................................................... 25
    3.2.4 Control Vanes ......................................................................................... 28
    3.2.5 Airframe ................................................................................................. 30
  3.3 Conclusions .................................................................................................... 37

Chapter 4 Attitude Control ....................................................................................... 39
  4.1 Attitude Error .................................................................................................. 41
    4.1.1 Quaternion Attitude Error ....................................................................... 41
    4.1.2 Resolved Tilt Twist ............................................................................... 43
    4.1.3 Resolved Euler Angle ............................................................................. 49
    4.1.4 Attitude Error Comparison ................................................................... 61
  4.2 Attitude Control ............................................................................................. 62
    4.2.1 PID ......................................................................................................... 63
    4.2.2 Backstepping ......................................................................................... 64
    4.2.3 REA-QA Hybrid .................................................................................... 71
    4.2.4 Adaptive Moment Generation ............................................................... 76
  4.3 Results ............................................................................................................. 77
4.4 Conclusions ................................................. 85

Chapter 5 Hover Control ......................................... 87
  5.1 Reference Models ......................................... 90
    5.1.1 Position Reference Model ................................. 90
    5.1.2 Altitude Reference Model ............................... 93
  5.2 Control Scheme .......................................... 96
    5.2.1 Position Controller .................................... 96
    5.2.2 Altitude Controller .................................. 98
  5.3 PID Modifications ........................................ 99
    5.3.1 Approaches ........................................... 100
    5.3.2 Comparison .......................................... 102
  5.4 Analysis .................................................. 104
    5.4.1 Position Control ...................................... 105
    5.4.2 Altitude Control ..................................... 107
  5.5 Flight Tests .............................................. 109
    5.5.1 Stationary Initialization ............................... 110
    5.5.2 Moving Initialization ................................ 111
  5.6 Conclusions .............................................. 115

Chapter 6 Level Flight Control ................................. 116
  6.1 Energy Definitions ....................................... 118
  6.2 Original TECS ............................................ 119
  6.3 Nonlinear TECS .......................................... 121
  6.4 Pitch and Thrust Control ................................. 128
  6.5 Other Longitudinal Controllers ............................ 131
  6.6 Lateral Control ......................................... 132
  6.7 Dynamics ................................................ 133
  6.8 Parameter Estimation ..................................... 134
  6.9 Simulation Results ....................................... 139
  6.10 Conclusions ............................................. 142

Chapter 7 Transition Control ................................... 145
  7.1 Introduction ............................................ 145
    7.1.1 Trajectory Design .................................... 146
    7.1.2 Control Design ....................................... 149
    7.1.3 Discussion ........................................... 152
  7.2 Dynamics ................................................ 153
  7.3 Trajectory Design ....................................... 153
  7.4 Control Algorithms ....................................... 157
    7.4.1 LQR ................................................ 157
    7.4.2 Energy Transition .................................... 158
  7.5 Simulation Results ....................................... 161
    7.5.1 LQR ................................................ 161
    7.5.2 Energy ............................................... 167
List of Tables

4.1 Attitude error runtime comparison ........................................ 62

5.1 Initial conditions for the hover controller initialization tests .......... 105

6.1 Airframe parameters [16] .......................................................... 134

6.2 Mean, standard deviation, and maximum of the norm of the parameter estimation error. A 30 second data averaging window and all nine airspeeds were used. ......................................................... 138

6.3 Mean, standard deviation, and maximum of the norm of the parameter estimation error. A 30 second data averaging window and the full constrained optimization were used. ......................................................... 138

6.4 Mean, standard deviation, and maximum of the norm of the parameter estimation error. All nine airspeeds and the full constrained optimization were used. ......................................................... 138

7.1 Linear model operating velocities .............................................. 162

A.1 Vertical Bat airframe parameters .............................................. 189
List of Figures

1.1 Fixed-wing VTOL aircraft designs ........................................ 2
1.2 Early manned tailsitter prototypes ........................................ 3
1.3 Unmanned tailsitter prototypes ........................................... 4
1.4 3/4 scale model V-Bat prototype .......................................... 5
1.5 V-Bat's control architecture .............................................. 7

3.1 Motor RPM model .............................................................. 23
3.2 Optimized propeller model without duct ................................... 24
3.3 Optimized propeller model with duct ....................................... 25
3.4 Simplified duct airflow ...................................................... 26
3.5 Control vanes ................................................................. 28
3.6 Control vane model ........................................................... 31
3.7 Level aerodynamic coefficients ............................................ 33
3.8 Hover heading, airflow, and forces definitions .......................... 35
3.9 Hover aerodynamic coefficients ........................................... 36

4.1 QA and MQA error comparison ............................................. 42
4.2 Ideal and QA rotation ....................................................... 43
4.3 QA error ....................................................................... 44
4.4 RTT tilt error ................................................................. 45
4.5 RTT twist error ............................................................... 47
4.6 RTT error rotation ........................................................... 48
4.7 RTT error ..................................................................... 49
4.8 Examples where the RTT error fails ....................................... 50
4.9 Additional examples where the RTT error fails ......................... 51
4.10 REA error ................................................................. 53
4.11 REA error rotation ........................................................ 53
4.12 Examples where the REA error fails ..................................... 54
4.13 Right triangle used to simplify the selective saturation derivation .... 56
4.14 REA error with selective saturation .................................... 58
4.15 Comparison of the REA saturation methods ............................ 61
4.16 REA-QA hybrid control guard concept .................................. 76
4.17 Attitude error trajectories for case 1 ..................................... 79
4.18 Attitude error trajectories for case 2 ..................................... 80
4.19 Attitude error trajectories for case 3 ..................................... 81
4.20 Attitude error trajectories for case 4 ..................................... 82
4.21 Comparison of the aircraft's velocity that the different attitude error methods generate for several different attitude commands ...................... 83
4.22 Aircraft position for the QA and REA PID controllers during a simultaneous tilt and heading command .................................. 84
4.23 Aircraft position for the REA-QA hybrid controller during a simultaneous tilt and heading command .................................. 84
4.24 Comparison of the aircraft’s attitude during a normal flight .......... 85
4.25 Stabilized RLS vane parameter estimates

5.1 The V-Bat’s hover control architecture
5.2 Position reference model state machine
5.3 Altitude reference model state machine
5.4 The commanded tilt and tilt direction for the hover position controller
5.5 Simulation results showing the response of the unmodified PID control scheme when it is initialized while the vehicle is moving at 5 m/s
5.6 Simulation results comparing the response of the PID modifications
5.7 Percentage of the tests where max \( \theta_i(t) \leq \theta \)
5.8 Linearized position control system
5.9 Linearized position decelerate control system
5.10 Linearized altitude control system
5.11 Linearized altitude climbing control system
5.12 Linearized altitude descending control system
5.13 Stationary hover flight test
5.14 Constant altitude waypoint following flight test
5.15 Varying altitude waypoint following flight test
5.16 Moving initialization hover flight test
5.17 Climbing initialization hover flight test

6.1 Comparison of the desired value generation methods
6.2 Nonlinear non-adaptive TECS control architecture
6.3 Realistic nonlinear non-adaptive TECS control architecture
6.4 Ideal simulation results
6.5 Realistic simulation results for wings level flight
6.6 Realistic simulation results for a banked flight

7.1 Trim states between hover and level flight
7.2 Sample transition trajectories
7.3 Comparison of the LQR transition controllers
7.4 Comparison of the LQR transition with different acceleration commands
7.5 Comparison of the LQR transition with model errors
7.6 LQR transition with wind and without wind correction
7.7 LQR transition with wind and with wind correction
7.8 Energy transition with no wind
7.9 Energy transition with wind
7.10 Energy LTH transition when the initial pitch is low
7.11 Incremental testing of the energy transition controller

8.1 Vertical Bat flight

C.1 Kestrel 2.4 HIL simulator
C.2 MAGICC Interface structure
C.3 Kestrel 3 hardware in the loop simulator
**Nomenclature**

- Unit vectors along the $x$, $y$, and $z$ axes are denoted as $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$.
- In general a matrix is denoted by a bold uppercase letter e.g. $\mathbf{A}$.
- In general a vector is denoted by a bold lowercase letter e.g. $\mathbf{v}$.
- A tilde over a variable represents the error.
- A hat over a variable represents an estimate of the variable.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>Angle of attack</td>
</tr>
<tr>
<td>$\mathcal{B}$</td>
<td>Body coordinate frame</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Side slip angle</td>
</tr>
<tr>
<td>$\mathcal{D}$</td>
<td>Desired coordinate frame</td>
</tr>
<tr>
<td>$\delta_a$</td>
<td>Aileron command</td>
</tr>
<tr>
<td>$\delta_e$</td>
<td>Elevator command</td>
</tr>
<tr>
<td>$\delta_r$</td>
<td>Rudder command</td>
</tr>
<tr>
<td>$\delta_t$</td>
<td>Throttle command</td>
</tr>
<tr>
<td>$\delta_v$</td>
<td>Vane command</td>
</tr>
<tr>
<td>$\mathcal{E}$</td>
<td>Error coordinate frame</td>
</tr>
<tr>
<td>$\mathbf{e}$</td>
<td>Error vector</td>
</tr>
<tr>
<td>$E$</td>
<td>Energy</td>
</tr>
<tr>
<td>$\eta$</td>
<td>Quaternion</td>
</tr>
<tr>
<td>$\eta_0$</td>
<td>Real component of the quaternion</td>
</tr>
<tr>
<td>$\bar{\eta}$</td>
<td>Imaginary component of the quaternion</td>
</tr>
<tr>
<td>$\mathbf{f}$</td>
<td>Force</td>
</tr>
<tr>
<td>$g$</td>
<td>Gravity</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Angle of rotation</td>
</tr>
<tr>
<td>$h$</td>
<td>Altitude</td>
</tr>
<tr>
<td>$\mathcal{I}$</td>
<td>Inertial coordinate frame</td>
</tr>
<tr>
<td>$\mathbf{J}$</td>
<td>Moment of inertia</td>
</tr>
<tr>
<td>$k$</td>
<td>Gain or parameter</td>
</tr>
<tr>
<td>$m$</td>
<td>Mass</td>
</tr>
<tr>
<td>$\mathbf{m}$</td>
<td>Moment</td>
</tr>
<tr>
<td>$\nu$</td>
<td>Axis of rotation</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Angular velocity</td>
</tr>
<tr>
<td>$\omega_n$</td>
<td>Natural frequency</td>
</tr>
<tr>
<td>$\mathbf{p}$</td>
<td>Position</td>
</tr>
<tr>
<td>$\mathbf{R}_a^b$</td>
<td>Rotation matrix from frame $a$ frame to $b$</td>
</tr>
<tr>
<td>$T$</td>
<td>Thrust</td>
</tr>
<tr>
<td>$\theta$</td>
<td>Pitch</td>
</tr>
<tr>
<td>$\mathbf{v}$</td>
<td>Velocity</td>
</tr>
<tr>
<td>$V_a$</td>
<td>Airspeed</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>Damping ratio</td>
</tr>
</tbody>
</table>
Superscript
\[ b \quad \text{Body coordinate frame} \\
\quad c \quad \text{Command} \\
\quad d \quad \text{Desired} \\
\quad d \quad \text{Desired coordinate frame} \\
\quad r \quad \text{Reference} \\
\quad e \quad \text{Error coordinate frame} \\
\quad i \quad \text{Inertial coordinate frame} \\
\top \quad \text{Transpose} \\
t \quad \text{Trim} \\
\]

Subscript
\[ d \quad \text{Down} \\
\quad e \quad \text{East} \\
\quad n \quad \text{North} \\
\]

Functions
\[ |\cdot| \quad \text{Absolute value} \\
\|\cdot\| \quad \text{Norm} \\
c_\theta \quad \cos(\theta) \\
\text{E2Q}(\cdot) \quad \text{Function that converts an Euler angle sequence to a quaternion} \\
\text{Q2E}(\cdot) \quad \text{Function that converts a quaternion to an Euler angle sequence} \\
\text{Q2RM}(\cdot) \quad \text{Function that converts a quaternion to a rotation matrix} \\
s_\theta \quad \sin(\theta) \\
t_\theta \quad \tan(\theta) \]
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>Continuous Ascent</td>
</tr>
<tr>
<td>CFD</td>
<td>Computational Fluid Dynamics</td>
</tr>
<tr>
<td>FMC</td>
<td>Flight Mode Controller</td>
</tr>
<tr>
<td>HTL</td>
<td>Hover to Level</td>
</tr>
<tr>
<td>LTH</td>
<td>Level to Hover</td>
</tr>
<tr>
<td>MPC</td>
<td>Model Predictive Control</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional Integral Derivative</td>
</tr>
<tr>
<td>QA</td>
<td>Quaternion Attitude</td>
</tr>
<tr>
<td>REA</td>
<td>Resolved Euler Angle</td>
</tr>
<tr>
<td>RTT</td>
<td>Resolved Tilt Twist</td>
</tr>
<tr>
<td>SLC</td>
<td>Successive Loop Closure</td>
</tr>
<tr>
<td>TECS</td>
<td>Total Energy Control System</td>
</tr>
<tr>
<td>V-Bat</td>
<td>Vertical Bat</td>
</tr>
<tr>
<td>VC</td>
<td>Virtual Cockpit</td>
</tr>
<tr>
<td>VTOL</td>
<td>Vertical Take-off and Landing</td>
</tr>
<tr>
<td>WM</td>
<td>Waypoint Manager</td>
</tr>
</tbody>
</table>
Chapter 1. Introduction

There are two traditional categories of aircraft, fixed-wing airplanes and helicopters, each having unique strengths and weakness. Fixed-wing airplanes have a long flight endurance and a high cruise velocity, however, they have restrictive requirements for take-off and landing. For example, manned fixed-wing airplanes typically require long runways while unmanned airplanes can require runways, launchers, or nets. At the very least, an unmanned fixed-wing aircraft requires a large open field for take-off and landing. Helicopters, or more recently quadcopters or rotorcraft in general, do not generate lift from wings. Instead, they counteract gravity solely using thrust generated by their propellers. While this allows them to take-off and land vertically (VTOL) in constrained locations, they do not have the endurance or cruising velocity of a fixed-wing aircraft.

There have been many different attempts to develop an aircraft with the best of both platforms: an aircraft with the flight speed and endurance of a fixed-wing aircraft and the VTOL capabilities of a rotorcraft. These approaches include using tilt-rotors like the V-22 Osprey, shown in Figure 1.1a and 1.1b, or directed thrust like the AV-8B Harrier and the new F-35B Lightning II shown in Figure 1.1c. While these aircraft have been used successfully, the VTOL capabilities add extra mechanical complexity that can hurt reliability, due to more failure points, and increase the cost and build time of the aircraft.

There are other airframe concepts that combine the VTOL, speed, and endurance capabilities. For example, one of Amazon’s Prime Air prototypes merges a quadcopter with a fixed-wing aircraft and has nine propellers. Eight of the propellers provide lift and attitude control authority while the remaining propeller provides forward thrust. The aircraft takes off and lands while it is horizontal, like the F-35B, and is able to achieve a high cruise velocity with the primary rear propeller. While details have not been released, it appears that the airframe will generate some lift during high-speed flight but it is unclear if the limited
surface area will generate enough lift to counteract gravity. While Amazon’s airframe might not be able to fly in a true fixed-wing mode, other quadcopter fixed-wing hybrids have been proposed which can. For example, the SkyProwler looks like a standard fixed-wing aircraft but it has retractable arms with propellers that turn it into a quadcopter. Like the tilt-rotor and the thrust vectoring approaches, these hybrids add significant mechanical complexity.

There are other approaches that merge the desired abilities in a way that do not add mechanical complexity. For example, the ducted fan or shrouded-fan aircraft is basically a ducted fan with control surfaces on the inside of the duct and a small fuselage [86]. It can take off and land vertically, and the duct itself generates lift in high-speed flight. While this design does not add mechanical complexity, the fuselage is typically small, which limits the potential payload. Some examples of ducted fan aircraft are the Hovereye [19, 91] and SLADe [90].

Another approach to merge the positive characteristics is the tailsitter which is a type of fixed-wing aircraft that can vertically take-off and land on its tail and transition into level flight. Like ducted fan aircraft, a tailsitter does not require additional moving components to achieve its VTOL capability. In addition, it can look more like a normal fixed-wing aircraft and carry a larger payload than ducted fan aircraft. However, this mechanical simplicity comes at the cost of increased control complexity during the transitions. For
example, quadcopter fixed-wing hybrids are able to transition into high-speed flight simply by ramping up the forward facing propellers while simultaneously ramping down the vertical propellers. On the other hand, a tailsitter needs to transition the entire aircraft from a vertical to horizontal orientation.

Tailsitters were first studied in depth in the 1950’s where several manned prototypes, including the Convair XF-Y1 and the Lockheed XF-V1 shown in Figure 1.2, were built and tested. However, pilots found controlling the aircraft difficult, especially while landing, and development on these aircraft was quickly shut down. Boeing restarted tailsitter research in the 1990’s with the development of the Boeing Heliwing. However, development was canceled after the first, and only prototype, crashed while landing during a flight test.

Recently interest has grown significantly in tailsitters due to the major advances in small unmanned aircraft, and several unmanned tailsitters have been developed. These tailsitters include research platforms, like the University of Sydney and Soncaom Pty Ltd.’s T-Wing shown in Figure 1.3a, commercial products for government and industry like AeroVironment’s SkyTote [31] which was canceled in 2010 and is shown in Figure 1.3b, Aerovel’s Flexrotor, and several other variants. There is also public interest in tailsitters which has led to several successful Kickstarter tailsitter projects like the Quadshot [36, 37, 97] in 2011.
and the X PlusOne [30] in 2014. In addition, Google’s Project Wing used a tailsitter in its early stages [78].

This surge of interest has led to a wide variety of tailsitter variants beyond the typical propeller on the top of the aircraft. Some common designs include having a propeller on each side of the main wing like the T-Wing, counter-rotating propellers on the nose of the aircraft like the SkyTote, and tailsitter-quadcopter hybrids like the X PlusOne.

![Figure 1.3: Unmanned tailsitter prototypes](image)

Martin UAV is working on a different tailsitter design called the Vertical Bat (V-Bat) which is a Tier 2 UAV. As shown in Figure 1.4, the V-Bat looks like a typical fixed-wing aircraft, however, it has a ducted fan which also acts as its landing gear. Not only does the duct increase the thrust generated by the propeller, it also allows people to be near it safely, unlike other large tailsitters such as Aerovel’s Flexrotor. On the other hand, having the propeller at the rear of the aircraft makes it inherently unstable in hover flight. Martin UAV partnered with Brigham Young University to research effective control schemes for this platform.

This chapter is organized in the following manner. Section 1.1 gives an overview of common tailsitter control architecture and our approach. Following this, a summary of our main contributions is in Section 1.2. Finally, an overview of the rest of the dissertation is given in Section 1.3.
1.1 Control Architecture

There are several control architectures that are commonly used for tailsitters. First, a single controller can control the aircraft in every flight mode. While having a single controller removes any problems that might occur when switching between control modes, these types of controllers tend to be more complex and have significant performance tradeoffs. For example, Johnson et al. developed a unified adaptive control scheme which consists of command estimates generated by dynamic inversion, a PD loop to reduce tracking errors, a neural network that estimates the model errors, and pseudocontrol hedging to reduce the effect of actuator saturation on the estimated neural network \cite{51}. Using only a simple linear model of the aircraft in hover, Johnson’s controller is able to control the aircraft in every flight regime. However, the slowest groundspeed they achieved was approximately 3 m/s. Bapst developed another unified controller which generates a desired acceleration by modeling the position and altitude errors as second-order systems with tunable time constants and damping ratios \cite{13,70}. The desired acceleration is combined with the estimated aerodynamic acceleration which provides the acceleration the propulsion system needs to generate from which attitude and thrust commands are generated. Low-level controllers track the attitude and thrust commands.

Another approach is to have a separate controller for the hover and level flight modes while the transitions utilize one of these controllers. Naldi and Marconi provide support for this approach in \cite{83} where they define the hover flight envelope as the region where the aerodynamic forces can be considered a small disturbance and the level flight envelope as
the region where the angle of attack is within the non-stall region and the airspeed is above a nominal value. With these definitions they show that, for some airframes, these envelopes can overlap. This suggests that a valid hover-to-level (HTL) transition approach is to use the hover controller to accelerate until the aircraft enters the shared envelope. Then the level controller takes over and accelerates to the nominal airspeed. The level-to-hover (LTH) transition could be performed in a similar manner. Frank et al. use this approach and use their LQR hover position controller to perform the HTL transition [38]. Myrand-Lapierre et al. also use this approach but their active controller arbitrarily switches when the pitch angle passes 50°. Myrand-Lapierre’s approach illustrates a common issue with this method. When should the active controller switch? Most hover or level controllers cannot handle the entire transition region so the switch must occur in an intermediate flight condition which is not ideal.

The most common approach is to use separate controllers for each flight mode. This allows the controllers to be specifically designed to control the aircraft in the specified flight regime allowing the proper definitions, including heading, to be used. However, switching between control schemes can present some challenges due to the switch itself and the wide range of possible initial conditions that might occur especially after a transition. One way of handling the switching is by using hybrid control. Casauer et al. developed a robust control scheme with four flight modes: hover, level, transition, and recovery, for a tailsitter where each flight mode has a controller with a stability guarantee [24–26]. Hybrid control techniques are used to guarantee the transition between the modes will not cause the aircraft to go unstable.

In many cases, a single attitude controller is shared with each flight mode. Some examples include Michini’s L1 adaptive attitude controller [79], the T-Wing’s angular rate controller that stabilizes the aircraft [10], and a MIMO attitude rate controller which uses structured singular value theory based on a linearized model of the aircraft at a trim speed of 9 m/s [12]. Thrust controllers are also commonly shared. For example, Knoebel’s controller uses the same adaptive thrust controller in each flight mode [58].

Control logic is needed to determine which flight mode is active at any given time. Typically, the flight mode behavior is structured as a state machine. For example, Knoebel
uses a state machine with hover, HTL transition, LTH transition, and level flight states [58]. Osborne describes a similar state machine in [89]. Stone describes the T-Wing architecture in [104] which has states for all of the flight modes as well as states allowing an adjustable amount of manual control.

1.1.1 Vertical Bat Control Architecture

The V-Bat uses a hierarchical control structure which is shown in Figure 1.5. As can be seen, the control architecture consists of a waypoint manager (WM), flight mode controller (FMC), flight mode specific controllers, and an attitude controller. The remainder of this section will briefly describe each component. More details can be found in [11].

Waypoint Manager

The V-Bat’s flight plan is an ordered list of waypoints starting with the take-off followed by an arbitrary number of hover and level waypoints and ending with a landing waypoint. The WM’s purpose is to keep track of the current waypoint and determine when to switch to the next waypoint. It sends the current waypoint type and the waypoint commands, such as position and velocity, to the FMC.
Flight Mode Controller

The FMC determines how the V-Bat will reach the current waypoint. When the WM switches to a new waypoint, the FMC determines if the aircraft needs to transition and, if so, where the transitions should occur. It then generates a series of sub-waypoints detailing the flight plan so the aircraft can achieve the waypoint. It maintains the flight mode state machine, keeps track of the current sub-waypoint, and sends the desired position, attitude, and/or orientation to the current flight mode controller. The V-Bat’s FMC is similar to the one discussed in |89|.

Flight Mode Dependent Controllers

Each flight mode has its own controller. The controllers take the desired states from the FMC and generate attitude and throttle commands. Chapter 5 will discuss the hover flight controller, Chapter 6 will discuss the level flight controller, and Chapter 7 will discuss the transitions. The literature reviews are included in the relevant chapters.

Attitude Controller

A common attitude controller is used by each controller. The attitude controller takes the attitude command and generates the aileron, elevator, and rudder commands. These commands are then mixed to generate the commands for each control vane. The attitude control system is discussed in Chapter 4.

1.2 Contributions

The main objective of this work was to develop control schemes, for all of the flight modes, that do not require a high-fidelity model of the aircraft. In addition, we desired that the controllers could be tuned in the field. These objectives guided the choice of the type of control algorithms used. The main contributions in this work are summarized below without context. The context for each contribution is included in the literature review in the relevant chapter. The main contributions in this work are:
• The resolved Euler angle attitude error method that computes the attitude error in a way that generates the desired trajectories for a hovering tailsitter which the standard approach in the literature does not do,

• A hybrid attitude control system, based on the resolved Euler angle attitude error, that provides almost global asymptotic stability,

• A hover control system that can handle being initialized in a wide variety of initial conditions,

• A novel nonlinear total energy controller for the altitude and the airspeed during fixed-wing flight that provides asymptotic stability even with inaccurate thrust and drag models,

• A simple constant altitude constant acceleration transition trajectory that can be incrementally tested and tuned,

• An energy based transition controller that requires almost no knowledge of the aircraft’s aerodynamics yet can perform near constant altitude fully autonomous transitions.

1.3 Organization

This dissertation is organized in the following manner. Chapter 2 briefly describes the main coordinate frames and attitude representations used throughout. Chapter 3 derives the kinematic and dynamic model for the V-Bat prototype. The hover control system is developed in Chapter 5, and the level control system is derived in Chapter 6. Several transition control schemes are developed in Chapter 7. Finally, conclusions are presented in Chapter 8. The V-Bat aerodynamic and airframe parameters are in Appendix A, and several attitude error algorithms are collected in Appendix B. Details about the hardware in the loop simulator are include in Appendix C.
Chapter 2. Coordinate Frames and Attitude Representations

This chapter describes the main coordinate frames and attitude representations used in this dissertation. First, the coordinate frames are defined in Section 2.1. Following this, the attitude representations are described in Section 2.2.

2.1 Coordinate Frames

The inertial, vehicle, body, and desired coordinate frames are used throughout this dissertation. Each coordinate frame consists of an origin and three orthogonal unit vectors which define the axes. The $x$-axis is denoted by $\mathbf{i}$, the $y$-axis by $\mathbf{j}$, and the $z$-axis by $\mathbf{k}$. A superscript will denote the coordinate frame. The inertial frame, $\mathcal{I}$, is centered at an arbitrary location and $\mathbf{i}^i$ points north, $\mathbf{j}^i$ points east, and $\mathbf{k}^i$ points down towards the center of the Earth. The vehicle frame’s, $\mathcal{V}$, origin is the center of mass of the aircraft and its axes are aligned with the inertial frame. In other words, $\mathbf{i}^v$ points north, $\mathbf{j}^v$ points east, and $\mathbf{k}^v$ points down. The body frame, $\mathcal{B}$, has the same origin as the vehicle frame but its unit vectors are aligned with the aircraft. We use the standard typically used for fixed-wing aircraft where $\mathbf{i}^b$ points out the nose of the aircraft, $\mathbf{j}^b$ points out the right wing, and $\mathbf{k}^b$ completes the right hand rotation and points out the belly of the aircraft. The desired frame, $\mathcal{D}$, also shares an origin with the vehicle and body frame but its axes are aligned with the desired attitude.

2.2 Attitude Representations

At the most fundamental level the attitude describes how one coordinate frame needs to be rotated such that its axes align with a second coordinate frame. While there are many different ways to represent the attitude (see [28, 96] for a wide variety of attitude representations), we use Euler angles and quaternions.
2.2.1 Euler Angles

Euler angles are perhaps the most common attitude representation for aircraft and represent the attitude as a sequence of three consecutive rotations. For example, the 3-2-1, or ZYX, Euler angle sequence, which is commonly used for fixed-wing aircraft, is as follows. First, the vehicle frame is rotated about $k^v$ by $\psi$ into the vehicle-1 frame. The vehicle-1 frame is then rotated about $j^v_1$ by $\theta$ into the vehicle-2 frame. Finally, the vehicle-2 frame is rotated into the body frame by rotating about $i^v_2$ by $\phi$. The rotation matrices for these operations are

$$R^v_1(\psi) \triangleq \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R^v_2(\theta) \triangleq \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R^b_v(\phi) \triangleq \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}$$

These rotation matrices can be combined to get the rotation matrix from $V$ to $B$ which is

$$R^b_v(\phi, \theta, \psi) = R^b_v(\phi)R^v_2(\theta)R^v_1(\psi)$$

$$= \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{bmatrix}, \tag{2.1}$$

where $c$ represents cosine and $s$ represents sine.

Euler angles are a minimal attitude representation in that they only require three parameters to describe a rotation. However, (2.1) demonstrates the main limitation of Euler
angles: gimbal lock. For example, if $\theta = 90^\circ$ then (2.1) becomes

$$R_b^\psi(\phi, 90^\circ, \psi) = \begin{bmatrix}
0 & 0 & -1 \\
-s_\phi c_\psi - c_\phi s_\psi & s_\phi s_\psi + c_\phi c_\psi & 0 \\
c_\phi c_\psi + s_\phi s_\psi & c_\phi s_\psi - s_\phi c_\psi & 0
\end{bmatrix},$$

which can be simplified to

$$R_b^\psi(\phi, 90^\circ, \psi) = \begin{bmatrix}
0 & 0 & -1 \\
\sin(\phi - \psi) & \cos(\phi - \psi) & 0 \\
\cos(\phi - \psi) & -\sin(\phi - \psi) & 0
\end{bmatrix}.$$

In this situation, called gimbal lock, it is impossible to distinguish between $\phi$ and $-\psi$ which can cause control and estimation algorithms to fail. While gimbal lock should never occur for most aircraft, given the right coordinate frame definitions and Euler angle sequence, it will obviously occur for a tailsitter. It is possible to avoid gimbal lock by using different Euler angle sequences for level and hover flight but doing so creates additional problems related to when the sequence switching should occur and ensuring that the switch will be well behaved. Fortunately, these problems can be avoided by using quaternions.

### 2.2.2 Quaternions

A quaternion is a hyper-imaginary number with three imaginary components denoted by $\eta = \eta_0 + \eta_1 i + \eta_2 j + \eta_3 k$, where the imaginary components satisfy

$$i^2 = j^2 = k^2 = ijk = -1.$$  \hfill (2.2)

A quaternion can also be represented in vector form by $\eta = [\eta_0 \; \eta_1 \; \eta_2 \; \eta_3]^\top$ or $\eta = [\eta_0 \; \bar{\eta}]^\top$, where $\bar{\eta} \triangleq [\eta_1 \; \eta_2 \; \eta_3]^\top$. 
If a quaternion has unit norm, $\|\eta\| = 1$, then it can be used to represent a rotation about the unit vector $\nu$ by the angle $\gamma$ where

$$\nu = \frac{\bar{\eta}}{\|\bar{\eta}\|} \quad (2.3)$$

$$\gamma = 2\cos^{-1}(\eta_0). \quad (2.4)$$

The quaternion that describes the rotation for a given angle and axis of rotation is given by

$$\eta = \begin{bmatrix} \cos\left(\frac{\gamma}{2}\right) \\ \\ \sin\left(\frac{\gamma}{2}\right)\nu \end{bmatrix}. \quad (2.5)$$

A quaternion can be converted to the corresponding rotation matrix by \cite{16}

$$R_b^k(\eta) = \begin{bmatrix} 2(\eta_0^2 + \eta_1^2) - 1 & 2(\eta_1\eta_2 + \eta_0\eta_3) & 2(\eta_1\eta_3 - \eta_0\eta_2) \\ 2(\eta_1\eta_2 - \eta_0\eta_3) & 2(\eta_0^2 + \eta_2^2) - 1 & 2(\eta_2\eta_3 + \eta_0\eta_1) \\ 2(\eta_1\eta_3 + \eta_0\eta_2) & 2(\eta_2\eta_3 - \eta_0\eta_1) & 2(\eta_0^2 + \eta_3^2) - 1 \end{bmatrix}. \quad (2.5)$$

The conversion from a quaternion to an Euler angle sequence is found by setting the corresponding rotation matrices equal to each other. For example, setting the ZYX Euler angle sequence rotation matrix (2.1) equal to (2.5) and solving for the Euler angles provides

$$\phi = \tan^{-1}\left(\frac{2(\eta_0\eta_1 - \eta_2\eta_3)}{2\eta_0^2 + 2\eta_3^2 - 1}\right)$$

$$\theta = \sin^{-1}(2(\eta_0\eta_2 - \eta_1\eta_3))$$

$$\psi = \tan^{-1}\left(\frac{2(\eta_0\eta_3 - \eta_1\eta_2)}{2\eta_0^2 + 2\eta_1^2 - 1}\right).$$

Euler angles can also be easily converted into quaternions. For instance, the ZYX Euler angles become

$$\eta_0 = \cos\left(\frac{\psi}{2}\right) \cos\left(\frac{\theta}{2}\right) \cos\left(\frac{\phi}{2}\right) + \sin\left(\frac{\psi}{2}\right) \sin\left(\frac{\theta}{2}\right) \sin\left(\frac{\phi}{2}\right)$$

$$\eta_1 = \cos\left(\frac{\psi}{2}\right) \cos\left(\frac{\theta}{2}\right) \sin\left(\frac{\phi}{2}\right) - \sin\left(\frac{\psi}{2}\right) \sin\left(\frac{\theta}{2}\right) \cos\left(\frac{\phi}{2}\right)$$

$$\eta_2 = \cos\left(\frac{\psi}{2}\right) \sin\left(\frac{\theta}{2}\right) \cos\left(\frac{\phi}{2}\right) + \sin\left(\frac{\psi}{2}\right) \cos\left(\frac{\theta}{2}\right) \sin\left(\frac{\phi}{2}\right)$$

$$\eta_3 = \sin\left(\frac{\psi}{2}\right) \cos\left(\frac{\theta}{2}\right) \cos\left(\frac{\phi}{2}\right) - \cos\left(\frac{\psi}{2}\right) \sin\left(\frac{\theta}{2}\right) \sin\left(\frac{\phi}{2}\right).$$
There are two different ways a vector can be rotated by a quaternion. The first is to convert the quaternion to a rotation matrix, using (2.5), and use it to rotate the vector. A faster approach is to use the quaternion directly by

\[ \mathbf{v}' = \mathbf{v} \otimes \begin{bmatrix} 0 \\ \mathbf{v} \end{bmatrix} \otimes \eta^{-1}, \]  

where \( \mathbf{v} \) is the vector to be rotated, the quaternion composition is

\[ \eta' \otimes \eta \equiv \begin{bmatrix} \eta_0' \eta_0 - \tilde{\eta}' \cdot \tilde{\eta} \\ \eta_0' \tilde{\eta} + \eta_0 \tilde{\eta}' - \tilde{\eta}' \times \tilde{\eta} \end{bmatrix}, \]

\( \times \) is the cross product and the inverse of a quaternion is

\[ \eta^{-1} = \eta^* = \begin{bmatrix} \eta_0 \\ -\tilde{\eta} \end{bmatrix}. \]

Note that the inverse rotation is given by the complex conjugate, or the inverse, of the quaternion. Quaternion composition is also used to generate a single quaternion that describes successive rotations. For example, suppose we want to rotate \( \mathbf{v} \) by \( \eta \) followed by \( \eta' \). This could be accomplished by using (2.6) twice, once for \( \eta \) and once for \( \eta' \), or by rotating the vector a single time by the quaternion \( \eta'' = \eta' \otimes \eta \).

The quaternion attitude representation is more computationally efficient than Euler angles. In addition, quaternions do not have a singularity so they can be used for every flight mode. However, quaternions are not perfect. First, they are not intuitive. It is not difficult to mentally rotate an object given a set of Euler angles but it is very difficult to do with the equivalent quaternion. Second, quaternions provide a double coverage of \( SO(3) \) in that \( \eta \) and \( -\eta \) describe the same rotation. Typically this is handled by keeping the scalar component, \( \eta_0 \), positive by negating \( \eta \) if \( \eta_0 \) ever becomes negative. However, this approach will not work in all cases because it introduces a discontinuity. For example, Mayhew et al. describe an attitude controller where this correction will break the stability guarantee [76]. A more detailed explanation of this problem is given in [28].
For more details about quaternions see [64] which gives an excellent presentation on the development and uses of quaternions, [110] which delves into quaternion algebra, or [100] which develops quaternion operations as matrix operations and provides one of the best introductions to quaternions from a linear algebra standpoint.
Chapter 3. Vertical Bat Kinematics and Dynamics

There are several common approaches for developing a model of a small UAV. The first approach is to use a simple mass-force model and a look-up table for the forces and moments. In this approach, the forces are precomputed for a wide variety of flight conditions using wind tunnel tests [19], numerical methods such as computational fluid dynamics (CFD) [12], or data from flight tests. Interpolation is used to estimate the forces and moments for flight conditions that are not in the database. This approach can be highly accurate and simulators using this method are computationally efficient. However, it does not provide any intuition on how the various components of the aircraft influence the overall performance. In addition, the database needs to be remade if any aspect of the design is changed besides simple scaling, and building the database can be very time intensive. For example, Stone modeled the T-Wing's aerodynamics by using a full azimuthal blade-element solution for the propellers and a fixed-wake panel method model of the airframe. The aerodynamic force and moment coefficients and their derivatives are precomputed for over 4300 flight conditions [102, 103, 107]. Precomputing a database of aerodynamic coefficients has also been used for a ducted fan aircraft where the coefficients are computed by a boundary element code based on the angle of attack and the ratio between the forward velocity and inflow velocity to the duct [12].

The second modeling approach is to use system identification. First, data is collected through wind tunnel tests, CFD, and flight tests. Then system identification techniques are used to develop a model and compute the coefficients such as in [57]. These models do not need to interpolate between known points, unlike the database approach, and require less memory to store. However, system identification techniques require that all of the dynamics are excited during the tests. In any case, system identification techniques can be used to generate data or estimate parameters for the other modeling approaches. For example,
developed transfer functions of an aircraft in level flight using system identification
techniques.

The third approach is to analytically develop the model. Typically, the aircraft is
divided into separate components, such as wings, duct, and control surfaces, and a model is
developed for each piece. The advantage of this approach is that if the aircraft design changes,
only the corresponding submodels need to be updated. However, it does not take into account
the interaction between the submodels. This approach has been used for ducted fan aircraft
[27, 51], normal fixed-wing aircraft [16], and many others [93]. Because current analysis
techniques suffer at high angles of attack [48], model parameters are typically computed
from, or at least checked by, data collected through CFD, wind tunnel tests, or flight tests.

Hybrid approaches are also possible. For example, [95] discusses a real-time simulator
for agile aircraft. In this simulator, the model is broken up into several submodels which are
developed through a combination of look-up tables and aerodynamic first principles. Some
components, such as the wings, are modeled as a series of small strips which are looked at
individually while other components, such as the tail, are modeled as a single piece.

It is possible is to develop models for each flight regime. For example, Oishi and
Tomlin create separate Euler-Lagrange-based models for the McDonell Douglas Harrier’s
hover, standard, and transition modes [87]. While this approach leads itself well to control
design it typically does not cover the full range of flight conditions that might be encountered
[17, 87]. Therefore, this approach is not well suited for a full simulator.

Due to the available resources, we decided to analytically develop separate models
for the various components of the V-Bat. The model parameters are either based on the
analysis or by fitting the models to data obtained through wind tunnel tests and flight tests.
The data collection procedure is described in [33].

This chapter is organized as follows. Section 3.1 describes the kinematics, and Section
3.2 derives the dynamics for the propeller, control vanes, duct, and the rest of the airframe.
Some issues with the model and potential solutions are discussed in Section 3.3. All of the
model parameters described in this chapter are listed in Appendix A.
3.1 Kinematics

The quaternion-based six-degree-of-freedom kinematic equations for a rigid body are

\[ \dot{p} = R_v^b(\eta)v \] (3.1)
\[ \dot{v} = -\omega \times v + \frac{f}{m} \] (3.2)
\[ \dot{\eta} = \Omega(\omega)\eta \] (3.3)
\[ \dot{\omega} = J^{-1}(-\omega \times J\omega + m), \] (3.4)

where \( p = [p_n \ p_e \ p_d]^\top \) are the inertial north, east, and down positions, \( v = [u \ v \ w]^\top \) are the body frame velocities, \( \eta = [\eta_0 \ \eta_1 \ \eta_2 \ \eta_3]^\top \) is a unit quaternion representing attitude, \( \omega = [p \ q \ r]^\top \) is the angular velocity measured about \( i^b, j^b, \) and \( k^b, f = [f_x \ f_y \ f_z]^\top \) are the net forces in \( B, m = [l \ m \ n]^\top \) are the net moments about the body axes,

\[
R_v^b(\eta) = \begin{bmatrix}
2\eta_0^2 + 2\eta_1^2 - 1 & 2(\eta_1\eta_2 - \eta_3\eta_0) & 2(\eta_1\eta_3 + \eta_2\eta_0) \\
2(\eta_1\eta_2 + \eta_3\eta_0) & 2\eta_0^2 + 2\eta_2^2 - 1 & 2(\eta_2\eta_3 - \eta_1\eta_0) \\
2(\eta_1\eta_3 - \eta_2\eta_0) & 2(\eta_2\eta_3 + \eta_1\eta_0) & 2\eta_0^2 + 2\eta_3^2 - 1
\end{bmatrix}
\]

is the rotation matrix describing the rotation from \( B \) to \( V, \)

\[
\Omega(\omega) \triangleq \frac{1}{2} \begin{bmatrix}
0 & -p & -q & -r \\
p & 0 & r & -q \\
q & -r & 0 & p \\
r & q & -p & 0
\end{bmatrix},
\]

\[
J = \begin{bmatrix}
J_x & 0 & -J_{xz} \\
0 & J_y & 0 \\
-J_{xz} & 0 & J_z
\end{bmatrix}
\]
is the moment of inertia which takes into account the V-Bat’s symmetry about the \( i^b - k^b \) plane, and \( m \) is the mass. Note that some sources, such as [16], state that a correction

18
The gravitational force is modeled as a force proportional to the mass acting on the center of mass in the \( \hat{k} \) direction. In the body frame it is

\[
f_g = m \mathbf{R}_b^c(\eta)^\top \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = mg \begin{bmatrix} 2(\eta_1 \eta_3 - \eta_2 \eta_0) \\ 2(\eta_2 \eta_3 - \eta_1 \eta_0) \\ 2\eta_0^2 + 2\eta_3^2 - 1 \end{bmatrix},
\]

where \( g \) is the acceleration due to gravity. Because the gravitational force acts on the center of mass, it does not generate a moment.
3.2.2 Propeller

While there are several methods to model the thrust generated by a propeller, many assume that the induced velocity is known \[81, 109\] or have very restrictive assumptions on the relative direction of the airspeed \[16\]. One method that provides the induced velocity and has been used successfully for modelling the propellers in several ducted fan aircraft was originally developed to model the rotors of large helicopters \[43, 99\]. Johnson et al. showed that this model is accurate enough to model the propeller in a small ducted fan even though the duct is not explicitly taken into account \[50\]. The following propeller model is derived using Johnson’s model as a starting point and then the duct is explicitly taken into account.

**Model Derivation**

The velocity at the propeller disk is

\[
v_b = u_{\text{prop}} + \frac{2}{3} \omega_p r_p \left( \frac{3}{4} K_{\text{twist}} \right),
\]

where the first term,

\[
u_{\text{prop}} = u - u_{\text{wind}},
\]

assumes that only the airspeed along \( \mathbf{i}_b \) reaches the propeller, the second term accounts for the relative velocity caused by the propeller spinning at an angular velocity of \( \omega_p \), \( r_p \) is the radius of the propeller, \( K_{\text{twist}} \) is the twist of the blade, and \( \mathbf{v}_{\text{wind}} = [u_{\text{wind}} \ v_{\text{wind}} \ w_{\text{wind}}]^T \) is the wind speed measured along \( \mathbf{i}_b, \mathbf{j}_b \), and \( \mathbf{k}_b \).

The net force generated by the propeller is

\[
f_{\text{prop}} = \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix},
\]

where the thrust is given by

\[
T = \frac{1}{4} (v_b - v_i) \omega_p r_p^2 \rho a_c b_c p,
\] (3.6)
$v_i$ is the induced velocity, $\rho$ is the air density, $a_0$ is the lift curve slope, $b_p$ is the number of blades, and $c_p$ is the blade chord. The farfield velocity is

$$v_f = \sqrt{(v - v_{\text{wind}})^2 + (w - w_{\text{wind}})^2 + (u_{\text{prop}} + v_i)^2}, \quad (3.7)$$

and the induced velocity is given by [50]

$$v_i = \frac{T}{2\rho \pi r^2 v_f}. \quad (3.8)$$

There is no closed form expression for $T$ and $v_i$ given the airspeed and $\omega_p$. Instead, (3.6)-(3.8) must be solved iteratively. First, an initial estimate of the induced velocity, $\hat{v}_i$, is selected. Then the estimated thrust, $\hat{T}$, and farfield velocity, $\hat{v}_f$, are computed using (3.6) and (3.7) and are used to check $\hat{v}_i$ by computing, with (3.8), the induced velocity, $\hat{v}_i'$, generated by $\hat{T}$ and $\hat{v}_f$. The induced velocity estimate is correct if $\hat{v}_i \approx \hat{v}_i'$, otherwise a new induced velocity estimate is formed by

$$\hat{v}_i^+ = \hat{v}_i + k_{vi}(\hat{v}_i' - \hat{v}_i),$$

where $k_{vi} > 0$ is a scaling factor, and the process is repeated. While this method is fast, especially if the initial $\hat{v}_i$ is the induced velocity found the last time the model was run, it is not guaranteed to converge. In this case, a brute force approach can find $v_i$ by computing $\hat{v}_i'$ for a wide range of $\hat{v}_i$ and selecting the $\hat{v}_i$ with the lowest absolute error $|\hat{v}_i - \hat{v}_i'|$.

The net moment,

$$\mathbf{m}_{\text{prop}} = \mathbf{m}_{\text{ind}} + \mathbf{m}_{\text{gyro}},$$

is generated by the power applied by the motor on the air and a gyroscopic moment caused by the propeller changing orientation. The power applied by the motor onto the air is given by [109]

$$P_{\text{ind}} = v_i T, \quad (3.9)$$

and the moment exerted on the plane due to this power is

$$\mathbf{m}_{\text{ind}} = \frac{P_{\text{ind}}}{\omega_r}. $$
Normally, the gyroscopic moment is negligible but it might become significant when the V-Bat is performing an aggressive maneuver such as a transition. Assuming \( \dot{\omega}_r \) is small, the gyroscopic moment can be modeled as \cite{50}

\[
\mathbf{m}_{\text{gyro}} = n_p \omega_p J_p \begin{bmatrix}
0 \\
-\tau \\
p
\end{bmatrix},
\]

where \( J_p \) is the propeller’s moment of inertia.

Instead of allowing the propeller’s angular velocity to instantaneously change, it is modeled as a second-order dynamic system with a transfer function of

\[
\Omega_p(s) = \frac{250}{s^2 + 15 + 250 \Omega_p^d(s)}.\]

**Parameter Identification**

While many of the propeller parameters, such as the radius, number of blades, and chord, are easily measured, some are more difficult. Several bench tests and wind tunnel tests were performed on the V-Bat prototype to gather the data needed to identify the remaining parameters.

In the first test the throttle command was slowly increased from 0% to 100% and then decreased to 0%. The propeller’s angular velocity was recorded throughout the entire test. The second test measured the induced velocity and the thrust generated at seven different angular velocities. Finally, the third test measured the thrust generated at 10% to 100% throttle at 10% increments and at an airspeed of 0, 13, 27, 38, 42.6, and 56 miles per hour. The angle of attack and side slip angles were both 0\(^\circ\) during these tests.

A parametric curve was fit to the angular velocity versus throttle command data and the resulting model is

\[
p_{\text{rpm}} = -3892\delta_t^2 + 11830\delta_t + 110, \quad (3.10)
\]

where \( \delta_t \in [0, 1] \) is the throttle command and \( p_{\text{rpm}} \) is the angular velocity in rotations per minute. The measured angular velocity and the predicted angular velocity from the model
are shown in Figure 3.1. There are two major problems with this model. The first is that the angular velocity does not vary with the battery voltage such as in [58, 80]. The second problem is that the angular velocity does not depend on the incoming airspeed. Extensive bench and wind tunnel tests are needed to correct these issues.

The propeller’s twist and lift curve slope were found by using an optimization algorithm that minimized the error between the measured thrust and the predicted thrust over the entire range of measured throttle commands and airspeeds. Note that the moment and induced velocity data were not included in the optimization cost function. As can be seen in Figures 3.2a-3.2d, this model matches the static propulsion tests and reasonably matches the generated moment and induced velocity. However, the predicted thrust drops off too quickly as the airspeed increases and when $\delta_t$ is low. The model’s accuracy can be improved by taking into account the effect of the duct.

**Improved Model**

An ideal ducted fan generates twice as much thrust as the propeller by itself. Using the Horn method [81], (3.6) is scaled by the thrust augmentation factor, $0 \leq k_{aug} \leq 1$, to obtain

$$T = \frac{1 + k_{aug}}{4}(v_b - v_i)\omega_p r_p^2 \rho a_o b_p c_p.$$  \hspace{1cm} (3.11)
Figure 3.2: Optimized propeller model not taking into account the effect of the duct

Note that \( k_{\text{aug}} = 0 \) models an open rotor while \( k_{\text{aug}} = 1 \) models an ideal ducted fan. The power applied by the motor to the air only depends on the thrust provided by the rotor, not the duct. Because of this (3.9) becomes

\[
P_{\text{ind}} = \frac{v_i T}{1 + k_{\text{aug}}}. \tag{3.12}
\]

The parameter optimization was rerun and included \( k_{\text{aug}} \) as one of the unknown parameters. The measured data and the new model predictions are shown in Figures 3.2a-3.2d. Note that the updated model matches the measured data better than the original model and the estimated twist of the blade is a much more realistic value. However, the
predicted thrust still drops off too quickly and becomes negative for large airspeeds and low throttle commands. This is probably caused by the limitations, mentioned earlier, in the angular velocity model (3.10) or by other model inaccuracies. These conditions should rarely, if ever, be encountered in normal flight and should not harm the accuracy of the simulation.

3.2.3 Duct

While the V-Bat’s duct generates significant lift and drag, the duct model only models the momentum drag. The incoming airflow, as shown in Figures 3.4a and 3.4b, arrives at the mouth of the duct with a magnitude of $V_0$, an angle of attack of $\alpha_{\text{duct}}$, and a heading of
(a) Horizontal view of the duct airflow and momentum drag. The incoming airflow is straightened by the duct and is accelerated by the propeller. The resultant airflow leaves the duct and eventually merges with the outside airflow producing the farfield velocity $V_\infty$.

(b) Top down view of the duct airflow and momentum drag

Figure 3.4: Simplified duct airflow

$\beta_{\text{duct}}$. When the airflow enters the duct, the duct walls start to straighten the airflow and align it with the duct as shown by $V_e$ and $\chi$. This exerts a force on the duct, called the momentum drag or ram drag, which attempts to align the duct with the incoming airflow. The magnitude of the momentum drag is significant even at moderate airspeeds and must be taken into account.

The magnitude of the airspeed at the mouth of the duct is

$$V_0 = \sqrt{u_{\text{duct}}^2 + v_{\text{duct}}^2 + w_{\text{duct}}^2},$$

where the relative airspeed at the duct entrance is

$$u_{\text{duct}} = u - u_{\text{wind}}$$

$$v_{\text{duct}} = v - v_{\text{wind}} + rl_{\text{duct}}$$

$$w_{\text{duct}} = w - w_{\text{wind}} + ql_{\text{duct}},$$
and $l_{duct}$ is the distance from the center of gravity to the top of the duct. The angle of attack at the duct entrance is

$$\alpha_{duct} = \tan^{-1}\left(\frac{\sqrt{u_{duct}^2 + w_{duct}^2}}{u_{duct}}\right),$$

and the sideslip is

$$\beta_{duct} = \tan^{-1}\left(\frac{v_{duct}}{w_{duct}}\right).$$

From this the momentum drag can be calculated by [109]

$$f_{d,duct} = k_{duct} \rho A_{duct} (V_0 \cos(\alpha_{duct}) + v_i) V_0 \sin(\alpha_{duct}),$$

where $A_{duct} = \pi r_{duct}^2$ is the area of the duct and $0 \leq k_{duct} \leq 1$ is a scaling factor that determines how much the duct straightens the airflow. A $k_{duct} = 1$ corresponds to an ideal ducted fan where the airflow is completely aligned with the duct and a $k_{duct} = 0$ corresponds to a normal propeller. To simplify the duct and vane dynamics we assume that $k_{duct} = 1$ and that the duct completely straightens the airflow. Note that this is the worst case in terms of the magnitude of the momentum drag.

The momentum drag is rotated into the body frame by

$$f_{duct} = \begin{bmatrix} 0 \\ -f_{d,duct} \sin(\beta_{duct}) \\ -f_{d,duct} \cos(\beta_{duct}) \end{bmatrix},$$

which generates a moment of

$$m_{duct} = \begin{bmatrix} 0 \\ f_{x,duct} l_{duct} \\ -f_{y,duct} l_{duct} \end{bmatrix}. \quad (3.13)$$

Note that (3.13) should use the distance between the center of gravity and the duct’s center of pressure. However, the duct’s center of pressure is unknown, based on our current data, and can change depending on the flight conditions [60]. Typically, the center of pressure for a duct is above its mouth so assuming it is at the mouth overestimates the magnitude of the momentum drag.
3.2.4 Control Vanes

The V-Bat’s main control surfaces are eight control vanes, shown in Figure 3.5a, that are equally spaced throughout the duct and extend past the edge of the duct. The control vane aligned with $-k^b$ is defined to be the first control vane and the numbering, shown in Figure 3.5b, proceeds clockwise when viewed from the back of the duct.

Although each control vane is independently actuated, we artificially couple their motion so the attitude control system only needs to provide an aileron, $\delta_a \in [-1, 1]$, elevator, $\delta_e \in [-1, 1]$, and rudder, $\delta_r \in [-1, 1]$, command. The individual vane commands are obtained by mixing these control signals by

$$
\begin{align*}
\delta_{v1} &= \text{sat}(-\delta_a + \delta_r, -1, 1) \\
\delta_{v3} &= \text{sat}(-\delta_a + \delta_e, -1, 1) \\
\delta_{v5} &= \text{sat}(-\delta_a - \delta_r, -1, 1) \\
\delta_{v7} &= \text{sat}(-\delta_a - \delta_e, -1, 1) \\
\delta_{v2} &= \text{sat}(-\delta_a + \delta_e + \delta_r, -1, 1) \\
\delta_{v4} &= \text{sat}(-\delta_a + \delta_e - \delta_r, -1, 1) \\
\delta_{v6} &= \text{sat}(-\delta_a - \delta_e - \delta_r, -1, 1) \\
\delta_{v8} &= \text{sat}(-\delta_a - \delta_e + \delta_r, -1, 1),
\end{align*}
$$

where $\delta_{vi} \in [-1, 1]$ is the $i$th vane’s command. While the actual control vanes accept pulse width modulation commands generated by these $[-1, 1]$ signals, the dynamics model requires the vane deflection in radians. The commands can be converted to radians by

$$
\theta_{vi}^d = \bar{\theta}_v \delta_{vi},
$$

(3.14)
where $\bar{\theta}_v$ is the maximum vane deflection in radians. The rotation of each control vane is modeled as a second-order dynamic system,

$$V_i(s) = \frac{250}{s^2 + 20 + 250} V^d_i(s),$$

that tracks the desired vane deflection.

Each control vane is modeled as an airfoil with the lift and drag forces given by

$$f_{L,vi} = \frac{1}{2} \rho v^2 v S_v \theta_v C_{L,v}$$
$$f_{D,vi} = \frac{1}{2} \rho v^2 v S_v \theta_v C_{D,v},$$

where $v_v = u_{duct} + v_i$ is the velocity of the airflow over the control vanes, $S_v$ is the area of the vane, $C_{L,v}$ is the lift slope coefficient, and $C_{D,v}$ is the drag slope coefficient. Note that the assumption that the duct fully straightens the airflow implies that the vane’s angle of attack only depends on the vane’s deflection. Therefore the drag force for each vane is aligned with $i^b$, providing

$$f_{x,vane} = -\sum_{i=1}^{8} |f_{D,vi}|.$$

The forces in $j^b$ and $k^b$ depend on the lift and are

$$f_{y,vane} = \sum_{i=1}^{8} \cos \left((i - 1) \frac{\pi}{4}\right) f_{L,vi}$$
$$f_{z,vane} = \sum_{i=1}^{8} \sin \left((i - 1) \frac{\pi}{4}\right) f_{L,vi}.$$

The net moment is

$$m_{vane} = \begin{bmatrix} -\frac{l_{vane}}{2} \sum_{i=1}^{8} f_{D,vi} \\ -d_{vane,cp} f_{z,vane} \\ -d_{vane,cp} f_{y,vane} \end{bmatrix},$$

where $l_{vane}$ is the distance from the center of the duct to the vane’s center of pressure and $d_{vane,cp}$ is the distance from the V-Bat’s center of gravity to the vane’s center of pressure.
The vane lift and drag coefficients were found by fitting wind tunnel data to the model. In these tests the motor was powered by a bench power supply that was only able to power the motor at 35% throttle due to current limitations. The aileron command was swept from 0 to 1 and the resultant moment was recorded. Following the aileron test, the rudder was tested in an identical manner. The lift and drag coefficients were selected to minimize the absolute difference between the model and the data. The measured moments and the model predictions are shown in Figures 3.6a and 3.6b. Notice that the moment is linear with respect to the vane deflection until 25 degrees. Above 25 degrees the moment does not increase significantly but the drag continues to increase. Because of this, the vanes are limited to a 30 degree deflection.

Any deflection of the control vanes can cause a significant reduction in the net thrust. This reduction is caused not only by the drag on the control vanes but by the duct being ‘choked’ or blocked. Separating these effects is impossible with the data we have but the model generally matches the overall decrease in thrust by modeling it as drag as shown in Figure 3.6d.

### 3.2.5 Airframe

The airframe model accounts for the rest of the significant forces and moments such as the lift and drag generated by the wings as well as the duct. It would also include the effect of the ailerons on the wings, however, we have found that the control vanes within the duct provide more than enough control authority about $\mathbf{i}$. There are separate models for level and hover flight instead of a single model because of the differences in the wind tunnel tests that were performed. These models are based on a Taylor series expansion of the aerodynamic equations about steady level flight or steady hover flight respectively. The level flight model is used when $V_a \leq 10$ m/s, otherwise the hover model is used. The aerodynamic forces and moments generated by the two models are very similar when the wings are level and the vehicle is oriented into the wind. Therefore, no smoothing needs to be performed while switching between airframe models when transitions occur.
0.5
1
1.5

0
10
20
30
40

0
10
20
30
40

Figure 3.6: Comparison of the predicted and measured forces and moments generated by aileron and rudder commands. The dots in the data lines are the measurements.

Aerodynamic Airframe Model for Level Flight

The incoming airflow can be expressed by the magnitude \( V_a \), the angle of attack \( \alpha \), and the side slip angle \( \beta \). These values can be computed from the airspeed \( u_r \), \( v_r \) and \( w_r \) by

\[
V_a = \sqrt{u_r^2 + v_r^2 + w_r^2}
\]

\[
\alpha = \tan^{-1}\left(\frac{w_r}{u_r}\right)
\]

\[
\beta = \sin^{-1}\left(\frac{v_r}{V_a}\right),
\]

(3.15)

(3.16)
where the relative airspeed is

\[ u_r = u - u_{\text{wind}} \]
\[ v_r = v - v_{\text{wind}} \]
\[ w_r = w - w_{\text{wind}}. \]

The three primary forces are lift, which is perpendicular to the wings and the oncoming airflow, drag, which is aligned with the airflow, and the side force which is in the i^b-j^b plane. Each aerodynamic force can be modeled by

\[ f = \frac{1}{2} \rho S V_a^2 C, \tag{3.17} \]

where \( S \) is the area and \( C \) is a non-dimensional coefficient. By expanding (3.17) and ignoring the higher order terms in the Taylor series expansion, the net force is

\[
\begin{bmatrix}
\frac{1}{2} \rho S V_a^2 \left( C_x + \frac{c}{2V_a} C_{xq} q \right) \\
\frac{1}{2} \rho S V_a^2 \left( C_y + \frac{b}{2V_a} \left( C_{yp} + C_{yr} r \right) \right) \\
\frac{1}{2} \rho S V_a^2 \left( C_z + \frac{c}{2V_a} C_{zq} q \right)
\end{bmatrix}
\tag{3.18}
\]

and the net moment is

\[
\begin{bmatrix}
\frac{1}{2} \rho S b V_a^2 \left( C_l + \frac{b}{2V_a} \left( C_{lp} + C_{lr} r \right) \right) \\
\frac{1}{2} \rho S c V_a^2 \left( C_m + \frac{c}{2V_a} C_{mq} q \right) \\
\frac{1}{2} \rho S b V_a^2 \left( C_n + \frac{b}{2V_a} \left( C_{np} + C_{nr} r \right) \right)
\end{bmatrix}.
\tag{3.19}
\]

Wind tunnel tests and analysis were used to identify the aerodynamic coefficients.

The wind tunnel tests were performed on a 1:10 scale model of the V-Bat and, due to wind tunnel limitations, at an airspeed comparable to 7.2 m/s. While this is significantly slower than the cruise velocity of the V-Bat, the coefficients found should be relatively close. The lift, drag, and pitching moment were measured by sweeping the angle of attack from -100 to 100 degrees and recording the resultant force. The measured lift and drag coefficients are shown in Figure 3.7a and the pitching moment coefficient, \( C_m \), is shown in Figure 3.7b.
Figure 3.7: Level flight lift, drag, and pitching moment coefficients

Note that the V-Bat wings were redesigned to improve the pitch stability after the scale model was created and these tests were performed. These tests were not redone using the new wing design. However, the design change was minor and the coefficients found should be relatively close.

In order to use these data, the range is extended to cover all possible angles of attack by assuming that the coefficients are identical when $\alpha$ is shifted by 180 degrees. While this assumption is invalid, it provides reasonable values outside the measured angle of attack. Parametric models were created for each coefficient. The pitching moment fits a linear model

$$C_m = C_{m0} + C_{ma}\alpha.$$
The lift coefficient is modeled by a piecewise function consisting of seven regions and is shown in Figure 3.7c. The drag coefficient is likewise modeled by a piecewise function. The lift and drag coefficients are rotated into the body frame to obtain

\[ C_x = -C_d \cos(\alpha) + C_L \sin(\alpha) \]
\[ C_z = -C_d \sin(\alpha) - C_L \cos(\alpha). \]

The moment coefficients generated by the lift and drag are also rotated into the body frame to get

\[ C_{x_q} = -C_{d_q} \cos(\alpha) + C_{L_q} \sin(\alpha) \]
\[ C_{z_q} = -C_{d_q} \sin(\alpha) - C_{L_q} \cos(\alpha). \]

The lateral aerodynamics were also tested in the wind tunnel. In these tests the forces and moments were measured as the side slip angle was swept between -10 and 10 degrees. These tests provided data on the side force coefficient \( C_y \), the rolling moment \( C_l \), and the yawing moment \( C_n \). In the range tested, the coefficients fit a linear model.

The wind tunnel tests were unable to provide values for \( C_{x_q}, C_{y_p}, C_{y_r}, C_{z_q}, C_{l_p}, C_{l_r}, C_{m_q}, C_{n_p} \), and \( C_{n_r} \). These parameters could be found by performing frequency sweeps on the control signals while the V-Bat is in level flight. Currently, these parameters are estimated by modeling the vehicle as a flat plate and computing the drag it would experience and the resultant moment.

**Aerodynamic Airframe Model for Hover Flight**

In hover flight there are three main components comprising the net force: lift, drag, and the side force. As shown in Figure 3.8, the side force is in the \( \mathbf{j}_b - \mathbf{k}_b \) plane, the drag force is aligned with the airspeed vector, and the lift force is orthogonal to the drag force and is within the plane formed by the airspeed vector and \( \mathbf{i}_b \). The hover angle of attack definition is identical to the level flight definition (3.15). However, the level flight side slip
definition (3.16) does not make sense for hover flight. Instead, we define

$$\beta_h = \tan^{-1}\left(\frac{v_r}{w_r}\right),$$

which is also shown in Figure 3.8.

The hover airframe force and moment models are identical to the level force and moment models given by (3.18) and (3.19). The body axes force coefficients are formed by

$$C_x = -C_d \cos(\alpha) + C_L \sin(\alpha)$$
$$C_y = C_s \cos(\beta_h) - C_h \sin(\beta_h)$$
$$C_z = -C_h \cos(\beta_h) - C_s \sin(\beta_h),$$

where $C_s$ is the side force coefficient and $C_h$ is a non-dimensional coefficient describing the force in the $j^b$-$k^b$ plane given by

$$C_h = | -C_L \cos(\alpha) - C_d \sin(\alpha) |.$$

Wind tunnel tests were performed to estimate the drag and side force coefficients as well as the rolling moment coefficient. For these tests, the 1:10 scale model V-Bat was positioned in the wind tunnel such that $\alpha = \frac{\pi}{2}$. The heading of the airflow, $\beta_h$, was swept from 0 to 360 degrees and the resulting forces and moments were measured. The drag, side force, and rolling moment were computed, from this data, and the coefficients were calculated. These coefficients are shown in Figures 3.9a-3.9c. Piecewise continuous models
Figure 3.9: Hover flight drag, side force, and rolling moment coefficients. Note that $\phi_a = \beta_h$.

of each coefficient were created and used in the simulator. Note that the hover lift coefficient is identical to the level lift coefficient.

There are several issues that should be further explored. First, the level drag coefficient for $\alpha = \frac{\pi}{2}$ and the hover drag coefficient for $\beta_h = 0$ should be compared. These two situations are identical and should have the exact same drag coefficient. However, the hover drag coefficient is significantly larger than the level drag coefficient. Our current thought is that this difference is caused by the airspeed difference between the tests. As mentioned earlier, the level aerodynamics were tested with an equivalent airspeed of 7.2 m/s while the hover dynamics were tested with an equivalent airspeed of 0.9 m/s. To correct this discrepancy, the hover parameters were scaled so that they produce the same forces that the level dynamics produce when $\beta$ is small. Furthermore, $C_s$, $C_d$, and $C_l$ should depend on $\alpha$ and
\( \beta_h \) instead of just \( \beta_h \). As \( \alpha \) approaches zero, these coefficients should decrease. One way to handle this situation for \( C_l \) is to modify how the moment is being computed. Normally the pitching moment is computed by

\[
M_l = \frac{1}{2} \rho S b V_d^2 C_l. \tag{3.20}
\]

By modifying (3.20) to only depend on \( v_r \) and \( w_r \),

\[
M_l = \frac{1}{2} \rho S b \sqrt{v_r^2 + w_r^2} C_l,
\]

the magnitude of the moment will have the correct behavior. Similar changes can be made for the other forces and moments. An alternative approach is to scale the coefficients based on \( \alpha \). For example, scaling \( C_d \) by

\[
C_d' = (0.97075 |\sin(\alpha)| + 0.02925) C_d
\]

generates a drag force similar to the level model over the entire range of measured \( \alpha \). Further investigation is required to determine which approach is best. Similarly, \( C_L \) should depend on \( \beta_h \). As \( \beta_h \) increases, the lift should decrease. Currently we are scaling the lift coefficient by

\[
C_L' = (0.9 |\cos(\beta_h)| + 0.1) C_L,
\]

which matches the level flight forces.

### 3.3 Conclusions

This chapter derived the V-Bat’s kinematic and dynamic model. Static tests, wind tunnel tests, and analysis were then used to estimate the parameters. Overall the model is reasonably accurate but there are several areas that could be improved.

First, the propeller and duct have different airflow assumptions where the propeller assumes that only the airspeed aligned with \( i^p \) reaches the propeller while the duct assumes that it captures and straightens all of the airflow. The propeller assumption is needed by
the propeller model while the duct assumption is used because we do not know how much the duct straightens the airflow. In most cases the difference between these two assumptions should be minor but it might become significant during a transition.

Second, the model overestimates the momentum drag because of the duct airflow assumption and by assuming the duct’s center of pressure is located at the mouth of the duct. Overestimating the momentum drag is better than underestimating it because the momentum drag fights the control vanes during a constant altitude LTH transition. As the vehicle pitches up, the momentum drag generates a large moment trying to pitch the vehicle down. If the transition approach can overcome the simulated momentum drag then it should easily overcome the momentum drag experienced during flight tests. The effect of overestimating the momentum drag during an HTL transition is mixed. On one hand, the momentum drag is trying to cause the V-Bat to level off when the V-Bat starts moving horizontally assuming it is maintaining a constant altitude. On the other hand, the control vanes might need to fight the momentum drag if the momentum drag is trying to cause the V-Bat to transition too fast.

Third, in an ideal simulator there would not be separate airframe models for hover and level flight. These separate models are needed due to the different wind tunnel tests that were performed. In addition, the restricted range of angle of attacks and side slip angles and the discrepancies between the level and hover coefficients demonstrate that this model could be improved. Either more extensive wind tunnel tests need to be performed or CFD approaches need to be used to fill in the missing data and help correct the discrepancies. In any case, the models agree when the wings are level and the vehicle is oriented into the wind which means smoothing is not needed when switching between the models during HTL or LTH transitions.

Finally, the propeller’s angular velocity model should be modified to include the battery voltage as well as the incoming airspeed. Hopefully, these modifications, especially the latter, would improve the propeller model’s performance at high airspeeds and low throttle commands. If not, further study should be performed to identify and correct the model discrepancy.
Chapter 4. Attitude Control

The attitude controller is arguably the most critical component of the tailsitter control system. Without a reliable attitude control system, none of the higher level controllers will work and the aircraft as a whole will not be functional. There are two different components of the attitude control system: representing and estimating the attitude error and the corresponding control method.

One of the most common quaternion attitude error methods is the quaternion attitude (QA) error [89] or a modified quaternion attitude (MQA) error [98]. However, as will be discussed, the QA error does not generate ideal trajectories for a tailsitter in hover. One possible way of handling this is to convert the QA error into something else. For example, the resolved tilt twist (RTT) method decomposes the QA error into the tilt and twist (heading) error in a manner that generates better trajectories [75]. However, the RTT error breaks down when the tilt error is large. Another approach is to convert the quaternion error into an Euler angle sequence which we call the resolved Euler angle (REA) error [38, 74, 82]. While the REA error generates similar trajectories to the RTT method, as will be shown, it has the same singularities that plague all Euler angle representations. Frank et al. avoid the singularities by switching to a different Euler angle sequence during the transition [38]. However, there is no guarantee that the attitude error will not reach a singularity. Marchini acknowledges the singularity issue and claims that “this problem can be easily avoided by using a logic gate to command a lesser pitch angle change” if the pitch error is greater than 90° [74, p. 22]. His ad-hoc approach is best described with a simple example. Let the current pitch be −10° and the desired pitch be 90°. In this case, converting the quaternion error to the ZYX Euler angles will have a problem because the pitch error is above 90°. Instead of initially using the full desired pitch, he claims that the pitch command should be 30°, which prevents the singularity, until \( \theta > 0° \) at which point the pitch command should switch to
Marchini’s approach requires that the quaternion attitude is converted to Euler angles before the error is computed in order to see if $\theta < 0$ and the pitch command needs to be modified. This approach can work but it is very specific to a transition and there is no guarantee that a singularity will not be reached during other times. Myrand-Lapierre et al. also convert the quaternion error to an Euler angle sequence but they do not consider the possibility of singularities [82]. Finally, the trajectory benefits of converting the quaternion error into an Euler angle sequence has not been considered in the literature.

There are several popular attitude control approaches in the tailsitter literature. First, the attitude error is used to generate a proportional desired angular velocity which the actuators track with a PID or other similar control scheme [13, 19, 54, 82, 90]. The desired angular velocity can also be created by a PI loop based on the attitude error [18]. Another popular method is using a PID, or PD, loop on the attitude error as in [89, 91, 116]. Often the attitude commands are the input to a reference model, either second order or based on the aircraft kinematics, which is used to generate the desired attitude for the controller [58]. Nonlinear controllers are also possible such as sliding mode control [42] which can provide stronger stability guarantees.

One downside with the previous control methods is that gain scheduling is needed because the moment generated for a given control surface deflection depends on the airflow over the vanes which can drastically change. Adaptive control methods can be used to account for this while providing stability guarantees. Knoebel and McLain use an adaptive backstepping control method, based on the quaternion error, with an RLS algorithm to estimate the thrust and moment parameters online [59]. Michini developed a MIMO L1 adaptive control scheme where an L1 adaptive controller augments a baseline control scheme, similar to the non-adaptive PD controller in [58], by attempting to match the angular velocity performance generated by a reference model. Jung and Shim developed a similar L1 adaptive control scheme except their L1 controller outputs actuator deflections while Michini’s method outputs a modified actuator velocity [54]. However, these controllers have only used the quaternion error directly or with a small variation. No controller with stability guarantees has been developed for the RTT or the REA errors.
This chapter makes several contributions to the literature. First, the REA error is developed and a saturation method is derived that can guarantee that a singularity will not be reached. We also show that the REA error generates similar trajectories to the RTT error which are better for a tailsitter in hover flight than the trajectories generated by the QA error. Then a backstepping controller similar to the one in [59] is derived for the REA error. However, this controller is only valid for the unsaturated REA error. Hybrid control is then used to merge the REA backstepping controller with Knoebel’s QA backstepping controller in a manner that is globally asymptotically stable.

This chapter is organized in the following manner. Section 4.1 develops the QA, RTT, and REA attitude error methods and discusses the strengths and weaknesses of each approach. The control methods, including PID, backstepping, and hybrid, are developed in Section 4.2. Section 4.3 includes simulation results comparing the different attitude error and control methods. Conclusions are given in Section 4.4.

4.1 Attitude Error

How the attitude error is computed can have a significant impact on the attitude response. In this section, three different quaternion based attitude error methods are discussed. A comparison of the three methods and when they should be used is included at the end of this section.

4.1.1 Quaternion Attitude Error

By far the most common quaternion based attitude error method is, what we call, the quaternion attitude (QA) error. Given the current attitude, $\eta$, and the desired attitude, $\eta^d$, the quaternion that describes the rotation from $B$ to $D$ is

$$\tilde{\eta} = \eta^d \otimes \eta^{-1}. \quad (4.1)$$

The imaginary component of (4.1),

$$e_{QA} \triangleq \tilde{\nu} = [\tilde{\eta}_1 \ \tilde{\eta}_2 \ \tilde{\eta}_3]^\top, \quad (4.2)$$
is the QA error and corresponds to the attitude error about the body axes. This approach is computationally simple and easily leads to control schemes with guaranteed stability and tracking performance.

One downside of the QA error is that it is not linear with the angle of rotation between $\boldsymbol{\eta}^d$ and $\boldsymbol{\eta}$. This nonlinearity is seen in Figure 4.1a which shows the QA error for $\boldsymbol{\eta} = [1 \ 0 \ 0 \ 0]^\top$ and $\boldsymbol{\eta}^d = [\cos(\gamma/2) \ \sin(\gamma/2) |0.22 \ 0.44 \ 0.87|]^\top$ as $\gamma$ ranges from $-180^\circ$ to $180^\circ$. The dotted red line is a straight line.

![Figure 4.1: Comparison of the QA and MQA errors about $\hat{i}^b$ for $\boldsymbol{\eta} = [1 \ 0 \ 0 \ 0]^\top$ and $\boldsymbol{\eta}^d = [\cos(\gamma/2) \ \sin(\gamma/2) |0.22 \ 0.44 \ 0.87|]^\top$ as $\gamma$ ranges from $-180^\circ$ to $180^\circ$. The dotted red line is a straight line.](image)

The nonlinearity can be removed by using, what we will call, the modified quaternion attitude (MQA) error which defines the attitude error as

$$
\mathbf{e}_{\text{MQA}} \triangleq 2 \cos^{-1}(\tilde{\eta}_0) \frac{\tilde{\eta}}{||\tilde{\eta}||}.
$$

Figure 4.1b shows the MQA error about $\hat{i}^b$ for the same $\boldsymbol{\eta}$ and $\boldsymbol{\eta}^d$ as in Figure 4.1a. Other similar modifications to the QA error are occasionally used such as in [79, 98].

While the QA error generates the minimal rotation from $\boldsymbol{\eta}$ to $\boldsymbol{\eta}^d$ it does not always produce an ideal rotation for a tailsitter. For example, consider the case when the V-Bat is vertical and is commanded to pitch down, a rotation about $\hat{j}^b$, by $20^\circ$ and change its heading, a rotation about $\hat{i}^b$, by $180^\circ$. In this case $\boldsymbol{\eta} = \begin{bmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \end{bmatrix}^\top$ and $\boldsymbol{\eta}^d = \begin{bmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \end{bmatrix}^\top$.

42
Figure 4.2: Top down view of the V-Bat pitching down 20° and changing its heading by 180°

Q2E_{ZYX}(0, 70°, 180°)]. Ideally, the attitude error would only exist in the $i^b$-$j^b$ plane which would cause the aircraft to rotate in a manner similar to what is shown in Figure 4.2a. However, the QA error is $e_{QA} = [-0.98 0 0.17]^\top$ which is in the $i^b$-$k^b$ plane. If the attitude controller used the QA error, the V-Bat would initially rotate towards the $j^b$ axis, as shown in Figure 4.2b, which would cause some undesired translational motion. This point is further illustrated in Figure 4.3 which shows $e_{QA}$ for $\eta = \left[\frac{1}{\sqrt{2}} 0 \frac{1}{\sqrt{2}} 0\right]^\top$ and $\eta^d = Q2E_{ZYX}(0, 70°, \psi)$ where $\psi \in [-180°, 180°]$. As can be seen, especially for $\psi = \pm 90°$ and $\psi = \pm 180°$, using $e_{QA}$ will cause the aircraft to have some undesired motion if the aircraft is commanded to pitch and change its heading at the same time.

4.1.2 Resolved Tilt Twist

This behavior was originally described by Matsumoto et al. who proposed a solution called the resolved tilt-twist (RTT) angle control [75]. Unlike the QA error which treats the attitude error as a single rotation, the RTT error treats it as a tilt error and a twist, or heading, error. As will be shown, this decoupling between tilt and twist generates attitude errors that fit a tailsitter in hover flight. This section builds upon their original work by making several improvements and corrections to their RTT derivation and is based on the derivation in [14] and [15].
The attitude error can be represented by the rotation matrix from \( \mathcal{B} \) to \( \mathcal{D} \). Given the rotation matrix from \( \mathcal{V} \) to \( \mathcal{B} \), \( \mathbf{R}_v^b \), and the rotation matrix from \( \mathcal{V} \) to \( \mathcal{D} \), \( \mathbf{R}_v^d \), the error rotation matrix is

\[
\mathbf{R}_b^d = \mathbf{R}_v^d \left( \mathbf{R}_v^b \right)^\top \equiv 
\begin{bmatrix}
\tilde{r}_{xx} & \tilde{r}_{xy} & \tilde{r}_{xz} \\
\tilde{r}_{yx} & \tilde{r}_{yy} & \tilde{r}_{yz} \\
\tilde{r}_{zx} & \tilde{r}_{zy} & \tilde{r}_{zz}
\end{bmatrix}.
\]

While there are many ways to interpret the rows and columns of \( \mathbf{R}_b^d \), we will use the fact that \( \tilde{r}_{a,b} \) is the desired \( a \)-axis projected onto the body \( b \)-axis. For example, \( \tilde{r}_{y,x} \) is \( \mathbf{j}^d \) projected onto \( \mathbf{i}^b \).

**Tilt Error**

The tilt error is the angle between \( \mathbf{i}^b \) and \( \mathbf{i}^d \), as shown in Figure 4.4a, and is computed by

\[
\theta_{\text{tilt}} = \cos^{-1} \left( \mathbf{i}^d \cdot \mathbf{i}^b \right). \tag{4.4}
\]

While this tilt error definition is useful when computing the twist error, the attitude controller needs the error about \( \mathbf{j}^b \) and \( \mathbf{k}^b \). These errors are found by projecting \( \mathbf{i}^d \) onto the \( \mathbf{i}^b - \mathbf{k}^b \) and
The RTT tilt error is the angle between $i^d$ and $j^b$ planes respectively. The error about $j^b$, shown in Figure 4.4b, is

$$\tilde{\theta}_y = -\text{atan2}(\tilde{r}_{13}, \tilde{r}_{11}),$$

where the negative sign accounts for the direction of rotation and the error about $k^b$, shown in Figure 4.4c, is

$$\tilde{\theta}_z = \text{atan2}(\tilde{r}_{12}, \tilde{r}_{11}).$$

**Twist Error**

The strategy for deriving the twist error is to find an intermediate coordinate frame with an identical twist error but whose $x$-axis is aligned with $i^d$. The twist error is then the angle of rotation about $i^d$ required to align the remaining axes. The angle of rotation needed to align $i^b$ and $i^d$ is $\theta_{\text{tilt}}$ computed by (4.4) and the unit length axis of rotation is given by

$$\nu_{\text{twist}} = \frac{i^b \times i^d}{\|i^b \times i^d\|}. \quad (4.5)$$
Because the vectors are expressed in the vehicle frame, $\mathbf{\nu}_{\text{twist}}$ is also expressed in the vehicle frame. For the rotation to occur correctly, this vector must be expressed in the body frame which is given by

$$\mathbf{\nu}^b_{\text{twist}} = \mathbf{R}_b^v \mathbf{\nu}_{\text{twist}}.$$  

The tilt error is removed from the total error by using Rodrigues’ rotation formula which produces a rotation matrix that rotates a vector by a given angle about an arbitrary axis. Using $\mathbf{\nu}^b_{\text{twist}}$ and $\theta_{\text{tilt}}$ as the axis and angle of rotation in Rodrigues’ formula provides

$$\mathbf{R}'_{\nu_{\text{twist}}} = \begin{cases} I + [\mathbf{\nu}^b_{\text{twist}}] \sin(\theta_{\text{tilt}}) + [\mathbf{\nu}^b_{\text{twist}}]^2 (1 - \cos(\theta_{\text{tilt}})), & \theta_{\text{tilt}} \neq 0, \\ I, & \theta_{\text{tilt}} = 0, \end{cases} \quad (4.6)$$

where

$$[\mathbf{\nu}^b_{\text{twist}}] = \begin{bmatrix} 0 & -\nu^b_{\text{twist},z} & \nu^b_{\text{twist},y} \\ \nu^b_{\text{twist},z} & 0 & -\nu^b_{\text{twist},x} \\ -\nu^b_{\text{twist},y} & \nu^b_{\text{twist},x} & 0 \end{bmatrix}.$$  

It is important to note that the rotation matrix obtained from (4.6) rotates a vector within a coordinate frame. The rotation matrix that rotates the coordinate frame is obtained by transposing $\mathbf{R}'_{\nu_{\text{twist}}}$ or by changing the angle of rotation to $-\theta_{\text{tilt}}$. With this, (4.6) becomes

$$\mathbf{R}_{\nu_{\text{twist}}} = \begin{cases} I - [\mathbf{\nu}^b_{\text{twist}}] \sin(\theta_{\text{tilt}}) + [\mathbf{\nu}^b_{\text{twist}}]^2 (1 - \cos(\theta_{\text{tilt}})), & \theta_{\text{tilt}} \neq 0, \\ I, & \theta_{\text{tilt}} = 0. \end{cases} \quad (4.7)$$

The condition on $\theta_{\text{tilt}}$ in (4.6) and (4.7) is required because (4.5) becomes ill-defined as $\mathbf{i}^b \times \mathbf{i}^d$ approaches zero which occurs when $\theta_{\text{tilt}}$ approaches zero. This condition is more appropriate than the one in [75], which is $\mathbf{R}_d^b = \mathbf{I}_3$, since the presence of twist error does not affect whether or not the $x$-axes are aligned. $\mathbf{R}_b^d = \mathbf{I}_3$ only when there is zero tilt and zero twist error.

The rotation matrix describing the intermediate coordinate frame with the $x$-axis aligned with $\mathbf{i}^b$ is given by

$$\mathbf{R}_a^b = \mathbf{R}_{\nu_{\text{twist}}} \mathbf{R}_v^b,$$
and the intermediate coordinate frame’s unit vectors are denoted by

\[ [i^a, j^a, k^a]^\top \triangleq R^a. \]

The angle of rotation needed to align \( k^d \) and \( j^a \) is

\[ \theta_{\text{twist}} \triangleq \cos^{-1} (k^a \cdot k^d). \]

However, this only gives the magnitude and not the sign of the error. The direction of the rotation can be determined by

\[ \theta_{\text{sign}} \triangleq \cos^{-1} (j^a \cdot k^d) \]

so the twist error is

\[ \tilde{\theta}_x = \begin{cases} -\theta_{\text{twist}}, & \theta_{\text{sign}} \leq \frac{\pi}{2}, \\ \theta_{\text{twist}}, & \theta_{\text{sign}} > \frac{\pi}{2}. \end{cases} \]

Figures 4.5a and 4.5b show this for a positive and negative twist error respectively.
Figure 4.6: Top down view of the V-Bat pitching down $20^\circ$ and changing its heading by $180^\circ$ when using the RTT error.

**RTT Error**

The RTT error is

\[
e_{RTT} \triangleq \begin{bmatrix} \tilde{\theta}_x \\ \tilde{\theta}_y \\ \tilde{\theta}_z \end{bmatrix}.
\]

(4.8)

Algorithm 1 shows the complete RTT error method.

As mentioned in the previous section, the main issue with the QA error is that in certain circumstances it generates non-ideal attitude errors which cause undesired translational motion. However, the RTT error causes a rotation that matches the ideal rotation shown in Figure 4.2a. For example, consider the case looked at for the QA error where $\eta = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}^\top$ and $\eta^d = E2Q_{ZYZ}(0, 70^\circ, 180^\circ)$. The QA error is $e_{QA} = [-0.98, 0, 0.17]^\top$ which is in the $i^b-k^b$ plane while the RTT error is $e_{RTT} = [3.1416, 0.3491, 0]^\top$ which is in the $i^b-j^b$ plane as desired. Figure 4.6 shows the rotation that the RTT error would cause. Note that it matches the ideal rotation shown in Figure 4.2a. Figure 4.7 shows the RTT error for the exact same cases as Figure 4.3. As can be seen, the RTT error matches the desired attitude error behavior.

While the RTT error produces attitude errors that make physical sense for a tailsitter, its complexity hinders detailed analysis. In addition, the RTT error should not be used when the attitude error is large because the RTT error can experience large jumps. This issue is shown in Figure 4.8 where $\eta = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^\top$, $\eta^d = \begin{bmatrix} \cos(\frac{\gamma}{2}) & \sin(\frac{\gamma}{2}) \nu^\top \end{bmatrix}^\top$, and $\gamma \in [-\pi, \pi]$. Figure 4.8a shows the RTT error for $\nu = [0, 1, 0]^\top$. Note that it is well behaved for $|\gamma| \leq \frac{\pi}{2}$ but $e_z$ jumps from 0 to $\pi$ when $|\gamma| > \frac{\pi}{2}$. This jump will cause a full rudder command in addition to the desired elevator command potentially causing the tailsitter to crash. Likewise,
as shown in Figure 4.8b, $e_y$ jumps from 0 to $\pi$ when $\boldsymbol{\nu} = [0\ 0\ 1]^T$ and $|\gamma| > \frac{\pi}{2}$. Furthermore, the RTT error has other inconsistencies when $\theta_{\text{tilt}}$ is large. For example, let $\boldsymbol{\nu} = [0\ 0.96\ 0.28]^T$ which is shown in Figure 4.8c. In this case, the error is fine when $\theta_{\text{tilt}}$ is low but $|e_z|$ rapidly increases as $\theta_{\text{tilt}}$ increases. At $|\gamma| = \pi$ both $|e_y|$ and $|e_z|$ equal $\pi$ which is incorrect because $|e_z|$ should be much lower than $|e_y|$ which should be less than $\pi$. This inconsistency on the maximum error is also shown in Figure 4.8d, where $\boldsymbol{\nu} = \left[0\ \frac{1}{\sqrt{2}}\ \frac{1}{\sqrt{2}}\right]^T$, and Figure 4.9 which shows the RTT error for two cases when the error about $i^b$ is nonzero. In general, these jumps occur when $\theta_{\text{tilt}}$ is large which is why the problem does not appear in Figure 4.7.

### 4.1.3 Resolved Euler Angle

While the RTT error method computes attitude errors that make physical sense for a tailsitter, its complexity and the problems that occur when $\theta_{\text{tilt}}$ is large prevent it from being used without further development. Instead of attempting to correct the RTT issues, we developed an alternate attitude error method with the following objectives:

- Produce attitude errors similar to the RTT error when $\gamma$ is small,
- Produce reasonable values when $\theta_{\text{tilt}}$ is large,
- Computationally simpler than the RTT method,
- Known error dynamics.
Our error method, which we call the Resolved Euler Angle (REA) method, draws inspiration on how the QA error and the RTT error treats the attitude error as a single rotation or a sequence of two rotations, tilt followed by twist, respectively. The REA error interprets the attitude error as a sequence of three rotations.

**REA Error Definition**

The REA attitude error is the attitude error treated as a sequence of three rotations; in other words, $\mathcal{B}$ is rotated into $\mathcal{D}$ by a sequence of three rotations. The first rotation
rotates $\mathcal{B}$ into the error-1, $\mathcal{E}_1$, frame by rotating about the unit vector $\mu_1$ by the angle $\tilde{\theta}_1$. The second rotation rotates $\mathcal{E}_1$ into the error-2, $\mathcal{E}_2$, frame by rotating about the unit vector $\mu_2$ by the angle $\tilde{\theta}_2$. Finally, $\mathcal{E}_2$ is rotated into $\mathcal{D}$ with a rotation about the unit vector $\mu_3$ by the angle $\tilde{\theta}_3$. In other words, the desired frame is the body frame that has undergone

$$
\mathcal{B} \xrightarrow{R^e_1(\tilde{\theta}_1)} \mathcal{E}_1 \xrightarrow{R^e_2(\tilde{\theta}_2)} \mathcal{E}_2 \xrightarrow{R^d_2(\tilde{\theta}_3)} \mathcal{D}, \quad (4.9)
$$

where $R^f_i(\theta_k)$ is the rotation matrix from frame i to frame j by rotating about the vector $\mu_k$ by the angle $\theta_k$. Note that $\mu_i$ can only have a single nonzero component and $\mu_1 \neq \mu_2 \neq \mu_3$.

The second constraint is not true of Euler angle sequences in general and will be explained later.

The rotation sequence (4.9) describes an Euler angle sequence which makes computing $\bar{\theta} = [\tilde{\theta}_1 \; \tilde{\theta}_2 \; \tilde{\theta}_3]^\top$ straightforward. The rotation angles are given by

$$
\bar{\theta} = Q2E(\eta),
$$
where the conversion, \( Q2E(\cdot) \), can be derived by setting the rotation matrix formed by the Euler angle sequence equal to the rotation matrix formed by the quaternion attitude error,

\[
R_{e_2}(\tilde{\theta}_3)R_{e_1}(\tilde{\theta}_2)R_{b}^{e_1}(\tilde{\theta}_1) = \begin{bmatrix}
2\tilde{\eta}_0^2 + 2\tilde{\eta}_1^2 - 1 & 2\tilde{\eta}_1\tilde{\eta}_2 + 2\tilde{\eta}_0\tilde{\eta}_3 & 2\tilde{\eta}_1\tilde{\eta}_3 - 2\tilde{\eta}_0\tilde{\eta}_2 \\
2\tilde{\eta}_1\tilde{\eta}_2 - 2\tilde{\eta}_0\tilde{\eta}_3 & 2\tilde{\eta}_0^2 + 2\tilde{\eta}_2^2 - 1 & 2\tilde{\eta}_0\tilde{\eta}_1 + 2\tilde{\eta}_2\tilde{\eta}_3 \\
2\tilde{\eta}_1\tilde{\eta}_3 + 2\tilde{\eta}_0\tilde{\eta}_2 & 2\tilde{\eta}_0\tilde{\eta}_1 - 2\tilde{\eta}_2\tilde{\eta}_3 & 2\tilde{\eta}_0^2 + 2\tilde{\eta}_3^2 - 1
\end{bmatrix},
\]

and solving for \( \tilde{\theta}_1, \tilde{\theta}_2, \) and \( \tilde{\theta}_3 \). The Euler angle error is then rearranged to provide the REA error

\[
e_{\text{REA}} \triangleq R_{s}^{b} \begin{bmatrix} \tilde{\theta}_1 \\ \tilde{\theta}_2 \\ \tilde{\theta}_3 \end{bmatrix} = R_{s}^{b} \tilde{\theta},
\]

where \( R_{s}^{b} = [\mu_1 \mu_2 \mu_3] \). For example, the conversion from the QA error to the ZYX REA error is

\[
\begin{align*}
\tilde{\theta}_1 &= \tan^{-1}\left( \frac{2(\tilde{\eta}_0\tilde{\eta}_3 - \tilde{\eta}_1\tilde{\eta}_2)}{2\tilde{\eta}_0^2 + 2\tilde{\eta}_1^2 - 1} \right) \\
\tilde{\theta}_2 &= \sin^{-1}(2(\tilde{\eta}_0\tilde{\eta}_2 - \tilde{\eta}_1\tilde{\eta}_3)) \\
\tilde{\theta}_3 &= \tan^{-1}\left( \frac{2(\tilde{\eta}_0\tilde{\eta}_1 - \tilde{\eta}_2\tilde{\eta}_3)}{2\tilde{\eta}_0^2 + 2\tilde{\eta}_3^2 - 1} \right),
\end{align*}
\]

and

\[
R_{s}^{b} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.
\]

Note that the rotation vectors, \( \mu_i \), where \( i \in \{1, 2, 3\} \), need to form a standard basis of \( \mathbb{R}^3 \) otherwise the error about one or more axis will always be zero. For example, let

\[
R_{s}^{b} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix},
\]

52
Figure 4.10: REA attitude error using the ZYX Euler angle sequence for $\eta = E2Q_{ZYX}(0, 90^\circ, 0)$ and $\eta^d = E2Q_{ZYX}(0, 70^\circ, \psi^d)$ where $\psi^d$ ranges from $-180^\circ$ to $180^\circ$.

Figure 4.11: Top down view of the V-Bat pitching down $20^\circ$ and changing its heading by $180^\circ$ when using the REA error which corresponds to the ZYZ Euler angle sequence. In this case, the error about $i^b$ is always zero.

The REA error is similar to the RTT error when the angle of rotation is small. For example, Figure 4.10 shows the ZYX REA error for $\eta = E2Q_{ZYX}(0, 90^\circ, 0)$ and $\eta^d = E2Q_{ZYX}(0, 70^\circ, \psi^d)$, where $\psi^d \in [-180^\circ, 180^\circ]$. Note that this is the same case used for Figures 4.3 and 4.7 and that the RTT and REA errors are almost identical. Like the RTT error, using the REA error will cause the aircraft to follow the ideal rotation as shown in Figure 4.11.

Like the RTT error, the REA error can jump when the angle of rotation is large. Unlike the RTT error, it is easy to determine that the REA error always jumps when $\tilde{\theta}_2 = \pm \frac{\psi}{2}$. Two examples are shown in Figure 4.12.
Figure 4.12: Examples where the REA error breaks down. Each example uses \( \eta = [1 \ 0 \ 0 \ 0]^\top \) and \( \eta^d = [\cos(\frac{\gamma}{2}) \ \sin(\frac{\gamma}{2}) \nu] \top \) where \( \gamma \) ranges from \(-180^\circ\) to \(180^\circ\). Note that the error jumps when \( e_y = \pm \frac{\pi}{2} \).

One explanation for these jumps is that the Euler angle sequence hits a singularity when \( \tilde{\theta}_2 = \pm \frac{\pi}{2} \). These singularities in the REA error are a significant issue that, like the RTT issues, could potentially cause a crash. However, the singularities can be avoided if the control scheme can guarantee that \( \left| \tilde{\theta}_2 \right| < \frac{\pi}{2} \) or by saturating \( e_{\text{REA}} \). Several saturation methods will be derived in the remainder of this section. Note that the saturation methods assume \( \tilde{\eta}_0 \geq 0 \). If \( \tilde{\eta}_0 < 0 \) then these methods can still be used by first negating \( \tilde{\eta} \), running the saturation algorithm, and then negating the result.

**Angle of Rotation Saturation Method**

The most straightforward saturation approach is to limit the error quaternion’s angle of rotation. Using this approach the saturated error quaternion is

\[
\tilde{\eta} = \begin{cases} 
\tilde{\eta}, & \cos^{-1}(\tilde{\eta}_0) \leq \frac{\tilde{\gamma}}{2}, \\
[\cos(\frac{\gamma}{2}) \ \sin(\frac{\gamma}{2}) \nu] \top, & \text{otherwise},
\end{cases}
\]

where \( \tilde{\gamma} \) is the angle of rotation limit. After \( \tilde{\eta} \) is saturated, if needed, it is converted to Euler angles providing the saturated REA error \( e_{\text{REA}} \). The angle of rotation saturation algorithm is shown in Algorithm 2.
The main advantage of this saturation approach is its simplicity. The main disadvantage of this approach is that it is overly aggressive. The unsaturated REA error will not have problems when \( |\tilde{\theta}_2| < \frac{\pi}{2} \) yet this saturation approach can saturate the error even when \( \tilde{\theta}_2 = 0 \). Saturating when \( \tilde{\theta}_2 \) is small removes the trajectory benefits which is the primary reason to use the REA error.

**Selective Saturation Method**

Instead of saturating the error when \( \gamma > \tilde{\gamma} \), a better approach is to only saturate the error when \( |\tilde{\theta}_2| \) is above the limit \( \tilde{\theta} \). If \( |\tilde{\theta}_2| > \tilde{\theta} \), then the angle of rotation needs to be modified such that \( |\tilde{\theta}_2| = \tilde{\theta} \) while the axis of rotation remains unchanged. Unfortunately, there is no general form for this selective saturation method. The following derivation is specifically for the ZYX Euler angle sequence but the derivation for the other Euler sequences follow the same procedure.

If \( 0 < \tilde{\theta} < \theta_2 \) then we need to find \( \tilde{\gamma} \) such that

\[
\sin^{-1}(2(\tilde{\eta}_0 \tilde{\eta}_2 - \tilde{\eta}_1 \tilde{\eta}_3)) = \tilde{\theta}.
\]

Inserting (2.3) and (2.4) into (4.12) and rearranging provides

\[
\nu_2 \cos\left(\frac{\tilde{\gamma}}{2}\right) \sin\left(\frac{\tilde{\gamma}}{2}\right) - \nu_1 \nu_3 \sin^2\left(\frac{\tilde{\gamma}}{2}\right) = \frac{\sin(\tilde{\theta})}{2}.
\]

Using the trigonometric identities

\[
\sin^2(x) = \frac{1 - \cos(2x)}{2}
\]

and

\[
\sin(2x) = 2 \sin(x) \cos(x)
\]

(4.13) simplifies to

\[
\nu_2 \sin(\tilde{\gamma}) + \nu_1 \nu_3 \cos(\tilde{\gamma}) = \sin(\tilde{\theta}) + \nu_1 \nu_3.
\]
We will examine a simpler case to solve (4.14) for $\bar{\gamma}$.

We start with $c \cos(x - y)$, where $y = \arctan2(b, a)$, and expand it to

$$c \cos(x - y) = c \cos(x) \cos(y) + c \sin(x) \sin(y).$$  \hspace{1cm} (4.15)

By using the right triangle shown in Figure 4.13 and noting that

$$\cos(y) = \frac{a}{c},$$
$$\sin(y) = \frac{b}{c},$$

(4.15) can be written as

$$c \cos(x - y) = a \cos(x) + b \sin(x).$$  \hspace{1cm} (4.16)

Using (4.16) and defining $a \triangleq \nu_1 \nu_3$, $b \triangleq \nu_2$, and $x \triangleq \bar{\gamma}$, (4.14) becomes

$$-\sqrt{\nu_2^2 + \nu_1^2 \nu_3^2} \cos(\bar{\gamma} - \arctan2(\nu_2, \nu_1 \nu_3)) = \sin(\bar{\theta}) + \nu_1 \nu_3,$$  \hspace{1cm} (4.17)

where the negative first term provides the correct sign. At this point (4.17) can be easily solved to obtain the saturated angle of rotation

$$\bar{\gamma}^+ = -\cos^{-1}\left(\frac{\sin(\bar{\theta}) + \nu_1 \nu_3}{\sqrt{\nu_2^2 + \nu_1^2 \nu_3^2}}\right) + \arctan2(\nu_2, \nu_1 \nu_3),$$  \hspace{1cm} (4.18)
where the \( \cdot^+ \) indicates a positive \( \theta_2 \). This process can be repeated for the case when \( \tilde{\theta}_2 < -\bar{\theta} < 0 \) to obtain the desired angle of rotation

\[
\tilde{\gamma}^- = \cos^{-1}\left( \frac{-\sin(\bar{\theta}) + \nu_1 \nu_3}{\sqrt{\nu_2^2 + \nu_1^2 \nu_3^2}} \right) + \text{atan2}(\nu_2, \nu_1 \nu_3).
\]

The saturated error quaternion is given by

\[
\tilde{\eta} = \begin{cases} \\
\tilde{\eta}, & \left| \sin^{-1}(2(\tilde{\eta}_0 \tilde{\eta}_2 - \tilde{\eta}_1 \tilde{\eta}_3)) \right| \leq \bar{\theta}, \\
\cos\left(\frac{\bar{\gamma}^+}{2}\right) \nu \\
\sin\left(\frac{\bar{\gamma}^+}{2}\right) \nu \\
\cos\left(\frac{\bar{\gamma}^-}{2}\right) \nu \\
\sin\left(\frac{\bar{\gamma}^-}{2}\right) \nu \\
\end{cases}
\]

Algorithm 3 shows the complete selective saturation method and Algorithm 4 computes the angle of rotation limit and the saturated quaternion.

**Modified Selective Saturation Method**

The REA selective saturation method successfully keeps the REA error from reaching the singularity that occurs when \( \tilde{\theta}_2 = \pm \frac{\pi}{2} \) while being less restrictive than the angle of rotation saturation method. However, problems can still occur with the REA error with selective saturation when the attitude error gets very large. For example, consider the case shown in Figure 4.14 where the desired rotation is entirely about \( \hat{j}^b \) and the REA with selective saturation is used with \( \bar{\theta} = 70^\circ \). Notice that the selective saturation kicks in when \( \gamma = \pm 70^\circ \) as it should. However, at \( |\gamma| \geq 110^\circ \), \( |\tilde{\theta}_2| < 70^\circ \), the error is no longer saturated, and \( \tilde{\theta}_1 \) and \( \tilde{\theta}_3 \) experience a large jump. If this situation occurs during a flight a full elevator and rudder command would be given instead of the desired elevator command. At best, these commands would cause a drastic maneuver and at worst would cause the aircraft to
Figure 4.14: The REA error with selective saturation for a desired rotation about $j^b$. The saturation limit is 70 degrees.

The selective saturation method needs to be modified to take these situations into account.

The cases we have determined to be problematic are when $\tilde{\theta}_1$ and $\tilde{\theta}_3$ experience a simultaneous significant jump which always occurs when the unsaturated $\tilde{\theta}_2 = \pm \frac{\pi}{2}$ providing a criteria to test for. However, the angle of rotation, $\bar{\gamma}$, where this occurs depends on the axis of rotation. This angle can be computed for any given $\nu$ by using the same approach that was used to determine the selective saturation limit. From (4.18), the angle of rotation needed for a given $\tilde{\theta}_2$ is

$$\bar{\gamma} = -\cos^{-1}\left(\frac{\sin(\tilde{\theta}_2) + \nu_1\nu_3}{\sqrt{\nu_2^2 + \nu_1^2\nu_3}}\right) + \text{atan2}(\nu_2, \nu_1\nu_3). \tag{4.19}$$

Looking at the term inside the $\cos^{-1}$ in (4.19) and setting $\tilde{\theta}_2 = \frac{\pi}{2}$ provides

$$x_1 = \frac{1 + \nu_1\nu_3}{\sqrt{\nu_2^2 + \nu_1^2\nu_3}},$$

which can range from $1 \leq x_1 < \infty$. Likewise, setting $\tilde{\theta}_2 = -\frac{\pi}{2}$ provides

$$x_2 = \frac{-1 + \nu_1\nu_3}{\sqrt{\nu_2^2 + \nu_1^2\nu_3}},$$
which can range from \(-\infty < x_2 \leq -1\). Only \(\nu\)s where \(x_1 = 1\) or \(x_2 = -1\) can cause the saturation issue and so the angle of rotation must be checked. Due to numerical errors it is better to check all of the cases where \(x_1 < 1 + \epsilon\) or \(x_2 > -1 - \epsilon\) where \(\epsilon\) is a small positive constant. We have found that \(\epsilon = 0.0001\) works well.

If \(x_1 < 1 + \epsilon\) then, by using (4.19), the angle of rotation that causes \(\theta_2 = \frac{\pi}{2}\) is

\[
\bar{\gamma} = \text{atan2}(\nu_2, \nu_1 \nu_3).
\] (4.20)

If \(x_2 > -1 + \epsilon\) then there are two possible \(\gamma\)s that cause \(\tilde{\theta}_2 = \pm \frac{\pi}{2}\). If \(\text{atan2}(\nu_2, \nu_1 \nu_3) > 0\) then

\[
\bar{\gamma} = \pi + \text{atan2}(\nu_2, \nu_1 \nu_3),
\] (4.21)

otherwise

\[
\bar{\gamma} = -\pi + \text{atan2}(\nu_2, \nu_1 \nu_3).
\] (4.22)

While rare, it is possible that both \(x_1 < 1 + \epsilon\) and \(x_2 > -1 - \epsilon\). In this case the smaller limit should be used.

The modified selective saturation method is as follows. First, the selective saturation method is run on \(\tilde{\eta}\). If the error was not saturated then \(\bar{\gamma}\) is calculated using (4.20)-(4.22). If \(\gamma > |\bar{\gamma}|\) then the error quaternion is saturated by

\[
\tilde{\eta}' = \begin{bmatrix}
\cos (\frac{\gamma}{2}) \\
\sin (\frac{\gamma}{2}) \nu
\end{bmatrix}
\] (4.23)

and the modified \(\tilde{\theta}_2'\) is calculated.

While (4.23) prevents \(\tilde{\theta}_2' = \pm \frac{\pi}{2}\) from occurring, it does not enforce \(|\tilde{\theta}_2'| \leq \tilde{\theta}\). To correct this the selective saturation method is run on \(\tilde{\eta}'\) which provides the true \(\tilde{\eta}\). Algorithm 5 lists the complete modified selective saturation algorithm.
Saturation Methods Comparison

Figure 4.15 compares the unsaturated REA error and the three REA saturation methods in four different situations which were selected to illustrate their strengths and weaknesses. Figures 4.15b, 4.15c, and 4.15d show that, like the RTT, the unsaturated REA error can experience large jumps that could cause the V-Bat to become unstable. Unlike the RTT, which has issues whenever $\theta_{\text{tilt}}$ becomes large, the REA error only jumps when $\tilde{\theta}_2 = \pm 90^\circ$.

Figure 4.15a demonstrates the angle of rotation saturation method’s main weaknesses. Note that although $\tilde{\theta}_2 = 0$ for all $\gamma$, the angle of rotation saturation method saturates the REA error. This saturation is overly aggressive because the REA error will only have a problem when $|\tilde{\theta}_2|$ approaches $90^\circ$. This problem is also shown in Figure 4.15b for a more complex $\nu$.

Figure 4.15c demonstrates the main weakness of the selective saturation method. While this method properly saturates the error when $|\theta_2| \geq \hat{\theta}$, as $\gamma$ increases $\tilde{\theta}_2$ drops below the saturation limit and $\tilde{\theta}_1$ and $\tilde{\theta}_3$ experience a large jump. The selective saturation method only delayed the point where this jump occurred instead of preventing it altogether. This situation is also shown in Figure 4.15d.

The modified selective saturate method handles each case properly. This method never unnecessarily saturates the REA error like the angle saturation method and it never turns off the saturation at the wrong times like the selective saturation method.

The unsaturated REA error should only be used if the control scheme can guarantee that $|\tilde{\theta}_2| < \frac{\pi}{2}$. The safer approach is to always saturate the REA error. The angle saturation is not a good option because it eliminates the primary benefit of the REA error. The selective saturation works as long as $\tilde{\theta}_2$ remains low. This could be accomplished, for a tailsitter, by using the YZX Euler angle error instead of the ZYX Euler angle sequence. However, there are no guarantees and problems could still arise. The best approach is to always use the modified saturation method.
Figure 4.15: Comparison of the REA error saturation methods. In each case, \( \tilde{\eta} = [\cos(\gamma/2) \sin(\gamma/2) \nu]^{\top} \) where \( \gamma \) ranges from \(-180^\circ\) to \(180^\circ\) and \( \nu \) depends on the figure.

4.1.4 Attitude Error Comparison

In order to compare the run times, each method was implemented in c++. A random \( \eta \) and \( \eta^d \) was generated and the attitude error was computed using each method. This was repeated one million times and the accumulated run time of each method is shown in Table 4.1. The limit for the angle of rotation saturation was \( \bar{\gamma} = 80^\circ \) and the limit for the selective and modified saturation was \( \bar{\theta}_2 = 80^\circ \). We are not overly interested in the actual run time for each method because this test was performed on an Intel Core i7 920 processor.
overclocked to 3.47 GHz. The run time on any autopilot would be significantly slower due to the differences in the processors’ architecture and clock speed. Instead, the comparative run times are more informative and are also shown in Table 4.1.

There are a few interesting points that should be mentioned. First, the MQA error is roughly five times slower than the QA error. The benefit provided by the MQA error is normally marginal and not worth the computational cost. Second, the REA error with the modified saturation runs 50% faster than the RTT method and it works over the full range attitude errors.

Determining which attitude error method to use depends on several factors. If the computational requirements are critical then the QA error should be used. Also the QA error’s mathematical simplicity leads itself to more advanced controllers with guaranteed performance and behavior. The REA error with the modified saturation should be used in applications where the QA error’s additional translational motion when tilting and rotating is undesirable such as a tailsitter taking off or landing in a constrained environment. At this point, there is no reason to use the RTT error over the REA attitude error.

Table 4.1: Attitude error runtime comparison. The second column shows the total runtime and the third column shows the runtime divided by the QA runtime.

<table>
<thead>
<tr>
<th>Method</th>
<th>Runtime (s)</th>
<th>Runtime Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA</td>
<td>0.3</td>
<td>1</td>
</tr>
<tr>
<td>MQA</td>
<td>1.68</td>
<td>4.90</td>
</tr>
<tr>
<td>RTT</td>
<td>8.06</td>
<td>26.9</td>
</tr>
<tr>
<td>REA - No Saturation</td>
<td>3.03</td>
<td>10.1</td>
</tr>
<tr>
<td>REA - Angle of Rotation Saturation</td>
<td>3.61</td>
<td>12.0</td>
</tr>
<tr>
<td>REA - Selective Saturation</td>
<td>3.07</td>
<td>10.2</td>
</tr>
<tr>
<td>REA - Modified Selective Saturation</td>
<td>5.39</td>
<td>17.9</td>
</tr>
</tbody>
</table>

4.2 Attitude Control

Several different attitude control schemes are explored in this section. First, a PID controller which an use all three attitude error methods will be discussed. While simple, the PID controller provides acceptable performance. However, gain scheduling is needed due to
the wide variation of the airflow over the control vanes that occurs during a typical flight. Next, two different backstepping controllers will be derived. The first is based on the QA error and guarantees almost global asymptotic stability. The second is based on the REA error but can only guarantee asymptotic stability. Both backstepping controllers assume that the aircraft can generate any desired moment. The two backstepping controllers are merged using hybrid automata techniques in order to achieve the REA trajectory benefits and the QA almost global asymptotic stability. Finally, an adaptive moment generation algorithm, for use with the backstepping and hybrid controllers, will be discussed.

4.2.1 PID

The simplest control strategy is to use a PID controller for each axis, i.e.

\[
\delta_i = k_{p,i}e_i - k_{p,d}\omega_i + \int_{t_0}^{t} e_i \delta \tau, \tag{4.24}
\]

where \(i \in \{x, y, z\}\) specifies the body axis. These commands can be relabeled to the standard aileron, elevator, and rudder commands by

\[
\begin{align*}
\delta_a &= \delta_x \\
\delta_e &= \delta_y \\
\delta_r &= \delta_z.
\end{align*}
\]

The main advantages of the PID controller are its simplicity and ease of tuning. In addition, all three attitude error methods can be used.

The main disadvantages of using (4.24) are that it does not take into account the varying control authority of the control vanes and its lack of stability guarantees. As shown in Chapter 3, the V-Bat’s control authority varies significantly based on the airflow over the control vanes. This means that a set of gains that work in hover flight will be too large to perform well in level flight. This issue can be handled through gain scheduling.
4.2.2 Backstepping

Backstepping is a nonlinear control design technique that can provide guaranteed tracking capabilities. While backstepping is only applicable to some nonlinear systems it is well suited to control the QA and REA errors. In the remainder of this section we will derive QA and REA attitude error based backstepping controllers based on a simplified dynamics model. These controllers assume that $\dot{\omega}^d$ is continuously differentiable and that the aircraft can generate any desired moment. Note that the QA controller is similar to one derived in [59].

QA Error

Recall that the QA error is given by (4.2) and the attitude kinematic equations are

$$\dot{\eta} = \Omega(\omega)\eta$$  \hspace{1cm} (4.25)

$$\dot{\omega} = J^{-1}(-\omega \times J\omega + m).$$

While this notation is perfectly valid, it hinders the derivation of the QA error dynamics. Using $\vec{\eta} = [\eta_1 \eta_2 \eta_3]^T$ and defining the matrix

$$\vec{\eta}^\times \triangleq \begin{bmatrix} 0 & -\eta_3 & \eta_2 \\ \eta_3 & 0 & -\eta_1 \\ -\eta_2 & \eta_1 & 0 \end{bmatrix},$$

(4.25) can be written as

$$\dot{\eta}_0 = -\frac{1}{2} \vec{\eta}^\top \omega$$

$$\dot{\vec{\eta}} = \frac{1}{2} (\vec{\eta}^\times + \eta_0 I_3) \omega.$$  \hspace{1cm} (4.26)

Given the desired angular velocity, $\omega^d$, the QA error dynamics are

$$\dot{e}_{QA} = \frac{1}{2} \left( \vec{\eta}^\times + \bar{\eta}_0 I_3 \right) (\omega^d - R_i^d \omega).$$  \hspace{1cm} (4.27)
Note that $\omega^d$ can be computed by

$$\omega^d = 2 \frac{\partial \eta^d}{\partial t} \otimes (\eta^d)^{-1},$$

if the control scheme does not provide it.

Define

$$V_1 = \frac{1}{2} \tilde{\eta}^\top \tilde{\eta}. \quad (4.28)$$

Taking the derivative of (4.28) and inserting (4.27) provides

$$\dot{V}_1 = \tilde{\eta}^\top \left( \frac{1}{2} \tilde{\eta}^\times + \tilde{\eta}_0 I_3 \right) R_b^d (R_b^b \omega^d - \omega), \quad (4.29)$$

which is an obvious candidate for backstepping. Adding and subtracting $\frac{1}{2} \tilde{\eta}^\top \left( \tilde{\eta}^\times + \tilde{\eta}_0 I_3 \right) R_b^d \omega_b$ to (4.29) provides

$$\dot{V}_1 = \tilde{\eta}^\top \left( \frac{1}{2} \tilde{\eta}^\times + \tilde{\eta}_0 I_3 \right) R_b^d (R_b^b \omega^d - \omega + \omega_b - \omega_b), \quad (4.30)$$

If we define

$$\ddot{\omega} \triangleq \omega - \omega_b$$

and set

$$\omega_b = R_b^b \omega^d + 2 R_b^b \left( \tilde{\eta}^\times + \tilde{\eta}_0 I_3 \right)^{-1} K_\eta \tilde{\eta}, \quad (4.31)$$

where $K_\eta$ is a positive definite gain matrix, then (4.30) becomes

$$\dot{V}_1 = -\tilde{\eta}^\top K_\eta \tilde{\eta} - \tilde{\eta}^\top \left( \frac{1}{2} \tilde{\eta}^\times + \tilde{\eta}_0 I_3 \right) R_b^d \ddot{\omega}. \quad (4.32)$$

Note that $-\tilde{\eta}^\top K_\eta \tilde{\eta}$ is negative definite.

Let the second Lyapunov function be

$$V_2 = V_1 + \ddot{\omega}^\top J \ddot{\omega}. \quad (4.33)$$
Taking the derivative of (4.33) and inserting (4.32) provides

\[ \dot{V}_2 = -\tilde{\eta}^T K_\eta \tilde{\eta} - \frac{1}{2} \tilde{\eta}^T \left( \tilde{\eta} \times \tilde{\eta}_0 I_3 \right) R_b^d \dot{\omega} + \dot{\omega}^T J \dot{\omega}, \] (4.34)

where

\[ J \dot{\omega} = -\omega \times J \omega + m - \dot{\omega}_b \] (4.35)

and \( \dot{\omega}_b \) is computed numerically. Inserting (4.35) into (4.34) provides

\[ \dot{V}_2 = -\tilde{\eta}^T K_\eta \tilde{\eta} - \frac{1}{2} \tilde{\eta}^T \left( \tilde{\eta} \times \tilde{\eta}_0 I_3 \right) R_b^d \dot{\omega} + \dot{\omega}^T \left( -\omega \times J \omega - J \dot{\omega}_b + m \right). \] (4.36)

By choosing the moment as

\[ m = J \dot{\omega}_b + \omega \times J \omega + \frac{1}{2} R_b^d \left( \left( \tilde{\eta} \times \right)^\top + \tilde{\eta}_0 I_3 \right) \tilde{\eta} - K_\omega \dot{\omega}, \] (4.37)

where \( K_\omega \) is a positive definite gain matrix, (4.36) becomes

\[ \dot{V}_2 = -\tilde{\eta}^T K_\eta \tilde{\eta} - \dot{\omega}^T K_\omega \dot{\omega}, \]

which is negative definite. Using Lyapunov’s second theorem, we know that \( V_2 \) asymptotically converges to the origin. This implies \( \tilde{\eta} \to 0 \) and \( \omega \to \omega_b \). As \( \tilde{\eta} \to 0 \), \( R(\tilde{\eta}) \to I_3 \) so \( \omega_b \to \omega^d \) which implies that \( \omega \to \omega^d \).

Unfortunately, the matrix \( \tilde{\eta} \times + \tilde{\eta}_0 I_3 \) is singular when \( \tilde{\eta}_0 = 0 \) and becomes ill-conditioned near this point. This means that there is a small region \( |\tilde{\eta}_0| < \epsilon_{QA} \), where \( \epsilon_{QA} \) is a small positive constant, where (4.27) cannot accurately describe the QA error dynamics and the QA backstepping controller is not guaranteed to work. Fortunately, \( \tilde{\eta}_0 = 0 \) corresponds to the worse possible attitude error, i.e. the aircraft is pointing the exact opposite of the desired direction, which should never occur in practice. However, this means that the QA backstepping controller is only almost globally asymptotically stable. In practice, if this situation ever occurs, \( \tilde{\eta}_0 \) should be artificially increased which states the attitude error
is not as large as it actually is. The attitude controller will rotate the aircraft in the correct direction and away from the error dynamics’ singularity.

REA

The REA error, which is given by (4.11), depends on the relative orientation of \( \mathcal{B} \) and \( \mathcal{D} \). Likewise, the REA error dynamics depend on the angular velocity of these coordinate frames. Let \( f_1 \) be the function that describes how the REA error changes due to the aircraft’s angular velocity and \( f_2 \) be the function that describes how the REA error changes due to the desired angular velocity. With these definitions the REA error dynamics are

\[
\dot{e}_{\text{REA}} = f_1(\omega, \dot{\theta}, \dot{\theta}, R_b^b) + f_2(\omega^d, \dot{\theta}, \dot{\theta}, R_b^b). \tag{4.38}
\]

The function \( f_1 \) can be determined by computing how the REA error changes when \( \omega^d = 0 \). Using the approach in [16], we can show that

\[
\omega = -R_s^b \begin{bmatrix} \dot{\theta}_1 \\ 0 \\ 0 \\ 0 \end{bmatrix} - R_{\xi_1}(\dot{\theta}_1) R_s^b \begin{bmatrix} 0 \\ \dot{\theta}_2 \\ 0 \end{bmatrix} - R_{\xi_1}(\dot{\theta}_1) R_{\xi_2}(\dot{\theta}_2) R_s^b \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_3 \end{bmatrix}. \tag{4.39}
\]

The right hand side of (4.39) can be simplified to obtain

\[
\omega = -\Psi_1(\dot{\theta}) \dot{\theta}, \tag{4.40}
\]

where the matrix \( \Psi_1 \) depends on the Euler angle sequence. Equation (4.40) can be solved for the REA error rate to obtain

\[
\dot{\theta} = f_1(\omega), \tag{4.41}
\]
where \( f_1 \triangleq -\Psi_1^{-1} \). A similar approach can be used to find \( f_2 \). First, we determine how the REA error changes when \( \omega^d = 0 \) which is

\[
\omega^d = R_s^b \begin{bmatrix}
0 \\
0 \\
\dot{\theta}_3
\end{bmatrix} + R_{E_2}^b(\tilde{\theta}_3) R_s^b \begin{bmatrix}
0 \\
\dot{\theta}_2 \\
0
\end{bmatrix} + R_{E_2}^b(\tilde{\theta}_2) R_{E_1}^b(\tilde{\theta}_2) R_s^b \begin{bmatrix}
\dot{\theta}_1 \\
0 \\
0
\end{bmatrix}. \quad (4.42)
\]

Simplifying (4.42) provides

\[
\omega^d = \Psi_2 (\tilde{\theta}) \dot{\theta}, \quad (4.43)
\]

where \( \Psi_2 \) also depends on the Euler angle sequence. Solving (4.43) for \( \dot{\theta} \) provides

\[
\dot{\theta} = f_2 \omega^d, \quad (4.44)
\]

where \( f_2 \triangleq \Psi_2^{-1} \). Inserting (4.41) and (4.44) into (4.38) leads to the REA error dynamics

\[
\dot{e}_{\text{REA}} = -\Psi_1^{-1}(\tilde{\theta}) \omega + \Psi_2^{-1}(\tilde{\theta}) \omega^d. \quad (4.45)
\]

However, (4.45) is valid only when \( \Psi_1 \) and \( \Psi_2 \) are invertible. For the ZYX Euler angles these matrices are

\[
\omega = \begin{bmatrix}
-s_{\tilde{\theta}_1} \dot{\tilde{\theta}}_2 + c_{\tilde{\theta}_1} c_{\tilde{\theta}_2} \dot{\tilde{\theta}}_3 \\
c_{\tilde{\theta}_1} \dot{\tilde{\theta}}_2 + s_{\tilde{\theta}_1} c_{\tilde{\theta}_2} \dot{\tilde{\theta}}_3 \\
\dot{\tilde{\theta}}_1 - s_{\tilde{\theta}_2} \dot{\tilde{\theta}}_3
\end{bmatrix} = \begin{bmatrix}
0 & -\tilde{s}_{\tilde{\theta}_1} & c_{\tilde{\theta}_2} \\
0 & c_{\tilde{\theta}_1} & \tilde{s}_{\tilde{\theta}_1} \\
1 & 0 & -\tilde{s}_{\tilde{\theta}_2}
\end{bmatrix} \begin{bmatrix}
\dot{\tilde{\theta}}_1 \\
\dot{\tilde{\theta}}_2 \\
\dot{\tilde{\theta}}_3
\end{bmatrix} = -\Psi_1 \dot{\theta}
\]

and

\[
\omega^d = \begin{bmatrix}
-s_{\tilde{\theta}_2} \dot{\tilde{\theta}}_1 + \dot{\theta}_3 \\
c_{\tilde{\theta}_2} s_{\tilde{\theta}_3} \dot{\tilde{\theta}}_1 + c_{\tilde{\theta}_3} \dot{\tilde{\theta}}_2 \\
c_{\tilde{\theta}_2} c_{\tilde{\theta}_3} \dot{\tilde{\theta}}_1 - s_{\tilde{\theta}_3} \dot{\tilde{\theta}}_2
\end{bmatrix} = \begin{bmatrix}
0 & 1 \\
c_{\tilde{\theta}_2} s_{\tilde{\theta}_3} & c_{\tilde{\theta}_3} \\
c_{\tilde{\theta}_2} c_{\tilde{\theta}_3} & -s_{\tilde{\theta}_3}
\end{bmatrix} \begin{bmatrix}
\dot{\tilde{\theta}}_1 \\
\dot{\tilde{\theta}}_2 \\
\dot{\tilde{\theta}}_3
\end{bmatrix} = \Psi_2 \dot{\theta}
\]

68
which are invertible with

\[
\Psi_1^{-1} = \begin{bmatrix}
c_{\bar{\theta}_1} t_{\bar{\theta}_2} & s_{\bar{\theta}_1} t_{\bar{\theta}_2} & 1 \\
- s_{\bar{\theta}_1} c_{\bar{\theta}_2} & c_{\bar{\theta}_1} c_{\bar{\theta}_2} & 0 \\
c_{\bar{\theta}_1} s_{\bar{\theta}_2} & s_{\bar{\theta}_1} c_{\bar{\theta}_2} & 0 \\
0 & s_{\bar{\theta}_3} & c_{\bar{\theta}_3} \\
0 & c_{\bar{\theta}_3} & -s_{\bar{\theta}_3} \\
1 & t_{\bar{\theta}_3} s_{\bar{\theta}_3} & t_{\bar{\theta}_2} c_{\bar{\theta}_3}
\end{bmatrix},
\]

\[
\Psi_2^{-1} = \begin{bmatrix}
0 & s_{\bar{\theta}_3} & c_{\bar{\theta}_3} \\
c_{\bar{\theta}_3} & s_{\bar{\theta}_2} & 0 \\
s_{\bar{\theta}_2} & c_{\bar{\theta}_2} & 0 \\
1 & t_{\bar{\theta}_3} s_{\bar{\theta}_3} & t_{\bar{\theta}_2} c_{\bar{\theta}_3}
\end{bmatrix},
\]

when \( \bar{\theta}_2 \neq \pm \frac{\pi}{2} \). Similarly, \( \Psi_1 \) and \( \Psi_2 \) will be singular when \( \bar{\theta}_2 = \pm \frac{\pi}{2} \) for any Euler angle sequence.

The dynamic equations needed for the REA backstepping controller derivation are

\[
\dot{\mathbf{e}}_{\text{REA}} = -\Psi_1^{-1}\left(\bar{\theta}\right)\mathbf{\omega} + \Psi_2^{-1}\left(\bar{\theta}\right)\mathbf{\omega}^d
\]

(4.46)

\[
J \dot{\mathbf{\omega}} = -\mathbf{\omega} \times J \mathbf{\omega} + \mathbf{m}.
\]

(4.47)

To simplify the control design, (4.46) is simplified to

\[
\dot{\mathbf{e}} = \Omega_1 \mathbf{\omega} + \Omega_2 \mathbf{\omega}^d,
\]

(4.48)

where \( \Omega_1\left(\bar{\theta}\right) \triangleq -\Psi_1^{-1}\left(\bar{\theta}\right) \) and \( \Omega_2\left(\bar{\theta}\right) \triangleq \Psi_2^{-1}\left(\bar{\theta}\right) \). Note that \( \Omega_1 \) and \( \Omega_2 \)'s dependence on \( \bar{\theta} \) and the subscript REA have been hidden for clarity.

Let the first Lyapunov function be

\[
V_1 = \frac{1}{2} \mathbf{e}^\top \mathbf{e}.
\]

(4.49)

Taking the derivative of (4.49) and inserting (4.48) provides

\[
\dot{V}_1 = \left(\Omega_1 \mathbf{\omega} + \Omega_2 \mathbf{\omega}^d\right)^\top \mathbf{e}.
\]

(4.50)
Adding and subtracting $\omega_b^\top \Omega_1^\top e$, where $\omega_b$ is the backstepping variable, to (4.50) provides

$$
\dot{V}_1 = (\omega - \omega_b)^\top \Omega_1^\top e + (\Omega_1 \omega_b + \Omega_2 \omega^d)^\top e.
$$

(4.51)

If we define $\dot{\omega} \triangleq \omega - \omega_b$ and set

$$
\omega_b = -\Omega_1^{-1} (K_e e + \Omega_2 \omega^d)
= \Psi_1 (K_e e + \Omega_2 \omega^d),
$$

(4.52)

where $K_e > 0$, then (4.51) becomes

$$
\dot{V}_1 = -e^\top K_e e + \dot{\omega}^\top \Omega_1^\top e.
$$

(4.53)

Note that $e^\top K_e e > 0 \ \forall \ e \neq 0$ and $e^\top K_e e = 0$ when $e = 0$.

Let the second Lyapunov function be

$$
V_2 = V_1 + \frac{1}{2} \dot{\omega}^\top J \dot{\omega}.
$$

(4.54)

Taking the derivative of (4.54) provides

$$
\dot{V}_2 = \dot{V}_1 + \dot{\omega}^\top J \dot{\omega}.
$$

(4.55)

Inserting (4.53),

$$
\dot{\omega} = \dot{\omega} - \dot{\omega}_b,
$$

(4.47), and

$$
\dot{\omega}_b = \dot{\Psi}_1 (K_e e + \Omega_2 \omega^d) + \Psi_1 \left( K_e (\Omega_1 \omega + \Omega_2 \omega^d) + \dot{\Omega}_2 \omega^d + \Omega_2 \dot{\omega}^d \right)
$$
into (4.55) provides

\[
\dot{V}_2 = \dot{\omega}^\top \left( \Omega_1 e + (-\omega \times J \omega) - J \dot{\psi}_1 \left( K_e e + \Omega_2 \omega^d \right) \right) 
+ \dot{\omega}^\top \left( -J \psi_1 \left( K_e \left( \Omega_1 \omega + \Omega_2 \omega^d \right) + \dot{\Omega}_2 \omega^d + \Omega_2 \dot{\omega}^d \right) + m \right) - e^\top K_e e. \tag{4.56}
\]

Let

\[
m = -K_\omega \dot{\omega} - \Omega_1 e - (-\omega \times J \omega) + J \psi_1 \left( K_e \left( \Omega_1 \omega + \Omega_2 \omega^d \right) + \dot{\Omega}_2 \omega^d + \Omega_2 \dot{\omega}^d \right) \tag{4.57}
- \dot{K}_\omega \omega - \Omega_1 e - (-\omega \times J \omega) + J \omega_b,
\]

where \( K_\omega > 0 \), then (4.56) becomes

\[
\dot{V}_2 = -e^\top K_e e - \dot{\omega}^\top K_\omega \dot{\omega}.
\]

Note that \( \dot{V}_2 < 0 \) except when \( e = 0 \) and \( \dot{\omega} = 0 \) at which point \( \dot{V}_2 = 0 \). This implies, from Lyapunov’s second theorem, that \( e \to 0 \) and \( \dot{\omega} \to 0 \) which implies \( \omega \to \omega_b \to -\Omega^{-1}_1 \Omega_2 \omega^d \).
As \( e \to 0 \), \( \Omega_1 \to I_3 \) and \( \Omega_2 \to I_3 \) so \( \omega \to \omega^d \). Therefore, the attitude angular velocity error will asymptotically approach zero as long as \( \theta_2 \neq \pm \frac{\pi}{2} \).

### 4.2.3 REA-QA Hybrid

If the trajectory benefits of the REA and the almost global asymptotic stability of the QA error backstepping controller are desired then a hybrid control scheme can be used. Unlike standard nonlinear systems that only model continuous variables, hybrid systems can model systems with continuous and discrete variables such as a quadcopter taking off [23]. In addition, hybrid systems can model a control scheme with discrete modes. Specifically, hybrid automaton, which is a subset of hybrid systems, have continuous states \( x \in \mathbb{R}^n \), continuous inputs \( u \in \mathbb{R}^m \), and a set of discrete operating modes \( Q \). The hybrid automaton also has
• A domain mapping $\mathcal{D}: \mathcal{Q} \rightarrow \mathbb{R}^n \times \mathbb{R}^m$. This defines, for each $q \in \mathcal{Q}$, the state space and input space where the states and inputs can be for each operation mode.

• A flow map $f: \mathcal{Q} \times \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$. This describes the state dynamics for each operation mode.

• A set of edges $\mathcal{E} \subset \mathcal{Q} \times \mathcal{Q}$. This describes which operative mode $q_i$ can switch to operative mode $q_j$.

• A guard mapping $\mathcal{G}: \mathcal{E} \rightarrow \mathbb{R}^n \times \mathbb{R}^m$. This describes, for each edge $(q_i, q_j) \in \mathcal{E}$, the conditions, $\mathcal{G}((q_i, q_j), \mathbf{x}, \mathbf{u})$, that $\mathbf{x}$ and $\mathbf{u}$ must satisfy for the transition between $q_i$ to $q_j$ to be possible.

• A reset map $\mathcal{R}: \mathcal{E} \times \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$. This describes how $\mathbf{x}$ changes when the operation mode switches. For example, the attitude representation can switch from one Euler angle sequence to another.

Basically, the hybrid system will operate normally within the mode $q_i$ and the states evolve by $\dot{\mathbf{x}} = f(q_i, \mathbf{x}, \mathbf{u})$. If the system enters the guard region $\mathcal{G}((q_i, q_j), \mathbf{x}, \mathbf{u})$ then the system can, but does not have to, switch to mode $q_j$. If the system switches modes then the states are modified by $\mathbf{x}^+ = \mathcal{R}((q_i, q_j), \mathbf{x}, \mathbf{u})$ and continue to evolve by $\dot{\mathbf{x}} = f(q_j, \mathbf{x}, \mathbf{u})$.

One possible method to extend the asymptotically stable region for the REA backstepping controller would be to have operation modes for the unsaturated error and the saturated error. However, the modified selective saturation method has three different saturation modes to account for. In addition, the complexity of these modes and the saturation method makes finding the error dynamics difficult if not impossible. These four operation modes have twelve different edges that would need to be analyzed separately. Overall, a purely REA based hybrid controller is impractical.

An alternative approach is to use the REA error when it is not saturated but otherwise use the QA error. In addition to simplifying the problem to two operation modes and two edges, it also makes physical sense. The REA error will only saturate when the attitude error about $\boldsymbol{\mu}_2$ is very large. Assuming that the control scheme is providing smooth, possible attitude commands then the REA will only saturate if something has gone wrong. In this
case, we do not care if the V-Bat’s trajectory is ideal. We only want to get the V-Bat’s attitude to a safe configuration.

The REA-QA hybrid controller has two operation modes labeled $Q = \{\text{REA, QA}\}$. The state vector, $x \in \mathbb{R}^7$ is

$$x = \begin{bmatrix} 0 \\ e \\ \omega \end{bmatrix},$$

when the REA mode is active and

$$x = \begin{bmatrix} \tilde{\eta} \\ \omega \end{bmatrix},$$

when the QA mode is active. The first element of $x_{\text{REA}}$ is always zero because the hybrid automation literature assumes that the dimension of the state vector does not change between modes. The inputs are $u = m \in \mathbb{R}^3$. The domain for the QA operation mode is

$$D(\text{QA}) = (0, 1] \times [-1, 1] \times [-1, 1] \times [-1, 1] \times \mathbb{R}^3 \times \mathbb{R}^3$$

or

$$D(\text{QA}) = [-1, 0) \times [-1, 1] \times [-1, 1] \times [-1, 1] \times \mathbb{R}^3 \times \mathbb{R}^3$$

depending on the initial sign of $\tilde{\eta}_0$. The domain mapping $D(\text{QA})$ also has the constraint that $\|x(1 : 4)\| = 1$. The domain for the REA operation mode depends on the Euler angle sequence used, but it is

$$D(\text{REA}) = 0 \times [-\pi, \pi] \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times [-\pi, \pi] \times \mathbb{R}^3 \times \mathbb{R}^3$$

for the ZYX Euler angle sequence. The state dynamics, or flow map, for the REA mode are

$$\dot{x} = f(\text{REA}, x, m) = \begin{bmatrix} 0 \\ -\Psi_1^{-1}(\hat{\theta})\omega + \Psi_2^{-1}(\hat{\theta})\omega^d \\ J^{-1}(-\omega \times J\omega + m) \end{bmatrix}$$
\[
\dot{x} = f(QA, x, m) = \begin{bmatrix}
-\frac{1}{2}\tilde{\eta}^\top (\omega^d - R(\tilde{\eta})\omega) \\
\frac{1}{2}(\tilde{\eta}^\times + \tilde{\eta}_0 I_3)(\omega^d - R(\tilde{\eta})\omega) \\
J^{-1}(-\omega \times J\omega + m)
\end{bmatrix}
\]

for the QA mode. Note that these are the error dynamics derived in the previous sections. Obviously there are two edges

\[\mathcal{E} = \{(REA, QA), (QA, REA)\}.
\]

When the mode switches from REA to QA, \(x\) needs to switch from tracking the REA error \(e\) to the quaternion error \(\tilde{\eta}\) and vice versa. Formally, the reset maps are

\[
\mathcal{R}((REA, QA), x, m) = \begin{bmatrix}
E2Q(x(2 : 4)) \\
x(5 : 7)
\end{bmatrix}
\]

\[
\mathcal{R}((QA, REA), x, m) = \begin{bmatrix}
0 \\
Q2E(x(1 : 4)) \\
x(5 : 7)
\end{bmatrix}.
\]

The controller’s objective, when the REA mode is active, is to drive \(x\) to

\[
x^d = \begin{bmatrix} 0 \\ \omega^d \end{bmatrix}.
\]

This can be accomplished using the backstepping controller given by (4.52) and (4.57). The controller’s objective, when the QA mode is active, is to drive \(x\) to

\[
x^d = \begin{bmatrix} \pm 1 \\ 0 \\ \omega^d \end{bmatrix}.
\]

This can be accomplished by using the backstepping controller given by (4.31) and (4.37). The only remaining component is to define the guard mappings.
Recall that the REA backstepping controller is only valid when the REA error is not saturated. This presents an obvious choice for the guard map between the REA and QA operating modes. Formally, the hybrid controller will switch to the QA mode when

\[
G(\text{REA}, \text{QA}), x, m = \left\{ (x, m) \mid \left( |\tilde{\theta}| \geq \tilde{\theta} \right) \text{ OR } (|\gamma| \geq |\bar{\gamma}|) \right\},
\]

where \(\tilde{\theta}\) is provided by the designer and \(\bar{\gamma}\) is computed by (4.20) - (4.22). The remaining guard map is not as straightforward to determine.

The REA backstepping controller does not claim that the REA error and the angular velocity error will individually asymptotically converge to zero. Instead, the controller guarantees that the Lyapunov function, repeated here for convenience,

\[
V_{\text{REA}}(x) = \frac{1}{2} e^\top e + \frac{1}{2} \hat{\omega}^\top J \hat{\omega},
\]

where

\[
\hat{\omega} = \omega - \Psi_1(K_e e + \Omega_2 \omega^d),
\]

asymptotically converges to zero. This means that over a short period of time the REA error \(e\) could increase while \(\hat{\omega}\) decreases enough to compensate. In addition, the boundary between the unsaturated and saturated REA error regions is non-convex. The REA backstepping controller could drive the system into the saturated REA error region forcing the hybrid controller to switch to the QA operation mode. Furthermore, \(V_{\text{QA}}(x_2) < V_{\text{QA}}(x_1)\), where \(V_{\text{QA}}\) is the QA Lyapunov function given by (4.33), does not guarantee that \(V_{\text{REA}}(x_2) < V_{\text{REA}}(x_1)\). If the guard map is

\[
G((\text{QA}, \text{REA}), x, m) = \left\{ (x, m) \mid \left( |\tilde{\theta}| < (1 - \epsilon_1)\tilde{\theta} \right) \text{ AND } (|\gamma| < (1 - \epsilon_2)|\gamma|) \right\},
\]

where \(\epsilon_1\) and \(\epsilon_2\) are small positive constants, the system could get stuck traveling between the two guard map regions, chattering between the two operating modes, and never driving the error to zero. Instead define a ball \(B_\epsilon \subset \mathbb{R}^7\) such that

\[
B_\epsilon = \{ x \mid V_{\text{REA}}(x) \leq (1 - \epsilon)V_{\text{REA},\text{min}} \},
\]

75
Figure 4.16: REA-QA hybrid controller guard map concept where the center dot is the desired state, the left side is the domain of $V_{QA}$ and the right side is the domain of $V_{REA}$. Note that the domains are infinite and are not bounded as portrayed. In addition, the shapes are only trying to convey the concept and not the actual shape of the domain. The dashed blue line on the right is the boundary between the unsaturated and saturated REA error. If the state reaches this boundary, the controller switches to the QA mode and the state is reset and is within the region enclosed by the solid blue line. Once the state enters the dashed red region, the controller switches back to the REA mode and the state is within the region enclosed by the solid red line.

where $\epsilon$ is a small positive constant and $V_{REA,\text{min}}$ is the smallest possible value that $V_{REA}$ can be and have saturated the REA error. Then if the REA backstepping controller is initialized within $x \in B_\epsilon$ the controller will not cause the REA error to saturate and the error will converge to zero. Therefore if the guard map is selected as

$$\mathcal{G}((QA, REA), x, m) = \{(x, m) \mid V_{REA} \leq (1 - \epsilon), V_{REA}\}$$

the hybrid system will asymptotically converge to the origin. The guard map concepts are shown in Figure 4.16

4.2.4 Adaptive Moment Generation

There are many different approaches to dynamically adapt parameters in order to generate a desired moment. Because this section is not a focus of our research, we used the recursive least squares approach developed in [59]. While this approach does not have
the theoretical guarantees that other approaches have, Knoebel showed that it performed better than a Lyapunov based adaption scheme with the QA backstepping controller. A brief description of the method is included. The vector of known basis functions is

$$\Phi_i = [1 \ 1 \hat{v}\text{vane} \delta_i^-]^T,$$

where $\hat{v}\text{vane}$ is the estimated airflow, induced velocity and airspeed, and $\delta_i^-$ is the previous command. The parameter estimates and covariance are updated by a stabilized recursive least squares algorithm given by

$$\hat{\Theta}_i = \hat{\Theta}_i + P_i \Phi_i \left( \dot{\omega}_i - \Phi_i^T \hat{\Theta}_i \right)$$

$$P_i^{-1} = \lambda P_i^{-1} + \Phi_i \Phi_i^T + \alpha (1 - \lambda),$$

where $\Theta_i$ are the unknown parameters, $P_i$ is the estimated covariance, $\alpha \in \mathbb{R}^{2 \times 2}$ is a stabilizing factor, and $\lambda$ is a forgetting factor. The actuator commands are generated, using these estimated parameters, by

$$\delta_i = \frac{1}{\hat{v}\text{vane} \hat{\Theta}_i,2} \left( \dot{\omega}_i^c - \hat{\Theta}_i,1 \right),$$

where the commanded angular acceleration is

$$\dot{\omega}_i^c = J^{-1}(-\omega \times J\omega + m^c).$$

Note that $i \in \{x, y, z\}$ specifies the axis and the command mapping is $\delta_x = \delta_a$, $\delta_y = \delta_e$ and $\delta_z = \delta_r$.

4.3 Results

The attitude error and controllers will be tested through several simulations. First, the attitude error methods will be compared by looking at the attitude trajectory they generate while using the PID attitude controller with no higher level controllers. Then, the REA and QA errors will be compared through a simulated hover flight. Finally, the
three different attitude error methods with PID and the REA-QA hybrid controller will be compared through a simulated flight covering all of the flight regimes.

The differences between the attitude error methods are illustrated through several different examples. In the first set of examples, designed to illustrate the QA error’s weakness in hover flight and the RTT and REA errors’ strength, the aircraft is initially vertical with the right wing pointed east. The aircraft is then commanded to tilt down and change its heading at the same time. Four different cases are presented, two where we want the aircraft to move north and two where we want the aircraft to move east or west. In each case, notice that the QA error does not rotate within the desired plane which causes significant velocity in undesired directions. PID controllers were used to control the attitude and no position control schemes were used to correct for the undesired motion.

In the first two cases, the aircraft is commanded to pitch down, rotation about $j^b$, by $45^\circ$ and change its heading, rotation about $i^b$ by $90^\circ$ and $180^\circ$ respectively. The rotation trajectories are shown in Figures 4.17 and 4.18. In these figures, the body frame axes are drawn onto the unit sphere as the rotation occurs. Note that the $z$-axis on the figures is negated so $i^b$ at the top of the unit sphere implies the aircraft is pointing up not down. In the second two cases, the aircraft is again commanded to change its heading by $90^\circ$ and $180^\circ$ degrees. However, instead of pitching down about $j^b$, the plane is commanded to tilt about $k^b$ by $45^\circ$. The trajectories for these two cases are shown in Figures 4.19 and 4.20. Note that the RTT and REA error decide to rotate in opposite directions in Figure 4.20. This is caused by small numerical differences in the computed error.

As can be seen in Figures 4.17-4.20, the QA error does not cause the aircraft to rotate in the desired plane while the RTT and REA error are much closer. This deviation from the desired plane causes significant velocity in undesired directions which is shown in Figure 4.21. Note that for each case the QA error causes an undesired velocity over twice as large as the RTT and REA errors.

Only the attitude controller was active in the previous examples. Including position control will reduce the difference between the REA and QA velocity responses. However, the REA and RTT errors still perform better than the QA error when the aircraft is commanded to tilt and change its heading at the same time. To show this, the aircraft was commanded
Figure 4.17: Trajectories that the different attitude error methods create when using the PID attitude controller. The aircraft is commanded to change its heading by 90° and pitch down 45°.

(a) Angled view
(b) Top down view

to climb five meters and then fly, in hover using the position controller described in the next chapter, twenty meters north with a heading of 0°. Once the aircraft was close to the waypoint, it was immediately commanded to fly back to the starting location while changing
Figure 4.18: Trajectories that the different attitude error methods create when using the PID attitude controller. The aircraft is commanded to change its heading by 180° and pitch down 45°.

its heading to 180°. Once it arrived at the starting location, it was commanded to change its heading back to 0°. The aircraft was not allowed to slow down or stop when it arrived at the first waypoint. Figure 4.22a shows the position of the aircraft when the QA error was
Figure 4.19: Trajectories that the different attitude error methods create when using the PID attitude controller. The aircraft is commanded to change its heading by 90° and rotate about \( k^b \) by 45°.

used with the PID attitude controller and Figure 4.22b shows the position of the aircraft when the REA error is used instead of the QA error. The initial takeoff is not shown. Notice that using the QA controller causes the vehicle to move west during the maneuver. This
Figure 4.20: Trajectories that the different attitude error methods create when using the PID attitude controller. The aircraft is commanded to change its heading by 180° and rotate about $k^b$ by 45°.

Westward motion is significantly reduced if the REA error is used. Figure 4.23 shows the aircraft’s position during the same maneuver when the REA-QA hybrid controller is used.
While the westward motion is larger than using the REA PID controller, it is much better than the motion caused by using the QA PID controller.

Figure 4.24 shows the attitude response for the PID attitude controller using either the QA, RTT, or REA error and the REA-QA hybrid controller. In this simulation, the aircraft was commanded to takeoff, transition to level flight, fly north-east, transition back to hover, and then land. The higher level controllers that were used will be discussed in later chapters, and the PID attitude controller switched the gains based on the current flight mode. Note that the QA, RTT, and REA PID attitude responses are almost identical. The
steady state error is due to the attitude controller integrator terms not being used. The REA-QA hybrid controller tracks the attitude commands better than the PID controller except during the transitions. In addition, the REA-QA hybrid controller is able to use the same gains for each flight mode because the current thrust parameters are estimated using the stabilized RLS method discussed earlier. The parameter estimates are shown in Figure 4.25.
4.4 Conclusions

First, we showed that the standard quaternion attitude error does not produce ideal trajectories for a tailsitter in hover flight and will cause undesired translational motion for some commands. We also showed that the resolved tilt twist error, which reduces the undesired translational velocity, breaks down when the attitude error is large. Converting the attitude error into Euler angles generates similar trajectories to the RTT error and a saturation algorithm is developed to improve the behavior with large attitude errors. A backstepping controller is derived to show that the REA error can be used with more advanced control approaches. The REA backstepping controller is combined with a QA
error backstepping controller through hybrid automata techniques to achieve almost global asymptotic stability. Simulation results showed that the REA-QA hybrid controller and PID controllers using the various error schemes all work for a complete tailsitter flight.
Chapter 5. Hover Control

There are many different control schemes that have been developed for tailsitters in hover. However, not all of these approaches are applicable to the V-Bat. For example, Hua et al. developed a control scheme based on Lyapunov arguments that can stabilize the attitude, velocity, and position of a tailsitter [44]. However, they assume that the net force on the aircraft, not including thrust, does not depend on the vehicle’s attitude. While an argument can be made for the validity of this assumption for shrouded-fan aircraft, such as the HoverEye, it is not valid for V-Bat type tailsitters with their large wings. We will only focus on the relevant control approaches.

Typically the hover dynamics are decomposed into several components each of which has its own controller. There are several ways to perform the separation which leads to different control approaches with different strengths and weaknesses. A brief overview of all of the approaches will be given as well as some of their position controllers. A discussion on the various altitude and heading control schemes will follow.

First, all of the hover dynamics can be considered at once such as with Johnson’s neural network based adaptive controller [50, 51], Anderson and Stone’s model predictive controller (MPC) [10], or Kriel’s LQR controller which generates a new linear model and LQR gains based on the current vehicle states every 200 ms [61]. In general, not separating the hover dynamics adds unnecessary complexity to the controller.

Second, the dynamics can be separated into longitudinal, lateral, and heading dynamics. For example, Escareño models the longitudinal dynamics as the altitude and the forward velocity, and the lateral dynamics as the altitude and the lateral velocity. With some simplification, these lateral and longitudinal dynamics can be modeled as a cascade of four integrators which can be controlled using nested saturation [34, 108]. Kriel also separates the dynamics into longitudinal and lateral components, which are controlled with the previously
mentioned LQR controller; however, his lateral dynamics consist of the lateral position and heading [61]. This decomposition approach is better suited to modeling the transition or level flight than hover flight. This is because the altitude needs to be included in both the longitudinal and lateral dynamics like Escareño’s approach but only one of those can be used to control the altitude. Alternatively, the altitude can be included in either the longitudinal or the lateral dynamics, like Kriel’s approach, but the decision is arbitrary because the altitude dynamics are just as related to the forward velocity as the lateral velocity.

Another approach is to separate the hover dynamics into horizontal, vertical, and heading components. For example, Knoebel’s controller uses a relative position based PID loop that generated a quaternion command [58]. Sobolic developed a similar inertial velocity PI controller [98]. This modeling approach allows the controller to limit the maximum commanded tilt.

The last approach is to separate the horizontal dynamics into relative forward and lateral components. Stone argues that this approach is only practical if the body frame velocities are considered and not the position. Otherwise heading would need to be fixed [105]. Overall, separating the horizontal dynamics into the relative forward and lateral dynamics allows the controller to take into account the different position dynamics, about each axis of the vehicle, that most tailsitters have.

This approach is the most common method and many controllers have been developed based on it. For example, Millet’s controller rotates the inertial position errors into the body frame which are used as the inputs for two PID loops that generate roll and pitch commands [80]. Stone et al. developed a gain scheduled LQR relative velocity controller [104,106]. Krogh also uses an LQR relative velocity controller but the states are augmented with integrators to reduce position tracking error [62]. Like Kriel, Wilson developed an LQR position controller that occasionally creates a new linear model, based on the vehicle’s current states, and computes the new LQR gains [115]. Frank et al. use an LQR relative position controller [38]. Peddle et al. created linear models of the body frame velocity response based on the attitude. Based on these models, the horizontal body velocity error is converted to a desired angle about each axis allowing the desired velocity to be tracked with steady state error due to disturbances [90]. A similar control system is derived in [115].
The relative velocity or position controllers, which are common with the last dynamics approach, work well but have issues when the heading changes. For example, consider a tailsitter in a stationary hover oriented north with strong wind coming from the north. The integrators in the forward control loop will have wound up to compensate for the wind. Then assume the vehicle rotates and is now oriented east. Because the controllers operate in the body frame, they assume that the wind is still coming towards the front of the vehicle. However, the wind is coming towards the side which means the controller’s wind compensation is in the wrong direction and it will take some time to correct. Inertial position or velocity controllers do not have this problem.

Various altitude control schemes have been developed including altitude error PD [80], altitude error PID [98], altitude error LQR [38], climb rate PI [90], and climb rate LQR [62]. Knoebel also uses an altitude error PD loop. However, his PD loop creates a thrust command which an adaptive controller tracks [58]. Typically a feedforward term is added to counteract gravity.

A variety of heading control schemes have been developed, including LQR [38], but these schemes are commonly a PID variant based on the heading [62] or heading rate [115].

One thing that is almost never considered is the vehicle’s response when the hover controller is initialized outside of stationary hover. If this issue is not considered then large initial commands can be generated which can cause very aggressive maneuvers and even instability. Because the V-Bat currently allows a human RC pilot to take, and give, control from the autopilot at any time, this issue must be considered. This issue can also arise when the vehicle leaves the transition mode after an LTH transition.

Our hover controller decouples the hover dynamics into the horizontal and vertical components and are based on the inertial position errors. This approach allows the wind to be better compensated for if the vehicle changes its heading. While the position controller cannot explicitly take into account the different forward or lateral dynamics, caused by the V-Bat’s large wings, the impact is lessened due to the lower-level attitude controller which does take the difference into account. Our hover controller consists of two main components shown in Figure 5.1. The reference models, which are described in Section 5.1, take the step commands generated by the flight mode controller and generate a desired position and
velocity for the controllers. The desired position and velocity are then tracked through simple PID controllers which are described in Section 5.2. Unlike all of the provided references, we explicitly look at the performance when the hover controller is initialized outside of stationary hover. Modifications to the baseline PID controllers that improve the stability and performance, when the controller is initialized outside the desired velocity constraints, are described in Section 5.3. The complete control system is analyzed in Section 5.4. Flight results are shown in Section 5.5, and some conclusions are given in Section 5.6.

5.1 Reference Models

The reference models convert the step-like position and altitude commands from the flight mode controller into smooth commands to be used by the position and altitude controllers. While there are many different ways of generating these reference models, such as LQR [69], we model the position and altitude dynamics as second-order systems. However, the desired velocity or climb rate from these second-order reference models can exceed the velocity limits so acceleration and velocity constraints are added. In addition, large commands can be generated if the hover controller is initialized while the aircraft is outside these velocity limits. If this occurs modified reference models are used until the vehicle is within the velocity constraints.

5.1.1 Position Reference Model

The position reference model takes the commanded inertial position from the flight mode controller and generates a smooth desired position and velocity for the position con-

Figure 5.1: The V-Bat’s hover control architecture
controller. It has two modes based on the initial ground speed: normal and decelerate. The decelerate mode is only used when the position controller is activated when the ground speed is above a user specified velocity limit. Once the ground speed drops below this limit, the normal reference mode is activated. The normal mode is also activated if the initial ground speed is below the velocity limit. Figure 5.2 shows the position reference model state machine.

Initialization

The reference model is initialized to the current state when the hover position control is activated. Specifically, \( \hat{p}^d_n(t_0) = \hat{p}_n(t_0) \), \( \hat{p}^d_e(t_0) = \hat{p}_e(t_0) \), \( \hat{p}_n(t_0) = p_n(t_0) \), and \( p_e(t_0) = \hat{p}_e(t_0) \), where \( t_0 \) is the time the controller is activated. If the ground speed is greater than the velocity limit, \( \sqrt{\left(\dot{\hat{p}}^d_n\right)^2 + \left(\dot{\hat{p}}^d_e\right)^2} > \bar{v}_p \), the decelerate reference model is used. Otherwise the normal reference model is used. Because the V-Bat does not track the desired position and velocity perfectly, the normal reference model is initialized to the current vehicle states instead of the decelerate reference model’s states once the V-Bat switches to the normal reference model.

Normal Reference Model

The normal position reference model is a second-order dynamic system with restrictions on the acceleration and velocity which describes the motion of the V-Bat in the inertial frame. While modeling the position, velocity, and acceleration of the V-Bat in the body frame would allow the model to take into account the different aerodynamics about the different body axes, it would require taking into account the heading and angular velocity of the vehicle significantly increasing the model’s complexity.
The desired north and east accelerations are given by

\[
\ddot{p}_n^d = -2\zeta_p \omega_{n,p} \dot{p}_n^d + \omega_{n,p}^2 (p_n^c - p_n^d) \\
\ddot{p}_e^d = -2\zeta_p \omega_{n,p} \dot{p}_e^d + \omega_{n,p}^2 (p_e^c - p_e^d),
\]

where \(\omega_{n,p}\) is the natural frequency and \(\zeta_p\) is the damping ratio. If the magnitude of the desired acceleration is above the acceleration limit, \(\sqrt{(\ddot{p}_n^d)^2 + (\ddot{p}_e^d)^2} > \bar{a}_p\), the acceleration is saturated by

\[
\ddot{p}_n^d = \bar{a}_p \cos(\theta_a) \\
\ddot{p}_e^d = \bar{a}_p \sin(\theta_a),
\]

where

\[
\theta_a = \tan^{-1}\left(\frac{\ddot{p}_e^d}{\ddot{p}_n^d}\right).
\]

The desired velocity is updated by

\[
\dot{p}_n^d(t) = \dot{p}_n^d(t) + T_s \frac{T}{2} (\dot{p}_n^d(t - T_s) + \dot{p}_n^d(t)) \\
\dot{p}_e^d(t) = \dot{p}_e^d(t - T_s) + T_s \frac{T}{2} (\dot{p}_e^d(t - T_s) + \dot{p}_e^d(t)).
\]

If the desired velocity is above the velocity limit, \(\sqrt{(\dot{p}_n^d)^2 + (\dot{p}_e^d)^2} > \bar{v}_p\), the velocity is saturated by

\[
\dot{p}_n^d = \bar{v}_p \cos(\theta_v) \\
\dot{p}_e^d = \bar{v}_p \sin(\theta_v),
\]

where

\[
\theta_v = \tan^{-1}\left(\frac{\dot{p}_e^d}{\dot{p}_n^d}\right).
\]
The desired position is updated by

\[ p^d_n(t) = p^d_n(t - T_s) + \frac{T_s}{2} (\dot{p}^d_n(t - T_s) + \ddot{p}^d_n(t)) \]  
(5.5)

\[ p^d_e(t) = p^d_e(t - T_s) + \frac{T_s}{2} (\dot{p}^d_e(t - T_s) + \ddot{p}^d_e(t)). \]  
(5.6)

**Decelerate Reference Model**

If the hover controller is initialized when the ground speed is above the specified limit then the reference model does not attempt to track the commanded position. Instead it attempts to slow the vehicle down in a safe and smooth manner. To do this the desired north and east accelerations are generated by

\[ \ddot{p}^d_n = -2\zeta_p \omega_{n,p} \dot{p}^d_n \]  
(5.7)

\[ \ddot{p}^d_e = -2\zeta_p \omega_{n,p} \dot{p}^d_e \]  
(5.8)

and are saturated by (5.1) and (5.2) if the magnitude is above the acceleration limit. The desired velocity is updated by (5.3) and (5.4) and the desired position is updated by (5.5) and (5.6).

There are two differences between the normal reference model and the decelerate reference model. First, the decelerate acceleration model (5.7) and (5.8) does not take into account the desired position. This prevents the desired velocity from increasing in certain cases. Second, the desired velocity is not saturated which would otherwise cause a large undesired jump in the desired velocity.

**5.1.2 Altitude Reference Model**

The altitude reference model takes the altitude command from the flight mode controller and generates smooth desired altitude and climb rates to send to the altitude controller. Like the position reference model, the altitude reference model takes into account the current climb rate when the controller is initialized. Unlike the position reference model, the altitude reference model has three modes: normal, climbing, and descending. The climbing
mode is used when the initial climb rate is above a user defined limit, i.e., \( \dot{h} > \bar{v}_h > 0 \). The descending mode is used when the initial climb rate is below a user defined limit \( \dot{h} < \bar{v}_h < 0 \). Once the climb rate is within the bounds then the normal reference mode is activated. The normal mode is also used when the initial climb rate is within the bounds. Figure 5.3 shows the altitude reference model state machine.

**Initialization**

The altitude reference model is initialized to the current state of the V-Bat when the autopilot enters an altitude control mode. Specifically \( \dot{p}_d^p(t_0) = \ddot{p}_d(t_0) \) and \( \dot{p}_d^d(t_0) = p_d(t_0) \), where \( t_0 \) is the time the controller is activated. Because the V-Bat does not track the desired altitude and climb rate perfectly, the normal reference model is initialized to the current vehicle states instead of the climbing or descending reference model’s states once the V-Bat switches to the normal reference model.

**Normal Reference Model**

The normal altitude reference model is a second-order dynamic system with restrictions on acceleration and velocity. The desired acceleration is generated by

\[
\ddot{h} = \begin{cases} 
\bar{a}_h \text{sign}\left(-2\zeta_h \omega_{n,h} \dot{h} + \omega_{n,h}^2 (h^c - h^d)\right), & \left|\dot{h}\right| > \bar{a}_h, \\
-2\zeta_h \omega_{n,h} \dot{h} + \omega_{n,h}^2 (h^c - h^d), & \text{otherwise}.
\end{cases}
\]
The desired climb rate is updated by

\[ \dot{h}^d(t) = \dot{h}^d(t - T_s) + \frac{T_s}{2} \left( \ddot{h}^d(t - T_s) + \dot{h}^d \right). \] (5.9)

If the desired climb rate is above the climb rate limit, \( \dot{h} > \upsilon_h \), it is set to

\[ \dot{h}^d = \upsilon_h, \]

and if the desired climb rate is below the descent rate limit, \( \dot{h} < \upsilon_h \), it is set to

\[ \dot{h}^d = \upsilon_h. \]

The desired altitude is updated by

\[ h^d(t) = h^d(t - T_s) + \frac{T_s}{2} \left( \dot{h}^d(t - T_s) + \dot{h}^d(t) \right). \] (5.10)

**Climbing Reference Model**

The primary objective, if the altitude controller is initialized when the V-Bat is climbing too fast, is to smoothly decrease the climb rate. A gentle deceleration is critical in this case because control authority can be lost if the throttle drops too low. To accomplish this the desired vertical acceleration is generated by

\[ \ddot{h}^d = \begin{cases} \bar{a}_h \text{sign} \left( -2\zeta_h \omega_{n,h} \dot{h} \right), & \left| \dot{h} \right| > \bar{a}_h, \\ -2\zeta_h \omega_{n,h} \dot{h}^d, & \text{otherwise}. \end{cases} \]

The climb rate is updated by (5.9) and the desired altitude is updated by (5.10). Note that the changes made are identical to the changes made to the position decelerate reference model.
Descending Reference Model

The worst possible case, in terms of stability, for the V-Bat is when it is descending rapidly in hover flight. In this situation, the incoming airflow competes with the induced velocity and decreases the overall control authority of the control vanes. To counter this, the V-Bat will not try to gently decelerate, like all of the other decelerate reference models used, when the hover controller is initialized when it is rapidly descending. Instead the desired climb rate is set to

\[ \dot{h}^d(t) = 0 \]

and the desired altitude is

\[ h^d(t) = h(t_0). \]

This will cause the position and velocity error in the controller to rapidly increase which will cause the throttle to rapidly increase until the V-Bat has stopped descending so quickly.

5.2 Control Scheme

PID controllers were selected because they are straightforward and easy to tune. Alternative control schemes can be used with the same reference models.

5.2.1 Position Controller

Let the north and east position error be denoted by \( \tilde{p}_n \) and \( \tilde{p}_e \) respectively. The integrated north and east error are given by

\[ n_i = \int_{t_0}^{t} \tilde{p}_n \delta \tau \]
\[ e_i = \int_{t_0}^{t} \tilde{p}_e \delta \tau. \]

Instead of saturating each integrator separately, we impose a saturation limit based on the \( l_2 \) norm. The integrator magnitude is

\[ i_m = \sqrt{n_i^2 + e_i^2}. \]
If \( i_m > \bar{i}_m \) then the integrator is saturated by

\[
 n_i = \bar{i}_m \cos(\theta_i) \\
 e_i = \bar{i}_m \sin(\theta_i),
\]

where \( \theta_i \) is the integrator angle given by

\[
 \theta_i = \text{atan2}(e_i, n_i).
\]

The outputs of the north and east position PID loops are

\[
 n_{\text{PID}} = k_{p,p} \ddot{p}_n - n_d + k_{i,p} n_i \quad (5.11) \\
 e_{\text{PID}} = k_{p,p} \ddot{p}_e - e_d + k_{i,p} e_i, \quad (5.12)
\]

where \( n_d \) and \( n_e \) are the damping terms given by

\[
 n_d = k_{d,p} \dot{p}_n \quad (5.13) \\
 n_e = k_{d,p} \dot{p}_e. \quad (5.14)
\]

The PID outputs are used to generate a commanded tilt

\[
 \theta_i^c = \sqrt{n_{\text{PID}}^2 + e_{\text{PID}}^2},
\]

where \( \theta_i^c = 0 \) is vertical, and a tilt direction

\[
 \theta_i^c = \text{atan2}(e_{\text{PID}}, n_{\text{PID}}),
\]

which are shown in Figure 5.4. The tilt command is saturated by

\[
 \theta_i^c = \text{sat}(\theta_i^c, 0, \bar{\theta}_i)
\]
and merged with a vertical quaternion and the commanded heading to generate the commanded attitude

\[ \eta^c = \begin{bmatrix} \cos \left( \frac{\psi^c}{2} \right) \\ - \sin \left( \frac{\psi^c}{2} \right) \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} \frac{\sqrt{2}}{2} \\ 0 \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix} \otimes \begin{bmatrix} \cos \left( \frac{\theta^c t}{2} \right) \\ \sin \left( \frac{\theta^c t}{2} \right) \nu_{\psi} \end{bmatrix}, \]

where

\[ \nu_{\psi} = \begin{bmatrix} \cos \left( \theta_{\psi} - \frac{\pi}{2} \right) \\ \sin \left( \theta_{\psi} - \frac{\pi}{2} \right) \\ 0 \end{bmatrix}. \]

### 5.2.2 Altitude Controller

The altitude controller is a PID loop based on the altitude error. Let \( \tilde{h} \) be the altitude error and \( h_i \) be

\[ h_i = \int_{t_0}^{t} \tilde{h} \delta \tau. \]

The integrator is saturated such that \( |k_i h_i| \leq \tilde{h}_i \). The throttle command is

\[ \delta_t = k_{p,\bar{h}} \bar{h} - k_{d,\bar{h}} \dot{\bar{h}} + k_{i,\bar{h}} h_i + \delta_{tff}, \]
where $\delta_{t_ff}$ is the estimated throttle command needed to maintain hover. Because the throttle command needed to maintain altitude constantly changes due to decreasing battery voltage the throttle integrator is initialized by

$$h_i = \frac{1}{k_{i,h}} \left( \delta_i - k_{d,h} \dot{h} - \delta_{t_ff} \right).$$

(5.15)

This is not without risk. If the throttle is initially too large, when the controller is initialized, then the integrator will be too high and the V-Bat will climb before the integrator is able to unwind. Likewise if the throttle command is too low then the V-Bat will descend before the integrator is able to correct itself. While the first situation is inconvenient, the second case is potentially dangerous. To counter this, the integrator is set to zero when the controller is initialized and $\dot{h} < v_h$ and $\delta_t < \delta_{t_ff}$.

5.3 PID Modifications

While the position and altitude reference models can smoothly handle the case where the controller is initialized outside the normal hover operating region, the hover control scheme presented in Section 5.2.1 cannot. For example, consider the case shown in Figure 5.5 where the hover control is initialized when the vehicle is moving at 5 m/s, it is commanded to hold its current position, and the position damping gain was $k_{d,p} = 0.7$. Immediately after the controller is activated, $\theta^*_t = 240^\circ$ and is immediately saturated to $\bar{\theta}$. This large attitude error causes a full elevator command which causes the V-Bat perform a sharp maneuver and decelerate much faster than the reference model commands. Eventually the error between the actual and the reference model position overpowers the damping term and the aircraft is commanded to tilt forward to accelerate This causes the aircraft’s velocity to oscillate around the reference model’s velocity.

Although the V-Bat should be able to survive this maneuver during a real flight, as shown by simulation results, it is still undesirable. The position controller needs to be modified to obtain a better response to this type of situation. Several modifications will be discussed in the remainder of the section.
5.3.1 Approaches

There are several modifications to the PID controller that we have considered. Some of these would always be applied to the hover control scheme while others would only be used when the decelerate reference models are in use. The modifications are:

1. **No Modifications** The controller given by (5.11) and (5.12) is used with no modifications.

2. **Saturated Damping Term** – Integrators typically have limits in order to prevent integrator windup and other issues. Similarly, the damping term is limited to reduce the large initial tilt command when the controller is initialized with a large initial velocity. If the V-Bat’s horizontal velocity is above the limit \( \sqrt{\dot{\rho}_n^2 + \dot{\rho}_c^2} > \bar{v}_p \) then the damping terms are

\[
\begin{align*}
    n_d &= k_{d,p} \bar{v}_p \cos(\theta_d) \\
    e_d &= k_{d,p} \bar{v}_p \sin(\theta_d),
\end{align*}
\]

where

\[
\theta_d = \tan^{-1}\left(\frac{\dot{\rho}_c}{\dot{\rho}_n}\right).
\]
Otherwise the damping terms (5.13) and (5.14) are used. Note that this approach is
dependent on $k_{d,p}$. If $k_{d,p} \bar{v}_p > \bar{\theta}_t$ then saturating the damping term will have no effect
and there will still be a large initial jump in the desired tilt if the controller is activated
when the vehicle is moving fast.

3. **Velocity Error** - The position reference model provides a desired velocity that is
unused. If the damping term is changed to

\[ n_d = k_{d,p}(\dot{p}_n - \dot{p}_n^d) \]
\[ e_d = k_{d,p}(\dot{p}_e - \dot{p}_e^d), \]

then the vehicle will attempt to track the desired velocity in addition to the desired
position.

4. **Limited Use Velocity Error** - The main problem with method 3 is that it modifies
the control scheme when the vehicle is in the normal hover operating conditions. This
approach only uses the modified damping term when the decelerate reference model is
being used.

5. **Integrator Initialization** - In all of the previous approaches, the tilt command is
very different from the actual tilt when the controller is initialized. We can force the
desired tilt to equal the initial tilt by initializing the integrator properly.

When the controller is initialized $\bar{p}_n = 0$ and $\bar{p}_e = 0$, so

\[ n_{PID} = -n_d + k_{i,p}n_i \] \hspace{1cm} (5.16)
\[ e_{PID} = -e_d + k_{i,p}e_i. \] \hspace{1cm} (5.17)

To ensure the tilt command matches the current tilt, (5.16) and (5.17) must satisfy

\[ \theta_t = \sqrt{n_{PID}^2 + e_{PID}^2} \] \hspace{1cm} (5.18)
\[ \theta_\psi = \tan^{-1}\left(\frac{n_{PID}}{e_{PID}}\right). \] \hspace{1cm} (5.19)
Inserting (5.16) and (5.17) into (5.19) and rearranging provides

\[ k_{i,p} n_i = k_{i,p} e_i \cot(\theta_\psi) + n_d - e_d \cot(\theta_\psi). \] (5.20)

Inserting (5.20) into (5.18) and performing some algebraic manipulation provides

\[ (1 + \cot(\theta_\psi)^2) e_i^2 - (2e_d - 2(n_d - t) \cot(\theta_\psi)) e_i - \theta_t^2 - n_d - e_d - t^2 + tn_d = 0, \] (5.21)

where

\[ t = n_d - e_d \cot(\theta_\psi). \]

Using the quadratic formula on (5.21) provides two options for the east integrator term \( e_{i,1} \) and \( e_{i,2} \), two possible north integrator terms, \( n_{i,1} \) and \( n_{i,2} \), and two possible tilt angles, \( \theta_{\psi,1} \) and \( \theta_{\psi,2} \). One of the \( \theta_{\psi,i} \) equals \( \theta_\psi \) while the other is 180 degrees off.

The integrator terms, \( n_i \) and \( e_i \), are set to the values that correspond to the correct \( \theta_{\psi,1} \). If \( \theta_\psi = 0 \) or \( \theta_\psi = \pm \pi \) then the previous approach will not work. In this case the integrators are initialized by

\[ n_i = \frac{n_d + \cos(\theta_\psi) \theta_t}{k_{i,p}} \]
\[ e_i = \frac{e_d}{k_{i,p}}. \]

6. **Integrator Initialization and Saturated Damping Term** - This is a combination of methods 2 and 5.

7. **Integrator Initialization and Limited Use Velocity Error** - This is a combination of methods 4 and 5.

**5.3.2 Comparison**

The PID modifications were tested for a stationary initialization and a moving initialization case and are compared based on the following criteria:
1. **Stability** - The primary consideration is if the vehicle goes unstable, defined to be \( \theta_t \geq 80^\circ \), during the maneuver.

2. **Maximum Actuator Command** - The maneuver should be smooth with no large jumps in the commanded tilt. This is measured by looking at the largest aileron and elevator commands because large commands correspond to a large tilt error.

3. **Position Overshoot** - The position overshoot should be minimized.

The velocity limit is 2 m/s in each test.

Figure 5.6a shows the controllers’ response when they are initialized while the vehicle is stationary and commanded to move ten meters. Note that methods 1, 2, 4, 5, 6, and 7 have an identical acceptable, if conservative, response to the position command. However, method 3 causes the vehicle to be overly aggressive which causes it to exceed the velocity limits. Figure 5.6b shows the controllers’ response when they are initialized while the vehicle is moving at 5 m/s and commanded to hold its current position. As discussed previously, the unmodified PID controller has a large initial elevator command which causes the vehicle to decelerate faster than the reference model commands which causes the elevator command and the velocity to oscillate. Method 2 dampens the response but it is still aggressive and decelerates too fast. Once again method 3 causes the vehicle to be too aggressive. Method 4 significantly reduces the overcorrection, however, it initially has a full elevator command. Method 5 provides the smoothest and most controlled deceleration, however, it has a very large overshoot. The overshoot is reduced by method 6 but this also increases the velocity oscillation. Finally, method 7 provides the best overall response and it significantly reduces the overshoot compared to methods 5 and 6 while not having a large oscillation in either the velocity or the elevator command.

In addition, methods 1, 2, 4, 5, 6, and 7 were tested with a large set of initial conditions. Most of these initial conditions, shown in Table 5.1, are worst case conditions unlikely to ever occur and start the vehicle far away from any trim condition. In any case, each control method was tested for every permutation, except the cases when \( |v_n| = |v_e| = 8 \) m/s, and compared based on the previously mentioned stability criterion. We did not expect that any control method would be able to handle every case. Figure 5.7 shows the percentage
Method:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
</table>

(a) V-Bat is initially stationary and is commanded to move 10 meters. Note that methods 1, 2, 4, 5, 6, and 7 have an identical response.

(b) V-Bat is initially moving at 5 m/s and commanded to hold its current position. Note that the large spikes around three seconds for methods 4 and 7 are caused by switching to the normal reference model so the velocity is used instead of the velocity error.

Figure 5.6: Simulation results comparing the response of the PID modifications of the tests where the maximum tilt angle reached by the V-Bat is below the specified tilt angle given by the $x$-axis. Note that a simulation was ended if the V-Bat reached $\theta_t = 80^\circ$.

Specifically, Figure 5.7a shows the cases where the vehicle was initially vertical and Figure 5.7b shows the cases where the vehicle was initially tilted. These plots show that initializing the integrator and using the limited velocity error had the lowest maximum tilt for the largest number of tests. The unmodified controller had the worst results when the vehicle was initially vertical, and initializing the integrator while saturating the damping term had the worst results when the vehicle was titled.

5.4 Analysis

Due to the complex nature of the switching reference model and the controllers, the hover control system is difficult to analyze. However, it becomes feasible with several simplifying assumptions.
Table 5.1: Initial conditions for the hover controller initialization tests. Note that the attitude is given in ZYX Euler angles

<table>
<thead>
<tr>
<th>Initial condition</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>North position error (m)</td>
<td>$\tilde{p}_n \in {0, 10}$</td>
</tr>
<tr>
<td>East position error (m)</td>
<td>$\tilde{p}_e \in {0, 10}$</td>
</tr>
<tr>
<td>Down position error (m)</td>
<td>$\tilde{p}_d \in {-10, 0, 10}$</td>
</tr>
<tr>
<td>North velocity (m/s)</td>
<td>$v_n \in {-8, -4, 0, 4, 8}$</td>
</tr>
<tr>
<td>East velocity (m/s)</td>
<td>$v_e \in {-8, -4, 0, 4, 8}$</td>
</tr>
<tr>
<td>Down velocity (m/s)</td>
<td>$v_d \in {-3, 0, 3}$</td>
</tr>
<tr>
<td>Attitude (deg)</td>
<td>$\eta \in {[0, 90, 0],[0, 50, 0],[30, 50, 0]}$</td>
</tr>
<tr>
<td>North wind (m/s)</td>
<td>$w_n \in {-3, 0, 3}$</td>
</tr>
<tr>
<td>East wind (m/s)</td>
<td>$w_e \in {-3, 0, 3}$</td>
</tr>
<tr>
<td>Down wind (m/s)</td>
<td>$w_d \in {-2, 0, 2}$</td>
</tr>
</tbody>
</table>

![Graphs of percentage of tests below max tilt](image)

(a) Vehicle is initially vertical

(b) Vehicle is initially tilted

Figure 5.7: Percentage of the tests where $\max \theta_c(t) \leq \theta$

5.4.1 Position Control

We first assume that the north and east components of the position controller and the aircraft’s response are decoupled which simplifies the analysis to one dimension instead of both. While this assumption is not exactly true, it is reasonable. The decoupled position controller generates commanded tilts that are typically within half a degree of the coupled position controller’s commands. In addition, many other position control systems assume that the aircraft’s position response is coupled. Next, we assume that the heading and altitude are constant and that no saturation limits have been reached. Finally, we assume
that the tilt tracking system can be modeled as a second-order system with a DC gain of $\epsilon$. With these assumptions the linearized one dimension position control system can be represented by Figure 5.8 where $\mu$ is the drag coefficient, $\omega_r$ and $\zeta_r$ are the natural frequency and damping ratio of the reference model, and $\omega_p$ and $\zeta_p$ are the natural frequency and damping ratio of the tilt response.

By assuming that the tilt response is much faster than the position response, which allows the tilt second-order system to be replaced by its DC gain, the transfer function from the position command to the position is

$$P(s) = \frac{\epsilon g \omega^2_p (k_p s + k_i)}{(s^2 + 2\zeta_r \omega_r s + \omega^2_r)(s^3 + (\mu + \epsilon g k_d)s^2 + \epsilon g k_p s + \epsilon g k_i)} P^c(s).$$

We know that the roots of $(s^2 + 2\zeta_r \omega_r s + \omega^2_r)$ are in the left half plane, otherwise the reference model would be unstable. Therefore, the system is stable if the roots of $s^3 + (\mu + \epsilon g k_d)s^2 + \epsilon g k_p s + \epsilon g k_i$ are in the left half plane. Using the Routh stability criterion we can conclude that the system is stable if $(\mu + \epsilon g k_d) k_p > k_i$ [39, p. 131]. If the system is stable then the final value theorem can be applied which states that $p(t) \to p^c(t)$ as $t \to \infty$ for a step position command. Thus we can conclude that the position system is stable and the aircraft will eventually reach the commanded position with zero steady state error.

The previous analysis assumes that the position controller is initialized within the normal operating mode. A similar analysis will be performed for the deceleration mode. The linearized version of the deceleration position controller is shown in Figure 5.9 where the same assumptions have been made. The transfer function between the commanded and
the actual velocity is

\[ V(s) = \frac{2\zeta_r \omega_r \epsilon g (k_d s^2 + k_p s + k_i)}{(s + 2\zeta_r \omega_r)(s^3 + (\mu + \epsilon g k_d)s^2 + \epsilon g k_p s + \epsilon g k_i)} V^c(s). \]

Note that the commanded velocity will always be 0 m/s for the deceleration system. The deceleration system shares the same stability requirements as the normal position system so we can conclude that the deceleration mode does not break the stability guarantee. The final value theorem also states that the \( v(t) \rightarrow v^c(t) \) as \( t \rightarrow \infty \) so we know that the velocity will eventually be less than the velocity limit so the normal position mode will eventually be reached. In addition, because the reference model and the integrators are reinitialized when the normal reference model takes control, we can conclude that the mode switching will not cause stability issues.

### 5.4.2 Altitude Control

In order to analyze the altitude control scheme we make the following assumptions:

- The aircraft is vertical,
- No saturation limits have been hit,
- The controller is able to control the thrust directly,
- The thrust dynamics are much faster than the altitude dynamics.

With these assumptions, the linearized version of the normal altitude control system is shown in Figure 5.10 where \( v \) is the climb rate, \( \mu \) is the drag coefficient, and \( \omega_r \) and \( \zeta_r \) are the natural
frequency and damping ratio of the reference model. Note that gravity is compensated for by
the feedforward term and does not show up in the simplified linearized model. The transfer
function between the commanded and the actual altitude is

\[
H(s) = \frac{\frac{\omega_r^2}{m} (k_p s + k_i)}{(s^2 + 2\zeta \omega_r s + \omega_r^2) \left( s^3 + (\mu + \frac{k_d}{m}) s^2 + \frac{k_p}{m} s + \frac{k_i}{m} \right)} H^c(s).
\]

Using the same approach that was used for the position control system analysis, we can
conclude that the altitude control system is stable if \((\mu + \frac{k_d}{m}) k_p > k_i\), and that the system
has no steady state error when tracking step altitude commands.

If the altitude controller is initialized when the aircraft is climbing too fast then the
climbing reference model is used. Figure 5.11 shows the linearized climbing control system,
and the transfer function between the commanded climb rate and the actual climb rate is

\[
V(s) = \frac{2\zeta \omega_r}{m} \frac{\left( k_d s^2 + k_p s + k_i \right)}{\left( s + 2\zeta \omega_r \right) \left( s^3 + (\mu + \frac{k_d}{m}) s^2 + \frac{k_p}{m} s + \frac{k_i}{m} \right)} V^c(s).
\]

This system is stable if the normal altitude control system is stable and we can conclude
that \(v_h(t) \to 0\) as \(t \to \infty\) so the climb rate will eventually drop below the limit and the
normal altitude control mode will be reached.

If the altitude controller is initialized when the aircraft is descending too fast then
the descending reference model is used. Figure 5.12 shows the linearized descending control
system. The transfer function from the commanded climb rate to the actual climb rate is

\[
V(s) = \frac{\frac{1}{m} \left( k_d s^2 + k_p s + k_i \right)}{\left( s^3 + (\mu + \frac{k_d}{m}) s^2 + \frac{k_p}{m} s + \frac{k_i}{m} \right)} V^c(s).
\]
Again, this system is stable if the normal altitude control system is stable, and the velocity steady state error is zero for step commands. This means we can guarantee that normal altitude mode will eventually be reached.

Because the altitude reference model and controller are reinitialized when then normal altitude mode is activated, we do not need to take into account the switching between the modes. Therefore, we can conclude that, assuming the listed assumptions are valid, the aircraft’s controlled altitude dynamics are stable no matter what the initial climb rate is. In addition, the normal altitude control mode will eventually be reached.

5.5 Flight Tests

The hover control system was extensively flight tested with a wide variety of scenarios some of which are presented here. For each test $\bar{v}_p = 3 \text{ m/s}$, $\bar{a}_p = 1.5 \text{ m/s}^2$, $\bar{v}_h = 1 \text{ m/s}$, $\bar{a}_h = -0.75 \text{ m/s}$, and $\bar{a}_h = 2 \text{ m/s}^2$. Note that unless otherwise noted the figures only show the telemetry while the autopilot was in control. All of the tests were performed when the wind speed was less than five miles per hour. The following list explains the procedure for a hover flight test.
1. The ESC is calibrated by holding the plane down and ramping the throttle command from 0% to 100%. The ESC that is used on the BYU V-Bat auto calibrates the scale for the throttle command. By default the ESC will command 100% of the thrust when the autopilot is only commanding 50% but the ESC’s scale will change based on the largest value it receives from the autopilot. If the full range of the throttle values are not sent to the ESC before the flight begins, the ESC’s scale can change in flight.

2. The RC pilot takes the plane off while someone holds onto a wing to stabilize the plane. Once the plane is a meter off the ground, the person lets go of the plane and the RC pilot has the plane climb.

3. The autopilot is given control and the test is performed.

4. Either the RC pilot takes control of the plane and lands it or the autopilot is commanded to land. Someone grabs one of the wings to guide the plane to the ground when the plane is low enough.

5.5.1 Stationary Initialization

The first set of tests involved initializing the hover controller while the V-Bat was approximately stationary and vertical. This starting condition is ideal and should be similar to the normal starting conditions in a real flight.

In the first test, the V-Bat is commanded to hold a position and altitude for 50 seconds. The V-Bat’s position and altitude are shown in Figure 5.13a, and Figure 5.13b shows a top down view of the vehicle’s position. Note that the vehicle stays within a square meter of the desired location and the altitude error never exceeds half a meter.

In the second test, the V-Bat is commanded to hold a constant altitude while the desired position is changed. Figure 5.14a shows the position and altitude, Figure 5.14b shows a top down view of the position, and Figure 5.14c shows the velocity. Note that there is approximately a two second delay between the desired and the actual position and velocity. This delay could be removed by adding a feed forward term to (5.11) and (5.12) or by making the attitude control more aggressive.
In the third test, the V-Bat is commanded to change its altitude and position. The V-Bat flew between four waypoints, roughly eight meters apart, twice. During the first iteration the V-Bat is commanded to maintain a constant altitude while the altitude varies for two waypoints in the second iteration. Figure 5.15a shows the position and altitude during a portion of the flight, Figure 5.15b shows a top down view of the position, and Figure 5.15c shows the velocity. Note that the V-Bat is not allowed to settle down before it is commanded to go to the next waypoint which explains why the position does not follow the desired position between the waypoints.

5.5.2 Moving Initialization

In the second set of tests, the hover controller was initialized while the V-Bat was moving above the velocity limit. These situations could occur when the RC pilot gives control to the autopilot while he is flying or when the autopilot switches from the hover to level transition to hover flight modes.

In the first test, the hover controller is initialized and the V-Bat is commanded to hold its current position and altitude when the V-Bat was moving at 4 m/s. Because this is greater than the velocity limit, the autopilot first attempts to slow the V-Bat down and then
Figure 5.14: Flight data for a constant altitude waypoint following hover flight test

Figure 5.15: Flight data for a varying altitude waypoint following hover flight test
move it to the commanded location. The position and altitude are shown in Figure 5.16a, the velocity is shown in Figure 5.16b, and Figure 5.16c shows the magnitude of the velocity error as well as the control flags. A velocity flag of 1 means the decelerate reference model is being used and 0 means the normal reference model is used. A control value of 0 means the V-Bat is being controlled by the RC pilot and 1 means the autopilot has control. In addition the shaded region is where the autopilot has control. Note that there is not a large jump in the elevator command, shown in Figure 5.16d, when the autopilot takes control which causes the vehicle to smoothly decelerate to the normal operating velocity. The large jump in $\delta_t$ around 35 seconds is caused by the RC pilot when he took control.
Figure 5.17: Climbing initialization hover flight test

In the second test, the hover controller is initialized when the V-Bat is climbing faster than the climb rate limit. Figure 5.17a shows the position and altitude, Figure 5.17b shows the velocity, and Figure 5.17d shows the commands. Notice that the climb rate continues to increase when the autopilot takes control as shown in Figure 5.17c. When the autopilot takes control the throttle integrator is initialized such that the throttle command does not immediately change. While this prevents the vehicle from undergoing a rapid climb or descent if the feedforward term is wrong, it can cause the observed behavior. In this case, the RC pilot is commanding a $\delta_t$ that would cause the vehicle to continue to accelerate and it takes several seconds for the throttle integrator to unwind.
5.6 Conclusions

In this chapter we developed a simple tailsitter hover control scheme which uses simple second-order reference models with saturation and standard PID loops for control. While better performance could be obtained by using more advanced control techniques, such as LQR or MRAC, this control scheme does not require any aerodynamic model of the vehicle and is straightforward to tune. In addition, we explored several modifications to the PID controller that improved its performance when it was initialized at high speeds and concluded that initializing the integrator and using the velocity error, instead of velocity, when the vehicle is moving too fast provides the best performance. While we currently do not have any theoretical stability guarantees, simulation results show that this control scheme can safely handle a wide variety of initial conditions. Finally, flight tests showed that this control scheme is physically feasible and provides acceptable real-world performance.
In level flight a tailsitter acts as a normal fixed-wing aircraft. Because of this, standard level flight control techniques can easily be applied to a tailsitter. In most control schemes the longitudinal, altitude and airspeed, and lateral, position and course, are assumed to be decoupled and are controlled separately. We will follow this tradition and have separate longitudinal and lateral control schemes. In this chapter we assume that we do not have a high-fidelity aerodynamic model of the aircraft and that low-cost sensors are used.

A common approach to control the airspeed and altitude for a fixed-wing aircraft is to assume that the airspeed and altitude dynamics are decoupled. Using this assumption, the altitude is controlled by the elevator while the thrust controls the airspeed [16]. While this type of controller has been used successfully on many autopilots, the underlying assumption that the airspeed and altitude dynamics are decoupled is inherently false. For example, consider the simple case when an aircraft pitches up while the thrust does not change. If the dynamics were truly decoupled, the aircraft’s altitude would increase while the airspeed would remain unchanged. However, as should be obvious, the airspeed will decrease while the altitude increases. In other words, some of the aircraft’s kinetic energy is converted to potential energy.

In the early 1980’s, Lambregts et al. realized that the airspeed and altitude of an aircraft could be controlled by manipulating the kinetic and potential energy of the system [65–67]. Approaching the problem this way allowed the coupling between the altitude and airspeed to be taken into account. Controllers based on this idea, called the Total Energy Control System (TECS), have been successfully used and tested on a variety of airframes [20,21,114]. While most of these controllers are of PI type, other variants have been developed such as adding pitch damping [113]. Viswanathan et al. use L1 adaptive control to make the
energy rates follow a reference model which eliminates the need for gain scheduling [112].

TECS ideas have also been explored to see if cockpit displays could be developed to present pilots with this information [8] and how they could be used as instructional tools [77].

TECS concepts have also been applied to the lateral control of an aircraft [22, 47], for the longitudinal control of a helicopter [29], and the translational control of a quadcopter [111].

In 2010, Akmeliawati and Mareels derived a nonlinear energy based control method for the longitudinal control of a fixed-wing airplane that has some similarities to the TECS [7]. In their approach they decompose the dynamics into fast dynamics, which account for the pitch and pitch rate, and slow dynamics, which account for everything else. Elevator and throttle controllers are then derived which provide stability and tracking guarantees. However, their controller requires accurate model parameters and model inaccuracies of only $\pm 2\%$ have been tested.

In this chapter, we derive a nonlinear TECS altitude and airspeed controller that is easy to apply to different airframes. Two variants of this controller are compared, through simulation, to the original TECS controller, several other TECS type controllers, and a standard decoupled successive loop closure controller. These simulations show that the nonlinear TECS controllers account for the coupling between the airspeed and altitude dynamics while having smaller oscillations in the response as compared to the original TECS control scheme. In addition, the nonlinear controllers have a better response than the other controllers.

This chapter is organized in the following manner. Section 6.1 defines the various energy values and their derivatives. The original TECS control scheme is described in Section 6.2. Section 6.3 contains the motivation and derivation for the nonlinear TECS controller, and Section 6.4 describes simple pitch and thrust control schemes that can be used while implementing the nonlinear TECS controller in hardware. For comparison, three other controllers are briefly described in Section 6.5. The simplified three degree of freedom dynamic model is described in Section 6.7, a simple parameter estimation scheme is presented in Section 6.8, and simulation results are shown in Section 6.9. Finally, conclusions are presented in Section 6.10.
6.1 Energy Definitions

If the aircraft is modeled as a point mass then the total energy is the kinetic energy plus the potential energy,

\[
E_T = mgh + \frac{1}{2}mV_a^2,
\]

and the total energy rate is

\[
\dot{E}_T = mgh + mV_a \dot{V}_a.
\]

The energy difference is defined to be the potential energy minus the kinetic energy,

\[
E_D = mgh - \frac{1}{2}mV_a^2,
\]

and the energy distribution rate, or the energy difference rate, is

\[
\dot{E}_D = mgh - mV_a \dot{V}_a.
\]

The airspeed, acceleration, altitude, and climb rate are easily computed from the energy definitions and are

\[
V_a = \left( \frac{E_T - E_D}{m} \right)^{\frac{1}{2}},
\]

\[
\dot{V}_a = \left( \frac{m}{E_T - E_D} \right)^{\frac{1}{2}} \left( \dot{E}_T - \dot{E}_D \right),
\]

\[
h = \frac{E_D + E_T}{2mg},
\]

\[
\dot{h} = V_a \sin(\gamma) = \frac{\dot{E}_D + \dot{E}_T}{2mg}.
\]

Typically, TECS controllers use a scaled version of the total energy rate and energy distribution [35]. Rearranging (6.2) provides

\[
\dot{E}_t = \frac{\dot{E}_T}{mgV_a} = \frac{\dot{V}_a}{g} + \frac{\dot{h}}{V_a}.
\]
If we assume that $\gamma$ is small then (6.6) becomes

$$\dot{E}_t = \frac{\dot{V}_a}{g} + \gamma,$$

which is the standard total energy rate value used in most TECS control schemes. Likewise, the energy distribution rate, (6.4), can be rearranged to

$$\dot{E}_d = \frac{\dot{E}_D}{mg} = -\frac{\dot{V}_a}{g} + \gamma.$$

### 6.2 Original TECS

Every TECS altitude and airspeed controller shares the same basic principles. First, kinetic energy, or the airplane’s airspeed, can be converted into potential energy, or altitude, and vice versa. Second, assuming a point mass model of the aircraft, the thrust generated by the propulsion system is the only way to add energy to the system and drag is the only way that energy is removed. TECS controllers also assume that the angle of attack is low and the flight path angle does not influence drag. Using these basic principles and assumptions the TECS longitudinal controller can be developed.

As mentioned earlier, the total energy rate is controlled by the thrust which implies that the commanded thrust should be a function of $\dot{E}_t$. Likewise, the elevator is approximately energy conservative and allows kinetic energy and potential energy to be converted to each other which implies that the elevator should be used to control the energy distribution rate. Using these ideas, the thrust command is

$$T^e = T_D + \Delta T,$$

where $T_D$ is the trim thrust needed to counteract drag and

$$\Delta T = k_{p,T} \dot{E}_t + k_{i,T} \int_{t_0}^{t} \left( \dot{E}^d - \dot{E}_t \right) \delta \tau,$$  \hspace{1cm} (6.7)
where $k_{i,T}$ and $k_{p,T}$ are the integral and proportional gains respectively and $\dot{E}^c_t$ is the commanded total energy rate computed from the commanded acceleration and flight path angle. Likewise, the pitch command is

$$\theta^c = k_{p,\theta} \dot{E}_d + k_{i,\theta} \int_{t_0}^{t} (\dot{E}^d_d - \dot{E}_d) \delta\tau,$$

(6.8)

where $k_{i,\theta}$ and $k_{p,\theta}$ are the integral and proportional gains respectively and $\dot{E}^c_d$ is the commanded energy distribution rate. This control scheme assumes that there are fast, low-level control loops that control $T$ and $\theta$. The desired acceleration and flight path angle are created by a proportional term [66]

$$\dot{h}^d = k_h (h^c - h),$$

(6.9)

$$\gamma^d = \frac{\dot{h}^d}{V_a},$$

$$\dot{V}_a^d = k_v (V^c_a - V_a),$$

(6.10)

where $k_h > 0$ and $k_v > 0$ are scaling parameters. In other words the desired acceleration (climb rate) is the scaled airspeed (altitude) error.

Over the years, variations of the controller given by (6.7) and (6.8) have been developed. For example, a pitch damping term, $-k_{d,\theta} q$, is added to (6.8) in [113]. In addition there have been several different approaches to tune the gains. Faleiro and Lambregts used eigenstructure assignment to systematically select the gains in [35] and an optimization algorithm with a cost function that included the mean square of the control response needed to reject Dryden turbulence and command tracking performance, among others, selected the gains in [113]. In addition, the preferred structure of the controller was changed to improve performance. The thrust controller does not change but the pitch control scheme is modified to [68]

$$\theta^c = k_{p,\theta} \left( (2 - k) \gamma - k \frac{\dot{V}_a}{g} \right) + k_{i,\theta} \int_{t_0}^{t} (\dot{E}^d_d - \dot{E}_d) \delta\tau.$$

(6.11)

In this method, the true energy distribution rate is not used. Rather a weighted difference between the potential and kinetic energy rates is used. The weighting is determined by
$k \in [0, 2]$, where $k = 0$ means only the scaled potential energy rate is used, $k = 2$ means only the scaled kinetic energy rate is used, and $k = 1$ means the true energy distribution rate is used.

### 6.3 Nonlinear TECS

The TECS concept can be used to derive an adaptive nonlinear control scheme which attempts to control $E_T$ and $E_D$ directly. As mentioned, the total energy can only change due to the thrust or drag on the aircraft otherwise the energy is just transitioning between kinetic and potential. This means that the total energy rate can be derived by looking at the case where the velocity changes and the altitude is constant. Assuming $\dot{h} = 0$, the total energy rate is

$$\dot{E}_T = mV_a\dot{V}_a = V_a F,$$

where $F$ is the net non-conservative force on the aircraft. Assuming that the angle of attack is low and that the thrust and drag are aligned then

$$\dot{E}_T = V_a(T - D).$$

(6.12)

In general, when $\dot{h} \neq 0$,

$$m g \dot{h} + mV_a\dot{V}_a = \dot{E}_T = V_a(T - D),$$

which means

$$\dot{E}_D = 2m g \dot{h} - V_a(T - D).$$

(6.13)

Because the drag is unknown we model it as

$$D = \hat{D} + \phi^T(x)\Psi,$$

where $\hat{D}$ is the aerodynamic model’s estimate of drag, $\phi(x) \in \mathbb{R}^n$ is a vector of known bounded basis functions, and $\Psi \in \mathbb{R}^n$ is a vector of unknown parameters.
Let the Lyapunov function be

\[ V = \frac{1}{2} \Gamma_T \tilde{E}_T^2 + \frac{1}{2} \Gamma_D \tilde{E}_D^2 + \tilde{\Psi}^\top \tilde{\Psi}, \quad (6.14) \]

where \( \tilde{E} \triangleq E^d - E \) is the error, \( \Gamma_T > 0 \), and \( \Gamma_D > 0 \). Taking the derivative of (6.14) and using (6.5), (6.12), and (6.13) gives

\[
\dot{V} = \Gamma_T \tilde{E}_T \left( \tilde{E}_T^d - \left( T - \dot{D} - \phi^\top \Psi \right) V_a \right) \\
+ \Gamma_D \tilde{E}_D \left( \tilde{E}_D^d - 2mgV_a \sin \gamma + \left( T - \dot{D} - \phi^\top \Psi \right) V_a \right) \\
+ \tilde{\Psi}^\top \dot{\tilde{\Psi}}, \quad (6.15)
\]

where we assume that the unknown parameters are slowly varying. If the thrust is chosen to be

\[
T^e = \dot{D} + \phi^\top \dot{\tilde{\Psi}} + \frac{\dot{E}_T^d}{V_a} + k_T \tilde{E}_T, \quad (6.16)
\]

where \( k_T > 0 \), then (6.15) becomes

\[
\dot{V} = -k_T \Gamma_T \tilde{E}_T^2 \\
+ \Gamma_D \tilde{E}_D \left( \dot{E}_D^d - 2mgV_a \sin \gamma + \dot{E}_T^d + k_T \tilde{E}_T \right) \\
+ \tilde{\Psi}^\top \left( \dot{\tilde{\Psi}} + \left( -\Gamma_T \tilde{E}_T + \Gamma_D \tilde{E}_D \right) \phi V_a \right). \quad (6.17)
\]

If the flight path angle is chosen to be

\[
\gamma^e = \sin^{-1} \left( \frac{\dot{E}_D^d + \dot{E}_T^d + k_T \tilde{E}_T + k_D \tilde{E}_D}{2mgV_a} \right), \quad (6.18)
\]

where \( k_D > 0 \), then (6.17) becomes

\[
\dot{V} = -k_T \Gamma_T \tilde{E}_T^2 - k_D \Gamma_D \tilde{E}_D^2 \\
+ \tilde{\Psi}^\top \left( \dot{\tilde{\Psi}} + \left( -\Gamma_T \tilde{E}_T + \Gamma_D \tilde{E}_D \right) \phi V_a \right). \quad (6.19)
\]
Note that (6.18) can be simplified, by using (6.5), to
\[
\gamma^c = \sin^{-1}\left(\frac{\dot{h}^d}{V_a} + \frac{1}{2mgV_a}\left(k_T\tilde{E}_T + k_D\tilde{E}_D\right)\right)
\]
(6.20)
and that the commanded flight path angle can be converted to a pitch command by \(\theta^c = \gamma^c + \alpha\).

The parameter adaption rate is chosen to be
\[
\dot{\Psi} = \left(\Gamma_T\tilde{E}_T - \Gamma_D\tilde{E}_D\right)\phi V_a,
\]
(6.21)
so (6.19) becomes
\[
\dot{V} = -k_T\Gamma_T\tilde{E}_T^2 - k_D\Gamma_D\tilde{E}_D^2,
\]
(6.22)
which is negative semi-definite. By realizing that \(V\) is lower bounded and assuming that the desired altitude and airspeed are bounded, we know that \(V(t) \leq V(t_0) \forall t \geq t_0\). This implies that \(\tilde{E}_T, \tilde{E}_D,\) and \(\dot{\Psi}\) are bounded. The derivative of (6.22) is
\[
\ddot{V} = -2k_T\Gamma_T\dot{\tilde{E}}_T\dot{\tilde{E}}_T - 2k_D\Gamma_D\dot{\tilde{E}}_D\dot{\tilde{E}}_D.
\]
(6.23)
Inserting the error rates and the commanded thrust and flight path angle into (6.23) provides
\[
\ddot{V} = -2k_T^2\Gamma_T\tilde{E}_T^2 - 2k_D^2\Gamma_D\tilde{E}_D^2
+ 2\dot{\Psi}\left(k_T\Gamma_T\dot{\tilde{E}}_T - k_D\Gamma_D\dot{\tilde{E}}_D\right)\phi V_a.
\]
which is bounded and finite. From this we can conclude, using Barbalat’s Lemma [55], that \(\dot{V} \to 0\) which implies \(\tilde{E}_T \to 0\) and \(\tilde{E}_D \to 0\).

There are several interesting things about the nonlinear TECS controller given by (6.16) and (6.20). First, if we assume that the drag is known and the thrust has been trimmed to counteract drag then the commanded change in thrust is.
\[ \Delta T = \frac{\dot{E}_T}{V_a} + k_T \frac{\tilde{E}_T}{V_a}, \]

which can be thought of as a PI controller where the proportional gain is 1 and the integral gain is \( k_T \). This is very similar to the original TECS controller (6.7). However, this derivation suggests that the desired total energy should be used instead of the actual total energy.

Second, using the energy definitions (6.1) and (6.3) (6.20) becomes

\[
\sin(\gamma^c) = \frac{\dot{h}^d}{V_a} + \frac{1}{2mgV_a} \left( (k_T - k_D)\tilde{E}_K + (k_T + k_D)\tilde{E}_P \right),
\]

where

\[
\tilde{E}_K \triangleq \frac{1}{2}m \left( (V_a^d)^2 - V_a^2 \right)
\]
\[
\tilde{E}_P \triangleq mg(h^d - h).
\]

This controller has three distinct behaviors depending on the relative magnitude of \( k_T \) and \( k_D \). If \( 0 < k_D < k_T \) then

\[
\sin(\gamma^c) = \frac{\dot{h}^d}{V_a} + \frac{1}{2mgV_a} \left( k_1\tilde{E}_K + k_2\tilde{E}_P \right),
\]

where \( k_1 \triangleq |k_T - k_D| \) and \( k_2 \triangleq k_T + k_D \). If \( \dot{V}_a^d > 0 \) then \( \gamma^c \) will increase which is the exact opposite of the desired behavior. If \( 0 < k_D = k_T \) then

\[
\sin(\gamma^c) = \frac{\dot{h}^d}{V_a} + \frac{k_2\dot{h}}{V_a}.
\]

In this case, \( \gamma^c \) only depends on the desired climb rate and the altitude error and cannot take into account the coupling between the altitude and airspeed. Finally, if \( 0 < k_T < k_D \), then

\[
\sin(\gamma^c) = \frac{\dot{h}^d}{V_a} + \frac{1}{2mgV_a} \left( -k_1\tilde{E}_K + k_2\tilde{E}_P \right).
\]

This case gives us the desired behavior because \( \gamma^c \) will increase if \( \dot{V}_a^d < 0 \) or \( \dot{h}^c > 0 \).
It is interesting to note that while the nonlinear thrust controller is very similar to the original TECS thrust controller, the flight path angle controllers have significant differences. First, the nonlinear controller uses the desired climb rate while the original controller uses the energy distribution rate. Furthermore, the nonlinear controller uses a weighted energy difference instead of the true energy difference. While using a weighted difference adds support to the modifications made in (6.11), the nonlinear controller has restrictions on the weighting. First, it is impossible to force the nonlinear controller to use the true energy difference because it would require $k_D = 0$. In addition, the potential energy error, $k_2 \in (k_T, 2k_T)$, is always weighted heavier than the kinetic energy error, $k_1 \in (0, k_T)$.

Unlike the original TECS controller, the nonlinear controller requires a desired airspeed, acceleration, altitude and climb rate. There are two different ways that the commands can be generated. The first approach is by using a reference model for the thrust and altitude where the desired climb rate and acceleration are generated by

$$\dot{h}^d = k_h(h^c - h) \quad (6.24)$$
$$\dot{V}_a^d = k_v(V^c_a - V_a) \quad (6.25)$$

The second approach uses the current state instead of the desired state to generate the desired rates which are computed by

$$\dot{h}^d = k_h(h^c - h) \quad (6.26)$$
$$\dot{V}_a^d = k_v(V^c_a - V_a) \quad (6.27)$$

In both approaches the desired altitude and airspeed are computed by

$$h^d = h^d(t_0) + \int_{t_0}^{t} \dot{h}^d(\tau) \delta \tau$$
$$V_a^d = V_a^d(t_0) + \int_{t_0}^{t} \dot{V}_a^d(\tau) \delta \tau,$$

where the desired values are initialized as $h^d(t_0) = h(t_0)$ and $V_a^d(t_0) = V_a(t_0)$. Note that the desired acceleration and climb rate should have saturation limits.
One interesting feature of the second approach is that \( h^c \) does not need to equal \( h^d \) in steady state. However, we have noticed, in simulation, that the interaction between the changing desired values and parameter estimates can cause the nonlinear adaptive TECS controller to go unstable. This issue does not occur when the adaptive component is not used, and the second approach removes the steady state error for a step input. For example, Figure 6.1a shows the velocity response to a 3 m/s step command where the desired acceleration is generated by (6.25), where \( k_v = 0.2 \), and is saturated by \( \pm 0.75 \text{ m/s}^2 \). The aircraft is being controlled by the nonlinear TECS controller but the adaptive element is not being used. Note that the desired airspeed is a smoothed version of the step command and there is steady state error due to modeling errors. Figure 6.1b shows the same situation but the desired acceleration is being generated by (6.27). In this case, the desired airspeed increases until the error between the commanded and actual airspeed goes to 0. We will show that this is true in general for a linearized version of the system.

Consider the system shown in Figure 6.2 where the dynamics are [16]

\[
\dot{h} = V_a \sin(\gamma) \\
\dot{V}_a = -g \sin(\gamma) - D + T,
\]
The drag is given by

\[ D = \frac{1}{2} \rho S V_a^2 C_{D_0}, \quad (6.28) \]

where \( C_{D_0} = 0.0439 \), which assumes a constant angle of attack, the controller is (6.16) with \( \phi = 0 \) and (6.20), and the guidance block can have feedback, given by (6.26) and (6.27), or no feedback, given by (6.24) and (6.25). Note that this ignores the pitch and thrust dynamics.

These systems, both with and without feedback, were linearized about the operating point \( \dot{h} = 0 \) and \( V_a = 13 \) m/s with \( k_T = 0.2 \), \( k_D = 0.25 \), \( k_h = 0.2 \), and \( k_v = 0.2 \) providing the linear system

\[
\dot{x} = Ax + Bu \\
\dot{y} = Cx.
\]

Both systems are stable and the eigenvalues of the state space matrix \( A \) are

\[
\text{eig}(A_{\text{noFeedback}}) = [-0.2187 \pm j0.05, -0.2, -0.2] \\
\text{eig}(A_{\text{feedback}}) = [-0.38, -0.25, -0.2 - 0.17].
\]

If we define the error as

\[
E(s) \triangleq R(s) - Y(s) = R(s) - C(sI_4 - A)^{-1}BR(s),
\]
where \( R(s) = 1/s \) for a step input, then \( H(s) \triangleq sE(s) \) is

\[
H_{\text{noFeedback}}(s) = \begin{bmatrix}
\frac{s(s^2+0.44s+0.05)}{d_{nf}(s)} & \frac{-0.004s+0.001}{d_{nf}(s)} \\
\frac{-s(0.08s+0.02)}{d_{nf}(s)} & \frac{s^3+0.33s^2+0.06s+0.008}{d_{nf}(s)} \\
\frac{s(s^3+0.75s^2+0.13s+0.002)}{d_f(s)} & \frac{s(-0.004s+0.001)}{d_f(s)} \\
\frac{s^2(-0.003s+0.001)}{d_f(s)} & \frac{s(s^3+0.64s^2+0.14s-0.01)}{d_f(s)}
\end{bmatrix}
\]

\[
H_{\text{feedback}}(s) = \begin{bmatrix}
\frac{d_f(s)}{d_{nf}(s)} & \frac{d_f(s)}{d_{nf}(s)} \\
\frac{d_f(s)}{d_{nf}(s)} & \frac{d_f(s)}{d_{nf}(s)}
\end{bmatrix}
\]

where

\[
d_{nf}(s) = s^3 + 0.64s^2 + 0.14s + 0.002
\]
\[
d_f(s) = s^4 + 0.95s^3 + 0.28s^2 + 0.03s + 0.0003.
\]

Note that the rows, of these matrices, correspond to the outputs \( h \) and \( V_a \) and the columns correspond to the inputs \( h^c \) and \( V_a^c \). Applying the final value theorem, which states [39]

\[
\lim_{t \to \infty} e(t) = \lim_{s \to 0} sE(s)
\]

if the system is stable, by taking the limit of \( H(s) \) as \( s \to 0 \) shows that the no feedback case has a steady state error in both altitude and airspeed due to the airspeed command and that adding feedback removes the steady state error completely.

### 6.4 Pitch and Thrust Control

Pitch and thrust control schemes are needed in order to implement the nonlinear TECS controller. In this chapter the pitch is controlled by a PID loop

\[
\delta_e = k_{p,\theta}(\theta^c - \theta) - k_{d,\theta}q + k_{i,\theta} \int_{t_0}^{t} (\theta^c - \theta) \delta\tau. \tag{6.29}
\]

The thrust is modeled by [16]

\[
T = \frac{1}{2} \rho S_{\text{prop}} C_{\text{prop}} \left( k_{\text{motor}}^2 \delta^2 - V_a^2 \right),
\]

128
where the parameters will be discussed in Section 6.7. This model can be simplified by combining terms to get

\[ T = k_{T,1}\delta_t^2 - k_{T,2}V_a^2 m \]

which means the commanded thrust can be converted to a throttle command by

\[ \delta_t = \sqrt{\frac{T^c + \hat{k}_{T,2}V_a^2}{k_{T,1}}} \quad (6.30) \]

where \( \hat{k}_{T,1} \) and \( \hat{k}_{T,2} \) are the estimates of the thrust parameters. Methods to estimate these parameters will be described later. The overall control architecture is shown in Figure 6.3 where the dashed blue line indicates the optional feedback in the guidance system.

One downside to this thrust controller is the lack of feedback. This is due to the lack of an accurate method to measure the thrust on small UAVs. Several different methods were explored to dynamically adapt these parameters based on the estimated drag and thrust and the measured acceleration. However, none of these methods worked well due either to the inaccuracies in the constant estimated drag parameters for the non-adaptive TECS controller or the dynamics of the estimated drag parameters for the adaptive TECS controller. However, it turns out that adding the feedback to the guidance block also corrects for inaccuracies in the thrust parameters. To show this we will perform a similar linearization analysis.
Consider the system

\[
\begin{align*}
\dot{h} &= V_a \sin(\gamma) \\
\dot{V}_a &= -g \sin(\gamma) - D + T \\
T &= k_{T,1} \delta_t^2 - k_{T,2} V_a^2,
\end{align*}
\]

where the drag and aerodynamic parameters are the same as the earlier example. Once again the estimated drag will be \( \hat{D} = 0 \). The flight path controller remains unchanged while the commanded thrust from (6.16) is converted to a throttle command using (6.30) where \( \hat{k}_{T,1} = 1.3k_{T,1}, \hat{k}_{T,2} = 0, k_{T,1} = 7.964, \) and \( k_{T,2} = 0.199 \).

Like the earlier example, both of the systems, with and without the guidance feedback, are stable. Computing \( E(s) \), applying a step input, and multiplying by \( s \) provides

\[
\begin{align*}
H_{\text{noFeedback}}(s) &= \begin{bmatrix}
\frac{s(s^2+0.95s+0.16)}{d_{nf}} & \frac{-0.001s+0.005}{d_{nf}} \\
\frac{-s(0.03s+0.006)}{d_{nf}} & \frac{s^3+0.91s^2+0.3s+0.03}{d_{nf}}
\end{bmatrix} \\
H_{\text{Feedback}}(s) &= \begin{bmatrix}
\frac{s(s^3+1.2s^2+0.22s+0.001)}{d_f} & \frac{s(0.001s-0.005)}{d_f} \\
\frac{-s^2(0.03s+0.007)}{d_f} & \frac{s(s^3+1.16s^2+0.36s+0.03)}{d_f}
\end{bmatrix},
\end{align*}
\]

where

\[
\begin{align*}
d_{nf} &= s^3 + 1.16s^2 + 0.236s + 0.03 \\
d_f &= s^4 + 1.4s^3 + 0.46s^2 + 0.05s + 0.0002.
\end{align*}
\]

Using the final value theorem we can conclude that adding feedback to the guidance block removes the steady state error even when no drag estimate is used and the thrust parameters are significantly off.

The nonlinear TECS controllers need to be able to estimate the angle of attack so that \( \gamma^c \) can be converted to a pitch command by \( \theta^c = \gamma^c + \alpha \). If the angle of attack is not measured or estimated then the pitch command can be approximated by \( \theta^c \approx \gamma^c + \alpha^t \), where \( \alpha^t \) is the trim angle of attack at the nominal airspeed. A single \( \alpha^t \) could be used or a
database of trim angles of attack for different airspeeds could be computed and used. This approximation reduces the overall accuracy of the nonlinear TECS control schemes.

6.5 Other Longitudinal Controllers

Three other altitude and airspeed controllers were implemented in order to provide a better comparison. The first controller consists of two PID loops on the total energy and the energy difference instead of the total energy rate and energy distribution rate that the original TECS controller uses. These control loops are given by

\[
T^c = T_D + k_{p,T} \dot{E}_T - k_{d,T} \ddot{E}_T + k_{i,T} \int_{t_0}^{t} \dot{E}_T \delta \tau
\]

\[
\theta^c = k_{p,\theta} \dot{E}_D - k_{d,\theta} \ddot{E}_D + k_{i,\theta} \int_{t_0}^{t} \dot{E}_D \delta \tau.
\]

The second controller is a standard decoupled successive loop closure control scheme where the thrust is controlled by a PI loop based on the airspeed error and the pitch is controlled by a PI loop based on the altitude error \[16\]. The commanded thrust and pitch are

\[
T^c = T_D + k_{p,T} \dot{V}_a + k_{i,T} \int_{t_0}^{t} \dot{V}_a \delta \tau
\]

\[
\theta^c = k_{p,\theta} \dot{h} + k_{i,\theta} \int_{t_0}^{t} \dot{h} \delta \tau.
\]

The final controller is a simplified version of the TECS controller used on the ArduPilot [92]. In this control scheme the thrust controller combines a feedforward term, given by

\[
T_{ff} = T_D + k_{T,ff} \dot{E}_t^d + k_{T,\phi} \left( \frac{1}{\cos(\phi)^2} - 1 \right),
\]

where \( k_{T,ff} > 0 \) controls the feedforward amount and \( k_{T,\phi} \) is a parameter that accounts for the increased drag while the plane banks, with a PID control loop based on the total energy error given by

\[
T = T_{ff} + k_{p,T} \dot{E}_t + k_{d,T} \ddot{E}_t + k_{i,T} \int_{t_0}^{t} \dot{E}_t \delta \tau.
\]

131
The pitch controller uses a scaled version of the energy difference given by

\[ E_3 \triangleq (2 - k_{ke})gh - \frac{k_{ke}}{2}V^2_a, \]

where \( k_{ke} \in [0, 2] \) determines the weighting between the altitude and airspeed errors. A \( k_{ke} = 0 \) means that only the altitude error is used and \( k_{ke} = 2 \) means that only the airspeed error is used. Note that this scaled energy error is similar to the scaled energy error used in the nonlinear TECS controller and the most recent version of Lambregts' TECS controller [68].

The commanded pitch is generated by a PID loop with a feedforward term and is given by

\[ \theta^c = \frac{1}{V_a} \left( k_{p,\theta} \tilde{E}_3 + \frac{\dot{E}_3}{g} + k_{d,\theta} \dot{E}_e + k_{i,\theta} \int_{t_0}^{t} \tilde{E}_3 \delta \tau \right). \]

### 6.6 Lateral Control

A simple lateral control scheme was implemented based on the approach in [16] and a brief summary is presented here. The objective of the lateral controller is to track a commanded course, \( \chi^c \), which is the actual direction the aircraft is moving after taking into account the wind. Given the course error, a roll command is generated by

\[ \phi^c = k_{p,\chi} \tilde{\chi} - k_{d,\chi} r + k_{i,\chi} \int_{t_0}^{t} \tilde{\chi} \delta \tau, \]  

(6.31)

which will attempt to drive the course error to zero. The roll error is used to generate the aileron command by

\[ \delta_a = k_{p,\phi} \tilde{\phi} - k_{d,\phi} p + k_{i,\phi} \int_{t_0}^{t} \tilde{\phi} \delta \tau. \]  

(6.32)

Note that (6.31) is saturated by ±\( \tilde{\phi} \) to limit the commanded roll and (6.32) is saturated by ±1.

The course command is generated by a vector field approach that drives the airplane to the straight line between two waypoints. Dubin’s paths are used to smooth the transition between waypoints. See [9, 16] for details about the path following and waypoint management methods.
6.7 Dynamics

These control schemes were tested on a Zagi airframe simulator from [16] instead of the V-Bat simulator. The kinematic equations are given by (3.1)-(3.4) where the net force is given by

\[ \mathbf{f} = \mathbf{f}_g + \mathbf{f}_a + \mathbf{f}_t, \]

and the net moment is

\[ \mathbf{m} = \frac{1}{2} \rho S V_a^2 \left[ \begin{array}{c} b \left( C_{l\beta} \beta + \frac{b}{2V_a} (C_{l\alpha} p + C_{l\alpha} r) + C_{L\alpha} \delta_a + C_{L\epsilon} \delta_r \right) \\ c \left( C_{m\alpha} + C_{m\alpha} \alpha + C_{m\alpha} \frac{V}{2V_a} q + C_{m\delta_e} \delta_e \right) \\ b \left( C_{n\beta} + \frac{b}{2V_a} (C_{n\alpha} p + C_{n\alpha} r) + C_{n\alpha} \delta_a + C_{n\delta_r} \delta_r \right) \end{array} \right]. \]

The force of gravity is given by (3.5), the aerodynamic force is

\[ \mathbf{f}_a = \frac{1}{2} \rho S V_a^2 \left[ \begin{array}{c} -C_D \cos(\alpha) + C_L \sin(\alpha) + C_L \delta_e \sin(\alpha) \delta_e \\ C_{Y\beta} \beta + \frac{b}{2V_a} C_{Y\alpha} p + C_{Y\alpha} \delta_r \\ -C_D \sin(\alpha) - C_L \cos(\alpha) - C_L \delta_e \cos(\alpha) \delta_e \end{array} \right], \]

and the thrust from the motor is

\[ \mathbf{f}_t = \left[ \begin{array}{c} \frac{1}{2} \rho S_{\text{prop}} C_{\text{prop}} (l_{\text{motor}}^2 - V_a^2) \\ 0 \\ 0 \end{array} \right]. \]

The lift coefficient is

\[ \mathbf{C}_L = (1 - \sigma(\alpha))(C_{L0} + C_{L\alpha} \alpha) + \sigma(\alpha) \left(2 \text{sign}(\alpha) \sin^2(\alpha) \cos(\alpha)\right), \]

where

\[ \sigma(\alpha) = \frac{1 + \exp^{-M(\alpha-\alpha_0)} \exp^{M(\alpha+\alpha_0)}}{(\exp^{-M(\alpha-\alpha_0)})(\exp^{M(\alpha+\alpha_0)})}. \]
Table 6.1: Airframe parameters [16]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{L_0}$</td>
<td>0.028</td>
<td>$C_{Y_3r}$</td>
<td>0.17</td>
<td>m</td>
<td>1.56 kg</td>
</tr>
<tr>
<td>$C_{L_{a}}$</td>
<td>3.45</td>
<td>$C_{l_{y}}$</td>
<td>0.12</td>
<td>$S$</td>
<td>0.2589 m²</td>
</tr>
<tr>
<td>$C_{L_{b}}$</td>
<td>0.36</td>
<td>$C_{l_{p}}$</td>
<td>-0.26</td>
<td>b</td>
<td>1.4224 m</td>
</tr>
<tr>
<td>M</td>
<td>50</td>
<td>$C_{l_{r}}$</td>
<td>0.14</td>
<td>c</td>
<td>0.3302 m</td>
</tr>
<tr>
<td>$\alpha_0$</td>
<td>0.4712</td>
<td>$C_{l_{y}}$</td>
<td>0.08</td>
<td>$C_{\text{prop}}$</td>
<td>1 m</td>
</tr>
<tr>
<td>$C_{D_0}$</td>
<td>0.03</td>
<td>$C_{l_{y}}$</td>
<td>0.105</td>
<td>$S_{\text{prop}}$</td>
<td>0.0314 m²</td>
</tr>
<tr>
<td>e</td>
<td>0.9</td>
<td>$C_{n_{b}}$</td>
<td>0.25</td>
<td>$J_x$</td>
<td>0.1147 kg m²</td>
</tr>
<tr>
<td>$C_{m_{a}}$</td>
<td>-0.38</td>
<td>$C_{n_{p}}$</td>
<td>0.22</td>
<td>$J_y$</td>
<td>0.0576 kg m²</td>
</tr>
<tr>
<td>$C_{m_{q}}$</td>
<td>-3.6</td>
<td>$C_{n_{r}}$</td>
<td>-0.35</td>
<td>$J_z$</td>
<td>0.1712 kg m²</td>
</tr>
<tr>
<td>$C_{m_{s_{r}}}$</td>
<td>-0.5</td>
<td>$C_{n_{s_{r}}}$</td>
<td>-0.032</td>
<td>$J_{xx}$</td>
<td>0.0015 kg m²</td>
</tr>
<tr>
<td>$C_{Y_{y}}$</td>
<td>-0.98</td>
<td>$C_{n_{s_{a}}}$</td>
<td>0.06</td>
<td>$k_{\text{motor}}$</td>
<td>20</td>
</tr>
<tr>
<td>$C_{Y_{p}}$</td>
<td>-0.26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

and $M$ and $\alpha_0$ are positive constants. The drag coefficient is

$$C_D = C_{D_0} + \frac{(C_{L_0} + C_{L_{a}} \alpha)^2}{\pi e AR},$$

(6.33)

where $e$ is the Oswald efficiency factor and $AR = \frac{b^2}{S}$ is the aspect ratio of the wing. The airframe parameters, used in the simulation, are in Table 6.1.

### 6.8 Parameter Estimation

Drag and thrust estimates are needed in order to implement the nonlinear TECS controller with the open loop thrust controller (6.30). While these estimates could be generated using the model developed in Chapter 3, we will develop a simple parameter estimation scheme that only requires data from several flight tests and possibly a static bench test.

The lift, drag and thrust can be modeled by [16]

$$F_L = \frac{1}{2} \rho S V_a^2 (C_{L_0} + C_{L_{a}} \alpha)$$

$$F_D = \frac{1}{2} \rho S V_a^2 (C_{D_0} + C_{D_{a}} \alpha)$$

$$T = \frac{1}{2} \rho S_{\text{prop}} C_{\text{prop}} \left( k_{\text{motor}}^2 s_t^2 - V_a^2 \right),$$

134
Note that the lift model is included because it is needed by the estimation algorithm. By combining terms these models can be simplified to

\[ F_L = k_5 V_a^2 + k_6 V_a^2 \alpha \]  \hspace{1cm} (6.34)

\[ F_D = k_1 V_a^2 + k_2 V_a^2 \alpha \]  \hspace{1cm} (6.35)

\[ T = k_3 \delta_t^2 - k_4 V_a^2 \]  \hspace{1cm} (6.36)

In steady level trim flight

\[ F_L - mg = T \sin(\theta) \]  \hspace{1cm} (6.37)

\[ F_D = T \cos(\theta). \]  \hspace{1cm} (6.38)

Inserting (6.34)-(6.36) into (6.37) and (6.38) provides

\[ k_5 V_a^2 + k_6 V_a^2 - mg = (k_3 \delta_t^2 - k_4 V_a^2) \sin(\theta) \]  \hspace{1cm} (6.39)

\[ k_1 V_a^2 + k_2 V_a^2 \alpha = (k_3 \delta_t^2 - k_4 V_a^2) \cos(\theta). \]  \hspace{1cm} (6.40)

By noting that \( \dot{h} = 0 \) implies \( \gamma = 0 \) and \( \alpha = \theta \), the lift, drag, and thrust parameters can be estimated by collecting flight data for several different constant altitude trim conditions and solving the system of equations. If least squares is used then flight data from three different trim conditions are needed to provide estimates for all six parameters.

However, the TECS controller only needs the drag and thrust parameters. The parameter estimation problem can be simplified by only looking at (6.40). If \( \theta \) is small then (6.40) can be approximated as

\[ k_1 V_a^2 + k_2 V_a^2 \alpha = k_3 \delta_t^2 - k_4 V_a^2. \]  \hspace{1cm} (6.41)

There is no way to determine the overall scale of the thrust and the drag from (6.41). The scale can be obtained by keeping the \( \cos(\theta) \) term, however, the small changes in \( \theta \) between the trim conditions are easily lost in noise. The scale ambiguity can be solved by performing some static bench tests in order to determine \( k_3 \). Another problem with (6.41) is that there
is no way to distinguish between the effects of $k_1 V_a^2$ of the drag and $k_4 V_a^2$ of the thrust. This can easily be corrected. Using these drag and thrust models, the throttle command is

$$
\delta_t^2 = \frac{1}{k_3} \left( k_1 V_a^2 + k_2 V_a^2 \alpha + \phi^\top \hat{\Psi} + \frac{\hat{E}_d}{V_a} + k_T \frac{\hat{E}_T}{V_a} + k_4 V_a^2 \right). \tag{6.42}
$$

The terms $k_1 V_a^2$ and $k_4 V_a^2$ can be combined into a single term $k_1' V_a^2$ where $k_1' \triangleq k_1 + k_4$. Using this, (6.41) becomes

$$
k_1' V_a^2 + k_2 V_a^2 \alpha = k_3 \delta_t^2. \tag{6.43}
$$

Adding the $\cos(\theta)$ back to (6.43), providing

$$
k_1' V_a^2 + k_2 V_a^2 \alpha = k_3 \delta_t^2 \cos(\theta), \tag{6.44}
$$

improves the accuracy slightly. Once again, $k_1'$ and $k_2$ can be estimated by flying the plane at several different trim conditions, measuring the airspeed, pitch, and throttle command, and then performing least squares.

The parameter estimation methods were tested in simulation. Nine simulated flights were performed where the aircraft was in a steady level flight at nine different airspeeds ranging from 11 m/s to 15 m/s in 0.5 m/s increments, and the estimated airspeed and pitch and the throttle command were recorded. To reduce the effect of the noisy state estimates, the data were averaged over a length of time. The aerodynamic parameters $k_1$ to $k_6$ were then estimated, using these data, by an unconstrained optimization using (6.39) and (6.40) and optimization using (6.39) and (6.40) where $k_3$ was constrained to be within 30% of the true value. The control parameters, $k_1'$, $k_2$, and $k_3$, were then computed from the aerodynamic parameters. In addition, the control parameters $k_1'$, $k_2$, and $k_3$ were directly estimated using an unconstrained optimization using (6.44) where $k_3$ was set to the true value and an unconstrained optimization where $k_3$ using (6.44) was set to be 130% of the true value. The
control parameter estimates were then compared to the true values computed by

\[ k'_{1} = \frac{1}{2} \rho S \left( C_{D0} + \frac{C_{L0}^{2}}{\pi e AR} \right) + \frac{1}{2} \rho S_{prop} C_{prop} \]  

(6.45)

\[ k_{2} = \frac{1}{2} \rho S \left( \frac{2C_{L0}C_{L_{\alpha}}}{\pi e AR} + \frac{C_{L_{\alpha}}^{2}}{\pi e AR} \right) \]  

(6.46)

\[ k_{3} = \frac{1}{2} \rho S_{prop} C_{prop} k_{motor}^{2} \]  

(6.47)

Note that first term in (6.45) and (6.46) come from the Taylor series expansion of (6.33) about the trim angle of attack \( \alpha^t \). All of the optimizations were performed using MATLAB’s fmincon function. Each test was repeated 50 times with each test using a different random seed for the sensor noise.

During these tests, the accuracy and consistency of each method were tested along with testing different airspeeds flown and the length of the data averaging window. Table 6.2 compares the methods when the full range of airspeeds were used and the data were averaged over thirty seconds of flight. The full, unconstrained optimization found a minimum that matched the data but obtained unrealistic parameters that would not work in a real flight. However, constraining the thrust parameter allowed the full optimization to find parameters that were relatively close to the true values. Similar parameter estimates could be found using the simplified optimization if the true thrust parameter was known. However, the simplified optimization was very sensitive to inaccuracies in the thrust parameter. It is interesting that the optimization algorithms were so consistent in the parameter estimation error. This suggests that there are additional factors that could be taken into account to improve the accuracy.

Table 6.3 explores how many trim conditions are needed when the full constrained optimization was used on data that were averaged over a thirty second window. The standard deviation of the parameter error increased as fewer airspeeds were included, however, the worst case and mean error decreased as the number of airspeeds included decreases. This is probably because the wider range of airspeeds had trim angle of attacks further from the value used to compute the true parameter. However, all of the airspeed ranges tested provided usable results.
Table 6.2: Mean, standard deviation, and maximum of the norm of the parameter estimation error. A 30 second data averaging window and all nine airspeeds were used.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean (%)</th>
<th>STD (%)</th>
<th>Max (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full - Unconstrained</td>
<td>172.6</td>
<td>1.55e-3</td>
<td>172.6</td>
</tr>
<tr>
<td>Full - Constrained Thrust</td>
<td>33.9</td>
<td>0.4</td>
<td>35.1</td>
</tr>
<tr>
<td>Simplified - True Thrust</td>
<td>34.0</td>
<td>0.4</td>
<td>35.3</td>
</tr>
<tr>
<td>Simplified - Thrust Error</td>
<td>45.4</td>
<td>0.3</td>
<td>46.3</td>
</tr>
</tbody>
</table>

Table 6.3: Mean, standard deviation, and maximum of the norm of the parameter estimation error. A 30 second data averaging window and the full constrained optimization were used.

<table>
<thead>
<tr>
<th>Airspeeds (m/s)</th>
<th>Parameter Estimation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>Increment</td>
</tr>
<tr>
<td>11</td>
<td>0.5</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>11.5</td>
<td>0.5</td>
</tr>
<tr>
<td>11.5</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0.5</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.4 compares different data averaging window lengths when the full constrained optimization was used and the data included all nine airspeeds flown. As expected, averaging the data over a longer window reduced the standard deviation and the maximum error. In our opinion a data window between five and ten seconds provides the needed accuracy.

Table 6.4: Mean, standard deviation, and maximum of the norm of the parameter estimation error. All nine airspeeds and the full constrained optimization were used.

<table>
<thead>
<tr>
<th>Data Averaging Window (s)</th>
<th>Parameter Estimation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean (%)</td>
</tr>
<tr>
<td>1</td>
<td>32.9</td>
</tr>
<tr>
<td>5</td>
<td>33.9</td>
</tr>
<tr>
<td>10</td>
<td>34.1</td>
</tr>
<tr>
<td>20</td>
<td>33.9</td>
</tr>
<tr>
<td>30</td>
<td>33.9</td>
</tr>
</tbody>
</table>
6.9 Simulation Results

The control algorithms were tested using an ideal and a realistic simulation. In the ideal simulation the pitch and thrust response are modeled by the second-order dynamic systems

\[
\ddot{\theta} = -2\zeta_\theta \omega_n \dot{\theta} + \omega_n \dot{\theta} (\theta^c - \theta)
\]

\[
\ddot{T} = -2\zeta_T \omega_n \dot{T} + \omega_n (T^c - T),
\]

where \(\zeta_\theta = \zeta_T = 0.707\) and \(\omega_n,\theta = \omega_n, T = 5\). In addition, this simulation assumes that the controller has perfect state knowledge, that the dynamics are restricted to two dimensions, and that the estimated drag is \(\hat{D} = 0.8D\) for both nonlinear TECS controllers. The realistic simulation uses an extended Kalman filter with simulated noisy sensors, see [16] for full details of the sensor models and the state estimation scheme, to generate the state estimates and the full three dimensional dynamics are used. In addition, (6.29) is used to control the pitch, the original and both nonlinear TECS controllers use the thrust controller given by (6.30), and a single \(\alpha^t\) for \(V_a = 13\) m/s is used because the estimation scheme does not estimate the angle of attack. The other control schemes’ thrust controllers are modified to output a throttle command instead of a thrust command. The nonlinear TECS controllers use the parameter estimates,

\[
\hat{k}_1' = 0.0248
\]

\[
\hat{k}_2 = 0.1131
\]

\[
\hat{k}_3 = 7.9614,
\]

obtained by the full constrained optimization where \(\hat{k}_3\) was constrained to be within 30% of the true value, the 12 m/s to 14 m/s in 0.5 m/s increments airspeeds were used, and the data was averaged for ten seconds. In every simulation the aircraft was initialized and commanded to fly at 13 m/s and \(h = 100\) m for 50 seconds. This allowed all of the controllers to settle down and the parameter estimates, for the adaptive TECS, to enter steady state.
Then altitude and/or velocity step commands were given and the simulation ran for 50 more seconds. Only the last 50 seconds of flight are displayed.

Figure 6.4a shows the ideal response for a positive altitude step command while the desired velocity is constant and demonstrates the main weakness of the traditional decoupled successive loop closure control approach. Notice that when the desired altitude increases, the airspeed for the decoupled controller immediately starts to decrease. This issue is also shown in Figures 6.4c and 6.4d.

The original TECS controller does not have this problem because it accounts for the coupling in the dynamics. However, it tends to have the largest overshoot and, as shown in Figures 6.4a and 6.4b, the state that is commanded to be held constant oscillates while the other state undergoes the step command. The oscillation and overshoot can be reduced by using the TECS PID based on the energy instead of the PI controllers based on the energy rate. The ArduPilot TECS also handles these cases very well with only a slight altitude gain while decreasing the airspeed.

Both nonlinear TECS controllers are able to follow the commands very well and are able to achieve the fastest rise time with little to no overshoot. The adaptive TECS controller is able to correct for its incorrect drag model but it has a minor oscillation in its airspeed response. However, this oscillation is significantly less than the original TECS controller’s oscillation. The oscillation can be completely removed by using the nonlinear controller without the adaptive element but with the guidance feedback.

The results of the realistic simulations for the exact same scenarios are shown in Figure 6.5. There are several interesting differences between the ideal and the realistic simulations. First, the original TECS controller’s altitude response always has several meters of error in steady state which appears at the beginning and end of the simulations. This error is caused by the inaccurate thrust control scheme being used. Second, the adaptive TECS has steady state error in the airspeed and altitude when the aircraft is not traveling at 13 m/s. This steady state error could be removed, or reduced, by measuring the angle of attack or by using several different trim angles of attack. It is interesting that the non-adaptive TECS controller is able to overcome incorrect trim angle of attack and still achieve very nice performance in each case. The ArduPilot TECS control scheme is still able to achieve decent
Figure 6.4: Ideal simulation results

performance, however, its rise time had to be significantly reduced due to its sensitivity to the noisy measurements. Finally, the TECS PID controller has a similar rise time as the original TECS control scheme yet it has a significantly smaller error.

Figure 6.6 shows two realistic simulations where the aircraft changes its heading by 180°. As expected the aircraft loses altitude during the turn with every control method. It is interesting that the ArduPilot has the largest variation in altitude and airspeed in Figure 6.6a even with the roll feedforward term. The successive loop closure loses a significant amount of airspeed in Figure 6.6b where the aircraft is climbing as well as turning. The nonlinear TECS controller has the best response in both tests.
6.10 Conclusions

In this chapter we derived a nonlinear altitude and airspeed controller for level flight based on TECS principles. The adaptive variant of this controller was shown to be able to track a desired airspeed and altitude using Lyapunov stability arguments and the non-adaptive variant with guidance feedback was shown to have zero steady state error for step inputs based on linearizing the system about a trim condition and applying the final value theorem. These nonlinear TECS controllers were compared with the original TECS control scheme as well as the standard successive loop closure, TECS PID, and the TECS controller used on the ArduPilot.
Simulation results, for the ideal system with perfect state knowledge and perfect thrust and pitch controllers, show that both nonlinear TECS variants handle the coupling between the altitude and airspeed dynamics and have a better response than the original TECS control scheme as well as the other control schemes. The realistic simulations, with noisy state estimates and realistic pitch and thrust controllers, show that even with inexact aerodynamic parameter estimates and an open loop thrust control scheme the nonlinear TECS controller has better performance than traditional control approaches. It should be pointed out that the realistic simulations assumed low-cost noisy sensors and no aerodynamic model of the aircraft. Even with these assumptions the nonlinear TECS controller performed admirably. This control scheme’s performance will only improve with better sensors, state estimation especially the angle of attack, a more exact drag model, and a thrust controller with feedback.

In addition, we presented a simple scheme to estimate the needed aerodynamic parameters using only some flight data and a simple bench test. These parameter estimates were used in the simulations which showed the method is accurate enough to generate useable parameters.

There are several aspects of this research that can be continued. First, this control scheme has only been tested for small, cheap UAVs. It would be interesting to see how
it performs on larger aircraft. Second, more advanced estimation schemes might be able to perform online estimation of the thrust and drag parameters based on the measured acceleration. This would improve the performance of the adaptive TECS controller. Finally, we would like to obtain stronger theoretical guarantees on the nonlinear TECS controller.

The feedback in the guidance system is similar to what is done in the observer-like model reference adaptive control (OL-MRAC) scheme [69]. It would be interesting to see if any of the techniques developed for the OL-MRAC could be applied to the nonlinear TECS control problem. This might provide additional insight leading to stronger stability and tracking guarantees.
7.1 Introduction

The transitions between horizontal and vertical orientations are a unique aspect of tailsitter, or ducted fan aircraft, control design. Obviously, the objective is to transition between flight modes, but there are many other questions that must be answered. For instance, should the transition be as fast as possible? Do we care about the distance the aircraft travels during the transition? What about the altitude variation? The desired behavior of the transition depends on the answers to these, and many more, questions.

In general, the transition control scheme consists of two parts. The first component is the trajectory, or the desired states during the transition, which depends on the transition objectives. The second component is the controller or how the aircraft will follow the transition trajectory. While both components are intricately linked, each will be discussed separately.

It is difficult to directly compare the transition results in the literature because of the wide variety in the airframes, including significant variation in size, weight, control authority, control surfaces, and thrust-to-weight ratio, used for the development and testing. For example, Green and Oh used a simple PD attitude controller with a step command on pitch to transition a small, light-weight acrobatic airplane, with very large control surfaces, with almost no altitude deviation \[40, 41\]. However, this type of transition controller will not work well in general for large aircraft or aircraft with smaller control surfaces. Kubo et al., as another example, used an aircraft which had adjustable flaps and slats which allowed it to perform a faster transition with less altitude variation \[63\]. Finally, Maqsood and Go used an aircraft with a variable-incidence wing and concluded that the variable-incidence wings significantly decreased the thrust-to-weight ratio needed to perform a constant altitude
transition with a specific time duration [71–73]. However, general conclusions can be formed. For example, the shorter the HTL transition time, the higher the thrust-to-weight ratio needs to be [72].

7.1.1 Trajectory Design

The first aspect that must be determined is the transition trajectory, or, in other words, the desired states throughout the transition. Determining the transition trajectory has several components including which states to control and how to compute the desired states.

There are four common themes for the trajectory objective. First, the most conservative transition approach is the continuous ascent (CA) approach where the aircraft has a positive flight path angle throughout the entire transition. In other words, the aircraft is always climbing during the transition. If desired, a CA trajectory can keep the angle of attack low during the entire transition which keeps the aircraft outside the highly nonlinear stall region. For example, Jung et al. developed a CA trajectory which consists of time-indexed angle of attack and flight path angle commands [52–54]. The HTL transition is divided into three segments. First, $\gamma^c$ is reduced while $\alpha^c = 0$, then $\gamma^c$ is rapidly decreased to 0° while $\alpha^c$ is increased to a value below the cruise trim angle of attack. Finally, $\alpha^c$ is increased to the cruise trim value while $\gamma^c = 0$. The LTH trajectory is generated in a similar manner. The length of the segments and the rate of change in the commands are selected based on “physical intuition combined with trial and error” [53]. While this trajectory successfully keeps the angle of attack low throughout the entire transition, thus avoiding the stall region, it requires that the angle of attack and flight path angle be estimated. Jung et al. also explore a similar CA trajectory which consists of velocity and flight path angle commands [54]. However, they do not discuss how the trajectory is generated. In any case, this alternative approach still requires that either $\alpha$ or $\gamma$ be estimated. A simpler CA type HTL trajectory is discussed in [11] where the throttle is set to 100% and the pitch command decreases as the airspeed increases.

CA trajectories are very conservative because they attempt to avoid the stall region. However, they accomplish this by gaining a significant amount of altitude. For example,
flight tests with Jung’s approach caused approximately 50 m of altitude gain during an HTL transition and approximately 30 m of altitude gain during an LTH transition [52]. This altitude gain may not be a problem, in fact it might be desired, for the HTL transition. However, a large altitude gain is typically not desired during an LTH transition because it takes a long time to descend in hover flight.

Another common objective is to maintain a constant altitude, or minimize the altitude variation, throughout the entire transition [56,63]. In our opinion, this approach is preferred for LTH transitions because it reduces the time needed to land. However, this approach typically requires that the aircraft enter the stall region or have additional control surfaces such as flaps or variable-incidence wings [63,71–73].

A few trajectory approaches attempt to specify the position and altitude where the transition will end. For example, Osborne generates the horizontal and vertical components of his trajectory separately. The horizontal trajectory is generated by commanding the constant acceleration that will cause the aircraft to arrive at the final position at the desired velocity, and the velocity and position commands are generated by integrating the commanded acceleration. The vertical trajectory is a smooth shift between the initial and final altitudes generated by several sigmoid functions [89]. However, care must be taken when generating these types of trajectories to ensure that they are feasible. In addition, wind can easily interfere with these types of transitions if it is not measured.

Finally, many trajectories do not have any objective other than successfully performing the transition. These types of trajectories do not care about how much altitude is gained or how long it takes. In addition, they tend to be the easiest to implement and do not require the same knowledge of the aircraft’s aerodynamics required by the other methods. However, these trajectories tend to have the worst performance. Some examples are in [58,104,106].

There are many different ways to generate the trajectories once the objectives have been determined. The simplest trajectories to implement are a simple step or ramp commands. For example, Johnson et al.’s trajectory is a velocity and pitch ramp command [51] while [41,88,104,106] use a step pitch command. A pitch step command can be sent through a reference model based on the aircraft’s attitude dynamics [58,89] or a second-order system [74] to obtain achievable attitude commands. The step pitch and velocity based tra-
jectories typically cause significant altitude variation during the transition. In addition, the transitions can be very abrupt and the step commands cannot be tuned. This type of trajectory generation typically uses the last trajectory objective.

Another trajectory generation approach is to generate smooth curves through trial and error. For example, the HTL trajectory, in [26], generates velocity and pitch commands that are smooth functions of time where the velocity increases and the pitch decreases at rates dependent on several user-provided parameters. The LTH trajectory is generated by the same functions but it consists of pitching up at a constant rate and the velocity command only starts to decrease when the pitch command is vertical. Kriel’s thrust and pitch commands are almost ramps and the angle of attack command initially decreases to $-7^\circ$ then smoothly increases to $7^\circ$ for the HTL trajectory and increases to $12^\circ$ and then down to $0^\circ$ for the LTH trajectory [61]. Jung’s approach, discussed earlier, is another example of this approach. Most of the smooth curve trajectories are CA and will gain a significant amount of altitude.

Optimization algorithms can generate a wide range of trajectories assuming a high-fidelity aerodynamic model of the aircraft is available. Several of the optimized trajectories are CA type. For example, Stone and Clarke generate optimal HTL trajectories where the objective is to minimize the transition time while ensuring that the aircraft climbs no more than 100 ft, that the final flight path angle is positive, that the magnitude of the angle of attack throughout the transition is less than $12^\circ$, and that the aircraft does not descend at all [101].

Most optimal trajectories attempt to minimize the altitude variation along with other metrics. For example, Kubo et al. develop optimal trajectories that minimize the weighted sum of the transition time, the total altitude variation, and the rate of change of the actuator commands while ensuring that the angle of attack does not enter the stall region [63]. Maqsood and Go minimize the weighted sum of the exit velocity error and total altitude variation, while constraining the pitch rate and thrust rate of change [71–73]. Kita et al. find the HTL trajectory that maximizes the horizontal force while keeping the vertical force 0 thus finding the minimal time transition while restricting the altitude change [56]. For
the LTH transition, they relax the vertical force constraint so the vertical force is bounded where the bounds are found through trial and error.

Other optimal trajectories do not fit within the four common themes discussed earlier. For example, Naldi and Marconi explore two different optimal trajectories [85]. The first trajectory minimizes the amount of time it takes to perform the transition and the second trajectory minimizes the overall energy used by the propeller during the maneuver. In addition, both optimizations include constraints that ensure the aerodynamic drag does not exceed the maximum thrust and that the control surfaces are large enough to counteract the aerodynamic moments.

One particularly impressive result is in [32] where Cory presents several methods that allow an unpowered glider to go near vertical and perch on a landing platform. In one approach the desired perching trajectory is generated using standard nonlinear optimization techniques which provides an open-loop controller. Then a time varying LQR (TVLQR) controller is created around the optimized trajectory. However, this system is sensitive to the initial velocity. The larger the difference between the actual and initial velocity the larger the final position error becomes. They solve this issue by using an approach called LQR-Trees which, at a basic level, builds up a library of trajectories and TVLQR controllers. Which trajectory and controller are used depends on the initial conditions of the aircraft.

Cory’s method demonstrates the main limitation of precomputing a single desired trajectory. If the initial conditions are not close to what is assumed then the performance will degrade and potentially cause the aircraft to go unstable. In addition, if the aircraft model is wrong then the generated trajectory may not be feasible.

Time-indexed commands can be generated for any state by simulating the transition. In addition, feedforward throttle and actuator commands can be generated the same way. Feedforward commands can also be generated through the trajectory generation algorithms, particularly the optimization approaches. An example of this is [61].

7.1.2 Control Design

The second aspect is the transition controller. As mentioned in Chapter 1, there are several common tailsitter control architectures that impact how the transition controller is
designed. First, the transition can be controlled through the hover or level flight controller. Second, the transition, as well as the other flight modes, can be controlled by a unified control scheme. Finally, the transition can have a dedicated controller. Each of these approaches, and the various control schemes that have been developed, will be discussed in this section.

The first approach is to use the hover or level flight controllers. For example, Myrand-Lapierre et al. initially use their level controller for the LTH transition and their hover controller for the HTL transition [82]. The active controller switches when the pitch passes $\theta = 50^\circ$. Frank et al. use their position LQR hover controller to perform the HTL transition [38]. However, many of the hover and level controllers are not suited for the flight conditions that might be encountered. Care must be taken with this approach because most hover or level controllers cannot handle the entire transition region. The active controller can switch in the middle of the transition, like Myrand-Lapierre’s approach, but care must be taken that the switch is well behaved.

Johnson et al. use an unified adaptive control scheme which consists of dynamic inversion generating the estimated commands, a PD controller to reduce tracking errors, a neural network that estimates the model errors, and psuedocontrol hedging to reduce the effect of actuator saturation on the estimated parameters [49–51]. Using only a simple linear model of the aircraft in hover, Johnson’s controller is able to control the aircraft in level, hover, and the transitions. However, they were never able to achieve stationary hover in flight tests.

Dynamic inversion has been used in other transition control schemes. Jung and Schim use dynamic inversion to generate the thrust and attitude commands and the attitude command is tracked by an L1 adaptive controller [52–54]. Naldi and Marconi use system inversion to compute the desired thrust and pitch based on the commanded velocity and flight path angle [84]. Model inaccuracies and disturbances are compensated for by PI loops based on the velocity and flight path angle error to the desired thrust and pitch respectively.

Other nonlinear transition controllers have been developed. For example, Osborne developed a feedback linearization controller based on the position and altitude dynamics and an MRAC controller which used second-order systems for the altitude and position reference models [89]. Sobolic also developed a backstepping position controller which can guarantee
stability based on Lyapunov arguments [98]. In general, the nonlinear controllers are difficult to tune and many, such as the feedback linearization or dynamic inversion controllers, require an accurate aircraft model.

Linear transition control approaches have not been explored as much as nonlinear control methods. Frank et al. used their relative position LQR controllers developed for hover flight to perform the HTL transition. Jeong et al. developed an LQTI velocity controller for a shrouded-fan aircraft that generates four different linear models at the trim conditions for \( v = 0, 5, 10, 15 \text{ m/s} \) [46]. They present simulation results for flights at three of these flight conditions, however, they do not discuss or present any results for switching, or interpolating, between the linear models. In other words, they do not show that this method can be used to transition the aircraft. Kriel’s LQR controller creates a new linear model every twenty time steps which is used to create new LQR gains allowing the controller to adjust to the changing dynamics [61]. In general, the linear transition controllers require an accurate model of the aircraft and do not handle the nonlinearities in the dynamics as well as the nonlinear controllers unless multiple linear models are used.

Rarely, open-loop control schemes are used. For example, Frank et al. generate commands for the elevator and rudder, during an LTH transition, based on the aircraft model, and the throttle is set such that it is slightly under what is needed to counteract gravity in hover [38]. Open-loop controllers can work indoors with no disturbances, as shown by Frank, but are not robust in general.

Many of the nonlinear and LQR transition controllers generate the throttle command in addition to the actuator or attitude commands. A wide range of throttle controllers have been developed for the other control schemes. Stone used two different controllers. The first selects the throttle with an open-loop pitch based throttle schedule, and the second is a velocity controller that tracks a pitch based velocity schedule [104,106]. One of Osborne’s controllers generates a thrust command using a altitude error PID loop [89]. Sobolic’s thrust controller is also a PID altitude controller with a feed-forward weight term [98]. However, if the pitch is low enough (\( \theta < 30 \)) the feed-forward weight term is removed and a component based on the horizontal velocity error is added. Marchini controls the throttle with a climb rate PID loop, however, the output of the throttle loop is overridden if the attitude error
is greater than a constant bound $[74]$. Michini uses a PI controller that tracks the desired $\dot{\beta}$ velocity $[79]$. Another approach is to use a constant throttle during the transition as in $[11, 101]$.

Most of these throttle controllers do not take into account the effect that pitch has on the thrust response. For example, using the throttle to track the altitude or climb rate commands works well when the aircraft is in near hover flight but it breaks down in level flight. This typically is not noticeable for fast transitions, especially the HTL transitions, but will significantly degrade the performance of slower transitions.

### 7.1.3 Discussion

In our opinion the transitions should attempt to maintain a constant altitude throughout the transition which eliminates most of the transition trajectory methods. The remaining approaches can be separated into two groups. The first group use simple approaches which have severe limitations. Green and Oh’s transition consists of a pitch step command with PD tracking. However, the throttle is manually controlled by an RC pilot and only the LTH transition is discussed. Frank et al. also perform a near constant altitude transition with a simple control method. Their HTL transition is performed by using their relative position based LQR hover controller, but it is unclear how the throttle is controlled. However, they use an open-loop controller to perform the arguably more difficult LTH transition. While they were able to successfully perform the open-loop LTH transitions, all of the tests were indoors, using a Vicon camera system, and so did not have to deal with any disturbances.

The other group of controllers that perform constant altitude transitions all use optimal control techniques to generate the trajectory and, potentially, generate reference actuator and throttle commands. However, predefining the trajectory can cause complications due to varying initial conditions, mainly due to the presence of wind, and developing the required aerodynamic model requires significant effort and resources.

In this chapter we develop a simple transition approach that is able to perform the transitions with very little altitude variation. Our approach does not require any knowledge of the aircraft’s aerodynamics and can be incrementally tested and tuned in the field. However, it does require that the aircraft has constant altitude trim conditions throughout
the entire region between hover and level flight. We also develop an LQR based transition approach for comparison.

This chapter is organized in the following manner. First, the simplified dynamics are described in Section 7.2. The trajectory strategy will be discussed in Section 7.3. Following this, two different control methods, an LQR controller and one based on the total energy error, are developed in Section 7.4. Simulation results are presented in Section 7.5 and some conclusions are in Section 7.6.

7.2 Dynamics

A transition typically occurs with the wings level. Not only does this simplify the control scheme and provide more elevator control authority, it allows the dynamics to be simplified to two dimensions. The simplified two dimensional dynamics are

\[
\begin{align*}
\dot{x} &= \frac{1}{m} \left( F_x(x, h, \theta, q) + T \cos(\theta) \right) \\
\dot{h} &= -g - \frac{1}{m} \left( F_z(x, h, \theta, q) - T \sin(\theta) \right) \\
\dot{\theta} &= q \\
\dot{q} &= \frac{1}{J_y} \left( M_y(x, h, \theta, q) + M_e \right),
\end{align*}
\]  

(7.1) (7.2) (7.3) (7.4)

where \(x\) is the horizontal distance along the direction of the transition, \(h\) is the altitude, \(\theta\) is the pitch angle where 0 is horizontal and \(\frac{\pi}{2}\) is vertical, \(q\) is the pitch rate, \(F_x, F_z, \) and \(M_y\) are the aerodynamic forces and moments, \(T\) is the thrust, and \(M_e\) is the moment generated by the elevator. The thrust, aerodynamic forces and moments, and elevator moment are computed using the models in Chapter 3.

7.3 Trajectory Design

One interesting feature of the V-Bat is that it has constant altitude trim conditions for all of the airspeeds between stationary hover and the nominal cruise airspeed. Figure 7.1 shows the trim pitch and elevator and throttle commands for the airspeeds between hover and level flight. The existence of the constant altitude trim conditions covering the
entire transition region suggests a possible transition trajectory that follows the trim curve. In essence, the transition controller will attempt to move the aircraft smoothly along the constant altitude trim conditions. Once the desired rate at which we traverse this curve is selected, time-indexed commands for position, velocity, pitch, and pitch rate can be generated. In addition, feedforward elevator and throttle commands can also be generated. This allows a wide variety of transition controllers to be used with this trajectory.

Using a simple constant acceleration command, the reference trajectory for all of the states is computed by

\[
\dot{x}^r(t) = \dot{x}(0) + \int_0^t a^c(\tau) \delta_r^c \delta \tau
\]

\[
x^r(t) = x(0) + \int_0^t \dot{x}^r(\tau) \delta_r^c \delta \tau
\]

\[
\theta^r(t) = \theta^t(\dot{x}^c(t))
\]

\[
\dot{\dot{h}}^r = \dot{h}^r = 0
\]

\[
h^r = h(0),
\]

where we assume that the transition starts at \( t = 0 \), \( \theta^t(\dot{x}) \) is the trim pitch corresponding to the velocity \( \dot{x} \), and \( a^c \) is the constant acceleration command. Note that \( a^c > 0 \) for
the HTL transition and \( a^c < 0 \) for an LTH transition. The reference trajectory’s final time, \( t_f \), is chosen so that \( \dot{x}^r(t_f) \) equals the nominal cruise velocity, for the HTL transition, and \( \dot{x}^r(t_f) = 0 \) for the LTH transition. The desired angular rate is found by numerically differentiating \( \theta^r(t) \). Figure 7.2 shows a trajectory generated using this approach for a 2 m/s\(^2\) acceleration command.

Precomputing the time-indexed commanded states requires a high-fidelity model of the aircraft. In addition, the presence of wind adds significant complications. For example, the trim pitch depends on the airspeed yet we cannot estimate or measure the wind speed in near-hover flight. This will cause discrepancies on the position command and the relationship between velocity and pitch. Furthermore, wind will change the initial conditions for an HTL transition which can interfere with following a predefined trajectory.
Instead of precomputing the reference trajectory, our reference transition trajectory will consist of a reference velocity and a constant altitude command. Specifically,

\[
\dot{h}^r(t) \equiv 0
\]
\[
\ddot{x}^r(t) = a^c,
\]
and the reference velocity is generated by

\[
\dot{x}^r(t) = \dot{x}(0) + ta^c.
\]

Once again, the final time is chosen such so \(\dot{x}^r(t_f)\) equals the nominal cruise velocity, for the HTL transition, and \(\dot{x}^r(t_f) = 0\) for the LTH transition. By only commanding a groundspeed, the controller is allowed to determine the pitch naturally which reduces the performance degradation that more complete trajectories suffer from when wind is present.

There are several factors that need to be considered when determining \(a^c\). Many transition approaches try to minimize the total transition time which suggests that \(a^c\) should be as large as possible. However, the larger the acceleration, the more aggressive the transition and the more control authority that is needed. In addition, Maqsood and Go showed that, for a constant altitude transition, the shorter the transition time the higher the required thrust-to-weight ratio [71]. These factors suggest that a lower commanded acceleration should be used.

Our proposed method is related to other strategies discussed in the literature. For example, Osborne specifies the horizontal component of the trajectories by commanding a constant acceleration which depends on the distance to the desired stopping point [89]. In addition, he does not attempt to maintain a constant altitude. Frank et al. perform a constant altitude HTL transition by carefully selecting a position command to send to the LQR hover position controller that does not limit the aircraft’s speed [38].

Kubo and Suzuki also explored the trim conditions between hover and wing-borne flight for a tailsitter with flaps and slats [63]. Their aircraft has constant altitude non-stall trim conditions, without using the flaps and slats, for most of the transition region.
However, during low speed flight, 4 – 8 m/s, the main wings are stalled in trim if the aircraft is maintaining a constant altitude. The aircraft has non-climbing, non-stall trim conditions over the entire region if the flaps and slats were used. Because they require the vehicle not stall during the transition, their aircraft climbs during the LTH transition if the flaps and slats were not used.

7.4 Control Algorithms

This thesis develops and uses two different control strategies. First, an LQR approach will be discussed followed by an energy based controller.

7.4.1 LQR

An LQR controller requires a linear model of the system and its performance degrades with significant nonlinearities. Because the transition between hover and wing-borne flight covers a wide range of airspeeds and angle of attacks, there are significant nonlinearities in the transition dynamics. To account for this, linear models, spaced throughout the transition region, will be generated and each linear model will have its own LQR controller. The following aspects need to be determined:

1. The operating points for the linear models,

2. How many linear models are needed, and

3. When to switch between controllers.

The dynamics, (7.1)-(7.4), are linearized about an operating condition, $x_i$, to obtain

$$\dot{x} = A_i(x - x_i) + B_i(u - u_i)$$

$$y - y_i = C(x - x_i),$$
where

\[ x = [z \ x \ \dot{z} \ \theta \ q]^T \]
\[ u = [\delta_e \ \delta_i]^T \]
\[ C = I_5. \]

The trim conditions between hover and wing-borne flight can be used to determine the linear models’ operating points. First, \( N \) velocities, \( \dot{x}_i \) where \( i \in \{1, 2, \ldots, N\} \), are selected as the linear models’ operating velocity. The corresponding other operating states, such as pitch angle, are computed based on the trim curve. The operating velocities are selected to provide a decent distribution of velocities and pitches.

The active LQR controller is based on the current pitch angle. Whichever operating model’s pitch angle is closest to the current pitch angle determines which LQR controller is active. Linear interpolation could also be used to smoothly shift between the LQR gains but we are currently not using this approach.

### 7.4.2 Energy Transition

During the transition the altitude and airspeed dynamics are strongly coupled. Ideally, the control scheme handles this coupling directly which lead us to see if the TECS level flight control scheme, discussed in Chapter 6, can be modified to perform transitions between hover and level flight. However, it soon became clear that the normal TECS ideas are not applicable to a transition. The reason is best shown through an example. Consider the case when a tailsitter is in a stationary hover at the commanded location with no wind. If the throttle is controlling the total energy error then the throttle will be set to 0 because there is no total energy error. As the aircraft starts to descend, the kinetic energy is converted to potential energy while keeping the total energy constant assuming drag is negligible. Because the total energy error has not changed, the throttle will not change. Instead, the energy difference is changing which will cause the elevator to try and correct it. This will cause the aircraft to tilt instead of counteracting the descent.
As shown, the entire concept of using thrust to control the total energy and the elevator to control the energy difference breaks down when the aircraft is near vertical. In this condition, not only does the thrust control the total energy, it also strongly influences the energy difference by controlling the climb rate. On the other hand, in hover the elevator controls the horizontal velocity and does not strongly influence the climb rate. Therefore the thrust should control the altitude while the aircraft is vertical and control the horizontal velocity while level. Likewise, the elevator should control the horizontal velocity in hover flight and the altitude in level flight. Therefore, the thrust and elevator responsibilities should be mixed between hover and level modes.

Because of the major differences between hover and level flight, the transition controller cannot be based on the total energy and energy difference. However the desired behavior can be achieved by using the total energy along the different body axes. The true total energy error along the body axes is

\[
\tilde{E}_x = \left( m g (h^d - h) \right) + \frac{1}{2} m \left( \left( \dot{h}^d \right)^2 - \dot{h}^2 \right) \sin(\theta) + \frac{1}{2} m \left( \left( \dot{x}^d \right)^2 - \dot{x}^2 \right) \cos(\theta) \tag{7.10}
\]

\[
\tilde{E}_z = \left( m g (h^d - h) \right) + \frac{1}{2} m \left( \left( \dot{h}^d \right)^2 - \dot{h}^2 \right) \cos(\theta) + \frac{1}{2} m \left( \left( \dot{x}^d \right)^2 - \dot{x}^2 \right) \sin(\theta). \tag{7.11}
\]

However, this definition of the kinetic energy error, \( \frac{1}{2} m \left( \left( v^d \right)^2 - v^2 \right) \), does not take into account the direction of the velocity. While this can cause issues for the horizontal velocity tracking when the aircraft is near hover, it is especially troubling for the climb rate tracking. For example, a controller based on (7.10) and (7.11) would generate the same commands for a 1 m/s climb rate error and a -1 m/s climb rate error. This sign ambiguity means that the true total energy cannot be used. One possible correction is to not use the vertical kinetic energy, however, a better approach is to redefine the kinetic energy error to be

\[
\tilde{E}_{KE} = \frac{1}{2} m g (v^d |v^d| - v|v|).
\]

With this kinetic energy error definition the energy error in the body frame becomes
\[
\tilde{E}_x = \left( mg(h^d - h) + \frac{1}{2} m (\dot{h}^d|\dot{h}^d| - \dot{h}|\dot{h}|) \right) \sin(\theta) + \frac{1}{2} m (\dot{x}^d|\dot{x}^d| - \dot{x}|\dot{x}|) \cos(\theta) \\
\tilde{E}_z = \left( mg(h^d - h) + \frac{1}{2} m (\dot{h}^d|\dot{h}^d| - \dot{h}|\dot{h}|) \right) \cos(\theta) + \frac{1}{2} m (\dot{x}^d|\dot{x}^d| - \dot{x}|\dot{x}|) \sin(\theta).
\]

Not only does this energy error definition include the climb rate error, but it also correctly accounts for the horizontal velocity error during the LTH transition.

In level flight, where \( \theta \approx 0 \), \( \tilde{E}_x \approx \frac{1}{2} m (\dot{x}^d|\dot{x}^d| - \dot{x}|\dot{x}|) \) and \( \tilde{E}_z \approx mg(h^d - h) + \frac{1}{2} m (\dot{h}^d|\dot{h}^d| - \dot{h}|\dot{h}|) \). In other words, \( \tilde{E}_x \) contains the horizontal velocity error and \( \tilde{E}_z \) contains the altitude and climb rate error. The horizontal velocity can be increased by decreasing altitude, through modifying \( \theta \), or by increasing the throttle. Likewise, \( h \) and \( \dot{h} \) can be increased by increasing \( \theta \) or by increasing the throttle which would increase the airspeed and therefore lift. In hover, where \( \theta \approx \frac{\pi}{2} \), \( \tilde{E}_x \approx mg(h^d - h) + \frac{1}{2} m (\dot{h}^d|\dot{h}^d| - \dot{h}|\dot{h}|) \) and \( \tilde{E}_z \approx \frac{1}{2} m (\dot{x}^d)^2 - \dot{x}^2 \). In other words, \( \tilde{E}_x \) contains the altitude error and \( \tilde{E}_z \) contains the velocity error. The velocity can be increased by decreasing \( \theta \) and \( h \) can be increased by increasing the throttle. These two boundary cases suggest that \( \tilde{E}_x \) should be used to control the throttle and \( \tilde{E}_z \) should be used to control the pitch. While this does not capture the entire coupling between altitude and airspeed, because only the throttle will be used to control airspeed when \( \theta \approx 0 \) instead of both the throttle and elevator, it captures the physics sufficiently well to design a successful transition controller.

The energy error can be controlled with the PID controller

\[
\delta_t = k_{p,t} \tilde{E}_x - k_{d,t} \dot{\tilde{E}}_x + k_{i,t} \int_{t_0}^{t} \tilde{E}_x \delta_r + \delta_{t,ff} \\
\theta^c = k_{p,e} \tilde{E}_z - k_{d,e} \dot{\tilde{E}}_z + k_{i,e} \int_{t_0}^{t} \tilde{E}_z \delta_r.
\]

However, we have found that using the energy error rate as the derivative term can add significant oscillation. Instead, using

\[
\delta_t = k_{p,t} \dot{E}_x - k_{d,t} \ddot{E}_t + k_{i,t} \int_{t_0}^{t} \ddot{E}_x \delta_r + \delta_{t,ff} \tag{7.12}
\]

\[
\theta^c = k_{p,e} \dot{E}_z - k_{d,e} \ddot{E}_z + k_{i,e} \int_{t_0}^{t} \ddot{E}_z \delta_r. \tag{7.13}
\]
where
\[ \dot{E}_t = m \left( \sqrt{\dot{x}^2 + \dot{h}^2 + gh} \right) \]
is the total energy rate, provides a better response.

7.5 Simulation Results

Both transition controllers were tested in a variety of different situations. These tests use the full V-Bat simulator but the dynamics are constrained to two dimensions by forcing the lateral forces and moments to be zero. The aircraft always transitions into the wind when it is nonzero. The target groundspeed for the HTL transition is 0 m/s and the target velocity for the LTH transition is 15 m/s.

7.5.1 LQR

Several different tests were performed on the LQR transition controller. First, simulations were run that tested how many linear models and set of LQR gains were needed. Then different acceleration commands were tested and the effect of modeling errors were tested next. Finally, the effect of wind was explored. Unless otherwise noted, four linear models, index 4 in Table 7.1, were used with no modeling errors, there was no wind, and \( a^e = 2 \text{ m/s}^2 \). In each test the LQR controller was commanded to track the state commands generated by

\[ \dot{x}^e(t) = \dot{x}^r(t) \]  \hspace{1cm} (7.14)
\[ \theta^e(t) = \theta^r(t) \]  \hspace{1cm} (7.15)
\[ h^e(t) = h(t_0) \]  \hspace{1cm} (7.16)

if \( t \leq t_f \) otherwise

\[ \dot{x}^e(t) = \dot{x}^r(t_f) \]  \hspace{1cm} (7.17)
\[ \theta^e(t) = \theta^r(t_f) \]  \hspace{1cm} (7.18)
\[ h^e(t) = h(t_0) \]  \hspace{1cm} (7.19)
where the reference states were generated by (7.5)-(7.9). The climb rate and pitch rate commands were 0. The reference position was not used because the LQR controller was not controlling the position.

First, four different LQR transition controllers were tested with each controller using a different set of linear models and corresponding LQR gains. The linear models are specified by the operating velocity and the corresponding operating pitch and elevator and throttle commands are found through the trim curve. The first controller only uses linear models based on stationary hover and the nominal cruise velocity. The second controller also has two linear models but the operating velocities are within the transition region. The third controller uses three linear models that are selected by equally spacing out the operating velocities between the hover and level nominal velocities. Finally, the fourth controller uses four linear models that are also equally spaced. The controller’s operating velocities are summarized in Table 7.1.

![Table 7.1: Linear model operating velocities](image)

<table>
<thead>
<tr>
<th>Index</th>
<th>Velocities (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0, 15</td>
</tr>
<tr>
<td>2</td>
<td>5, 10</td>
</tr>
<tr>
<td>3</td>
<td>0, 7.5, 15</td>
</tr>
<tr>
<td>4</td>
<td>0, 5, 10, 15</td>
</tr>
</tbody>
</table>

Figure 7.3 shows the HTL and LTH transitions for the four different LQR transition controllers. As can be seen, all four controllers are successfully able to perform both transitions. It interesting that the first controller has the least altitude variation during the HTL transition but it has the most for the LTH transition. The second controller has steady state error in altitude for the HTL and groundspeed for the LTH due to the difference between the linear models and the true nonlinear dynamics at the final conditions. The remainder of the tests use the fourth controller.

Next, the LQR transition controller was tested with varying constant acceleration commands and the results are shown in Figure 7.4. Note that most of the acceleration commands tested are successful with comparable altitude response. The two exceptions are the \( a^c = -6 \text{ m/s}^2 \) and \( a^c = -7 \text{ m/s}^2 \) LTH transition which dropped eight meters. In
general, increasing the acceleration command increases the magnitude and number of the angular velocity and climb rate spikes for the HTL transition. Larger acceleration commands also correspond to a larger pitch overshoot for the LTH transition as well as larger angular velocities. These simulations suggest acceleration commands up to $5\,\text{m/s}^2$ for HTL and $-3\,\text{m/s}^2$ for LTH can be used for relatively smooth transitions.

Figure 7.5 compares the LQR transitions when there is significant modeling error. In each case, the predicted forces and moments are $\hat{F} = kF$, where $k$ is a constant scale factor and $F$ is the true force or moment. These tests show that while the LQR transitions are successful for $\pm 30\%$ modeling error, the performance can degrade significantly.

Figure 7.6 compares the LQR transitions when there is wind and the controller does not try to account for it. While the transitions are still successful, wind reduces the overall performance in regards to the altitude variation. However, there are ways that the controller
Figure 7.4: Comparison of the LQR transition with different acceleration commands

can try to mitigate the effect of wind. Assuming the the initial groundspeed is zero and the wind is constant, the HTL transition controller can use the initial pitch to determine the starting location on the transition trajectory. Let

\[ t_{\text{shift}} = \arg\min_t (|\theta^r(t) - \theta(0)|) \]  

be the time shift needed on the transition trajectory such that \( \theta^r(t_{\text{shift}}) \approx \theta(0) \). Then a velocity shift can be computed by looking at the velocity that corresponds to the pitch, i.e. \( v_{\text{shift}} = \dot{x}^r(t_{\text{shift}}) \). Assuming a perfect aerodynamic model then this velocity shift is the wind
speed. The trajectory commands can then be generated by

\[ \dot{x}^c(t) = \dot{x}^r(t + t_{\text{shift}}) - v_{\text{shift}} \]
\[ \theta^c(t) = \theta^r(t + t_{\text{shift}}) \]
\[ q^c(t) = q^r(t + t_{\text{shift}}), \]

and the altitude and climb rate commands are not modified.

A similar approach can be used for the LTH transition controller, however, the final time must be adjusted instead of the initial time. We assume that the wind can be measured in level flight so \( v_{\text{shift}} \) is known. The correct final time occurs when the ground speed is 0
m/s and can be computed by

\[ t_f = \arg \min_t (|\dot{x}^r(t) - v_{\text{shift}}|). \]  
(7.21)

The velocity trajectory command is then modified by

\[ \dot{x}^c(t) = \dot{x}^r(t) - v_{\text{shift}}, \]

and the other commands remain unchanged. Figure 7.7 shows the effect of the wind compensation. As can be seen, the altitude response significantly improves in all cases. However, these corrections have restrictive assumptions on the modeling accuracy and the wind.
Like the LQR transition controller, the energy transition controller was tested with a variety of cases. Figure 7.8a shows the HTL energy transition where $a^c = 2 \text{ m/s}^2$ and the target velocity is 15 m/s. Note that the energy transition controller has an initial delay before it starts accelerating and it has a small steady state error at the end. The steady state error is not a problem because the level flight controller will normally take over soon after the transition is complete. The altitude variation during the entire transition is roughly less than ±1 m which is impressive for a non-optimized transition strategy.

Figure 7.8b shows the LTH energy transition where $a^c = -3 \text{ m/s}^2$ and the initial velocity is 15 m/s. Like the HTL transition, the LTH transition has a slight delay in the velocity response. In addition, there is a velocity overshoot that the controller is able to cor-
This overshoot is not a problem because the hover flight controller would normally take control soon after the transition is complete. Once again, the altitude variation is less than ±1 m.

Figure 7.8: Energy transition with no wind

Figure 7.9 shows the energy transitions with varying amounts of constant wind. Note that as the wind increases during the HTL transition, Figure 7.9a, the oscillation in the altitude and climb rate increases at the end of the transition. This oscillation is caused by the energy transition controller not being tuned to operate the aircraft in level flight and will not cause a problem because the level flight controller will activate before the oscillation starts to occur. In any case, the oscillation is stable and will slowly decay. Even with this oscillation, the wind only increases the maximum altitude variation by centimeters. Arguably, wind improves the performance of the LTH energy transitions, Figure 7.9b, as
the velocity overshoot decreases as wind increases. The LTH transition altitude response is almost identical for all of the wind velocities tested.

Figure 7.9: Energy transition with wind

Figure 7.10 shows a situation where the energy transition controller is not ideal for the LTH transition. In level flight, the altitude and airspeed dynamics are coupled; however, the energy transition controller will only use the throttle to control the velocity when \( \theta = 0 \). If the transition is initialized when the pitch is too small, the controller will only decrease the throttle, in order to track the velocity command, and the pitch command will not increase. Unfortunately, the throttle does not decrease fast enough and drag does not increase fast enough to allow the aircraft to track the velocity command. Ideally, the aircraft would pitch up slightly to increase the drag but the energy error along \( k^b \) is too small. After a while, the
aircraft has lost enough airspeed that it loses altitude which causes $\dot{E}_z$ to increase and the pitch command to increase. The pitch increase causes the drag to increase which increases the aircraft’s deceleration and allows the aircraft to start tracking the velocity command. The root cause of this issue is that the energy transition controller does not fully take into account the coupling between airspeed and altitude in level flight. Fortunately, this issue can be avoided by using the level controller to decrease the airspeed enough and then initialize the transition controller. In our experience, the initial pitch should be around $15^\circ$ for the energy LTH transition to work properly.

![Figure 7.10: Energy LTH transition when the initial pitch is low](image)

One interesting feature of the energy transition controller is that it allows the transitions to be tested incrementally by selecting a lower target velocity. Figure 7.11 shows the ability to incrementally test the energy transition controller with the constant altitude constant acceleration trajectory. In both figures, the aircraft starts in stationary hover with
no wind and is commanded to accelerate at 3 m/s² until it reaches the velocity command. The aircraft then holds the velocity for two seconds and is then commanded to decelerate at 2 m/s². The energy transition controller is used during the entire test.

![Graphs showing flight parameters](image)

(a) 6 m/s maximum velocity command
(b) 11 m/s maximum velocity command

Figure 7.11: Incremental testing of the energy transition controller

### 7.6 Conclusions

In this chapter two different trajectory generation schemes and two different control schemes were developed. The LQR control scheme and its complete trajectory required an accurate model of the aircraft’s aerodynamics while the energy based controller and its trajectory do not require any aerodynamic model. However, both trajectory approaches require that the aircraft has a smooth constant altitude trim curve throughout the entire region between hover and level flight.
The LQR transition controller, which requires the more complete transition trajectory, is a series of LQR controllers with operating points distributed along the constant altitude trim curve. Tests showed that this controller, with a perfect model, could perform transitions with almost no altitude variation. As expected, modeling error degraded the performance but the transitions were still successful. In addition, wind also reduced the performance but it could be mitigated if the trim curve is accurate.

The energy transition controller with the simple constant altitude constant acceleration trajectory was able to perform transitions with very little altitude variation and it requires no knowledge of the aircraft’s aerodynamics. In addition, wind produced only minor changes in the transition and did not need to be taken into account. This controller is able to be incrementally tested allowing it to be easily tested and tuned in the field.

There are several things that could be explored in the future. First, flight tests should be performed on both transition approaches to validate them. Second, the adaptive control techniques developed in [69] could be added to the LQR transition controller. This would enable the controller to better handle the model mismatch. Finally, as mentioned earlier, it is difficult to compare the different transition approaches due to the wide variety of aircraft they were developed for. It would be very beneficial to implement most, if not all, of the relevant approaches on a single airframe allowing direct comparisons to be made.
Chapter 8. Conclusions

In this dissertation we developed a complete control system for a tailsitter with a ducted fan. In our approach, each flight mode has its own controller which share a common attitude control scheme. Overall, our control scheme generates acceptable performance for the Vertical Bat.

Figure 8.1 shows a simulated complete flight. This flight used the REA-QA hybrid backstepping attitude controller with the stabilized RLS parameter estimation scheme, the inertial position PID hover controllers with the modification to improve performance when initialized at large velocities, the nonlinear non-adaptive TECS controller, and the constant altitude, constant acceleration transition controller.

The remainder of this chapter is organized in the following manner. Section 8.1 contains an overview of the conclusions for each control scheme and some recommendations. Some ideas for future research are included in Section 8.2.

8.1 Conclusions and Recommendations

The main objective of this dissertation was to develop control schemes that provided excellent performance while not requiring any aerodynamic information. This has been accomplished to varying degrees for each flight mode. To reiterate, the main contributions in this dissertation were:

- The resolved Euler angle attitude error method that computes the attitude error in a way that generates the desired trajectories for a hovering tailsitter which the standard approach in the literature does not do,

- A hybrid attitude control system, based on the resolved Euler angle attitude error, that provides almost global asymptotic stability,
A hover control system that can handle being initialized in a wide variety of initial conditions,

A novel nonlinear total energy controller for the altitude and the airspeed during fixed-wing flight that provides asymptotic stability even with inaccurate thrust and drag models,

A simple constant altitude constant acceleration transition trajectory that can be incrementally tested and tuned,

An energy based transition controller that requires almost no knowledge of the aircraft’s aerodynamics yet can perform near constant altitude fully autonomous transitions.

The main conclusions and recommendations for the individual controllers are summarized in the following subsections.

### 8.1.1 Attitude Control

We showed that the standard QA error does not generate ideal trajectories for a hovering tailsitter when the aircraft is commanded to change its heading and tilt at the same time. We also showed that the only approach to correct this in the literature, the RTT
error, breaks down for large attitude errors. Instead of attempting to correct the RTT error, we developed the REA error, which generates similar trajectories to the RTT error, and a saturation method that guarantees that the REA error will not break down. The REA error can be used with a PID based attitude controller with no knowledge of the aircraft’s model and it performs better than using the QA error. Guaranteed almost global asymptotic stability can be obtained through a hybrid backstepping controller, which uses the REA and QA error, if the moments of inertia matrix and the control vane response are known. While the inertia matrix can be computed relatively easily, the effect of the control vanes cannot. However, the control vane response can be estimated with a RLS algorithm or another similar estimation algorithms. Simulation results show that PID controllers based on the REA and RTT error produce almost identical responses and reduce undesired translational velocity, as compared to the QA error, when the aircraft is commanded to tilt and pitch at the same time. Simulation results also confirm that the hybrid controller reduces the undesired translational velocity.

If quaternions are used to represent the attitude then we recommend that the REA error with saturation be used over the standard QA error. While the situations where the REA error provides a better response than the QA error will not occur very often, the computational burden of the REA error is only slightly larger than the QA error. The REA-QA hybrid controller can be used if the stability guarantee is desired. In addition, the REA-QA hybrid controller does not require gain scheduling, unlike the PID attitude controllers, however it is significantly more difficult to tune compared to the PID controllers.

8.1.2 Hover Control

Our hover controller is an inertial position based PID controller that tracks position and altitude second-order reference models and does not require any knowledge of the aircraft’s aerodynamics. Flight tests showed that this controller has acceptable performance when it is initialized when the aircraft is close to a stationary hover. However, performance significantly degrades when the aircraft has a large initial velocity. While most controllers, in the literature, do not take this behavior into account, we argue that it is necessary because there is a wide range of initial conditions that may occur after a transition. The baseline
controller was then modified to improve the performance during these situations. Simulation and flight tests confirmed that the modifications, modifying the derivative term when the velocity is large and initializing the integrator, drastically improve the performance when the controller is initialized when the aircraft is moving fast.

8.1.3 Level Flight Control

We derived a nonlinear version of the TECS controller to control the altitude and airspeed of the aircraft in level flight. Unlike typical level altitude and airspeed controllers, TECS controllers take into account the coupling between altitude and airspeed. Our nonlinear TECS controller provides additional mathematical justification for the structure of the original and current standard TECS controllers. In addition, the nonlinear TECS controller can guarantee asymptotic stability if it includes an adaptive element. However, better performance can be achieved by removing the adaptive component and adding feedback to the guidance command generation. The non-adaptive nonlinear TECS controller loses the asymptotic stability guarantee but linear analysis shows the system is stable and there is zero steady state error for step altitude and airspeed commands. In any case, the nonlinear TECS controllers provide better performance than typical SLC controllers and is much easier, qualitatively, to tune than other TECS controllers.

Both variants of the nonlinear TECS controller require knowledge of the aircraft’s thrust and drag characteristics. We developed a method, which only requires a static thrust bench test and data from several flights, to estimate the required parameters. While the estimated parameters are not perfect, simulated flights using the estimated parameters showed they are accurate enough.

8.1.4 Transition Control

We developed two different transition approaches that are able to perform near constant altitude transitions; however, both approaches require that the aircraft has constant altitude trim conditions between stationary hover and the level-flight cruise velocity. The first controller consisted of several LQR controllers, based on linear models distributed along
the constant altitude trim curve, and the active LQR controller was selected based on the current pitch. This controller was able to perform near constant altitude transitions, by following the constant altitude trim curve with a constant acceleration, but model inaccuracies and wind degraded the performance. The second controller was inspired by the TECS concepts used for the level flight controller and is based on the total energy error along the body axes. While the energy transition controller had more altitude variation during the transitions, as compared with the LQR transition controller, it was still able to perform near constant altitude transitions by tracking a constant acceleration command. In addition, the energy transition controller was resistant to wind and required almost no knowledge of the aircraft’s aerodynamics. This approach is the only method, to our knowledge, that can perform autonomous near constant altitude transitions without a detailed model of the aircraft.

We cannot recommend our transition approach for a general tailsitter because not all tailsitters can maintain a constant altitude at all airspeeds between hover and the nominal cruise airspeed. In addition, many people prefer that the transitions occur as fast as possible which our approach does not attempt. However, for the aircraft that are capable of performing it and for those who want a smoother, less aggressive transition, our approach is an attractive option.

8.2 Future Work

Several suggestions for future work are summarized below.

- There are several issues with the Vertical Bat aerodynamic model caused by the limited range of wind tunnel tests that were performed. These issues could be resolved through more extensive wind tunnel tests or significant computational fluid dynamics modeling and testing. Furthermore, the latter approach would allow a better duct model to be used. In any case, the current model matches reality reasonably well in hover as shown by the minimal tuning that was needed for flight tests.

- The propeller model should be dependent on the current battery voltage such as in [58]. This would improve the accuracy of the simulation during longer flights.
• Online parameter estimation schemes might provide better performance than the adaptive nonlinear TECS variant.

• It might be possible to use some of the OL-MRAC ideas to incorporate the guidance feedback into the nonlinear TECS controller derivation. If possible, this would strengthen the stability guarantee for that nonlinear TECS controller variant.

• The LQR transition controller could be augmented with an adaptive element, such as in [69], to reduce the impact of modeling errors.

• The LQR transition controller could be modified so it linearly interpolates between the LQR controllers, based on the current pitch, instead of using hard switches. This should reduce the spikes in the commands and the angular velocity that currently occur during the simulated flights.

• Flight tests should be performed on the nonlinear TECS level flight controller, the LQR transition controller, and the energy based transition controller. Not only will this validate the controllers but it will validate the constant altitude constant acceleration trajectory scheme.

• As mentioned earlier, it is impossible to compare the performance of the various transition algorithms in the literature due to the wide range of airframes used during the development and testing. It would be very informative if the various approaches were implemented on a single airframe allowing direct comparisons. However, not all of the approaches could be compared this way because some methods require specific aircraft features such as variable-incidence wings. In any case, this comparison would allow a designer to pick the best control scheme for their application.

• Performance could be improved by controlling the thrust generated by the propeller instead of selecting a throttle command. The simple thrust controller developed in Chapter 6 assumes that the airspeed is always perpendicular to the propeller. While this is reasonable for level flight, especially with the duct, it is not true in general and cannot be used in the other flight modes. One possible approach is the adaptive thrust controller developed in [58].
Bibliography


[103] ——, “Aerodynamic Modelling of a Wing-In-Slipstream Tail-Sitter UAV,” in Biennial International Powered Lift Conference and Exhibit, 2002. 16


186


Appendix A. Vertical Bat Aerodynamic Parameters

This appendix contains the airframe parameters used in the dynamics model. The lift coefficient model is

\[
C_L(\alpha) = \begin{cases} 
-0.0894 + 1.27\alpha, & -35.5 \leq \frac{\pi}{180} \alpha \leq -9.5, \\
-0.7522 + 0.2\alpha, & -54 \leq \frac{\pi}{180} \alpha \leq -35.5, \\
-2.4427 - 1.6\alpha, & -100 \leq \frac{\pi}{180} \alpha \leq -54, \\
0.483 + 4.78\alpha, & -9.5 \leq \frac{\pi}{180} \alpha \leq 7.5, \\
1.2399 - \alpha, & 7.5 \leq \frac{\pi}{180} \alpha \leq 12, \\
0.7 + 1.27\alpha - 0.941\alpha^2, & 12 \leq \frac{\pi}{180} \alpha \leq 53, \\
2.55 - 1.6\alpha, & 53 \leq \frac{\pi}{180} \alpha \leq 100.
\end{cases}
\]

The level flight drag coefficient model is

\[
C_D(\alpha) = \begin{cases} 
-0.7 - 3.37\alpha - 1.15\alpha^2, & -100 \leq \frac{\pi}{180} \alpha \leq -66, \\
-0.1074 - 1.55\alpha & -66 \leq \frac{\pi}{180} \alpha \leq -12, \\
0.07 - 0.14\alpha + 2.7\alpha^2, & -12 \leq \frac{\pi}{180} \alpha \leq 11, \\
1.48 + -0.08\alpha, & 11 \leq \frac{\pi}{180} \alpha \leq 70, \\
-1.57 + 4.6\alpha - 1.55\alpha^2, & 70 \leq \frac{\pi}{180} \alpha \leq 100.
\end{cases}
\]

The hover flight drag, side force and rolling moment coefficients are

\[
C_D(\beta_h) = 3.14 \sin(1.03\beta_h + 1.48)^2 + 0.7220
\]

\[
C_S(\beta_h) = 0.94 \sin(2\beta_h)
\]

\[
C_l(\beta_h) = -0.13 \sin(2\beta_h).
\]

The rest of the parameters are listed in Table A.1.
Table A.1: Vertical Bat airframe parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>5.443 kg</td>
<td>$C_{mq}$</td>
<td>-1.3990</td>
</tr>
<tr>
<td>$J_x$</td>
<td>0.3745 kg m$^2$</td>
<td>$C_{yp}$</td>
<td>0</td>
</tr>
<tr>
<td>$J_y$</td>
<td>1.4327 kg m$^2$</td>
<td>$C_{yr}$</td>
<td>0</td>
</tr>
<tr>
<td>$J_z$</td>
<td>1.6931 kg m$^2$</td>
<td>$C_{tp}$</td>
<td>-0.3209</td>
</tr>
<tr>
<td>$J_{xz}$</td>
<td>0.0015 kg m$^2$</td>
<td>$C_{tr}$</td>
<td>0.03066</td>
</tr>
<tr>
<td>$J_{xx}$</td>
<td>0.696 m$^2$</td>
<td>$C_{np}$</td>
<td>0.01297</td>
</tr>
<tr>
<td>$b$</td>
<td>1.8479 m</td>
<td>$C_{nr}$</td>
<td>-0.00434</td>
</tr>
<tr>
<td>$c$</td>
<td>0.2 m</td>
<td>$C_{m0}$</td>
<td>-0.0721</td>
</tr>
<tr>
<td>$C_{y,0}$</td>
<td>0</td>
<td>$C_{ma}$</td>
<td>-0.7120</td>
</tr>
<tr>
<td>$C_{l,0}$</td>
<td>0</td>
<td>$C_{y\beta}$</td>
<td>0.125</td>
</tr>
<tr>
<td>$C_{n,0}$</td>
<td>0</td>
<td>$C_{l\beta}$</td>
<td>-0.167</td>
</tr>
<tr>
<td>$CL_q$</td>
<td>2.8932</td>
<td>$C_{Lq}$</td>
<td>0.0475</td>
</tr>
<tr>
<td>$C_{dq}$</td>
<td>0</td>
<td>$K_{twist}$</td>
<td>0.4860</td>
</tr>
<tr>
<td>$a_o$</td>
<td>1.3860</td>
<td>$C_{L,v}$</td>
<td>0.25</td>
</tr>
<tr>
<td>$b_p$</td>
<td>2</td>
<td>$C_{D,v}$</td>
<td>0.1659</td>
</tr>
<tr>
<td>$c_p$</td>
<td>0.04</td>
<td>$S_{v}$</td>
<td>0.17 m$^2$</td>
</tr>
<tr>
<td>$r_p$</td>
<td>0.2286</td>
<td>$d_{v,cp}$</td>
<td>0.71 m</td>
</tr>
<tr>
<td>$k_{aug}$</td>
<td>0.4548</td>
<td>$l_v$</td>
<td>0.15 m</td>
</tr>
<tr>
<td>$c_{duct}$</td>
<td>0.1524 m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{duct}$</td>
<td>0.2377 m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$l_{duct}$</td>
<td>0.7 m</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Algorithm 1: Resolved Tilt Twist attitude error

- **input**: $\eta$, $\eta^d$
- **output**: $e$

\[
R_v^b = \text{Q2RM}(\eta) ; \\
R_v^d = \text{Q2RM}(\eta^d) ; \\
R_b^d = R_v^d(R_v^b)^\top ; \\
\tilde{\Theta}_y = -\text{atan2}(R_d^b(1, 3), R_b^d(1, 1)) ; \\
\tilde{\Theta}_z = \text{atan2}(R_d^b(1, 2), R_b^d(1, 1)) ; \\
\Theta_{\text{tilt}} = \cos^{-1}(R_v^b(1,:)(R_v^d(1,:))^\top) ; \\
\text{if } \Theta_{\text{tilt}} \geq 0.001 \text{ then} \\
\begin{align*}
\nu^B &= R_v^b \left( \frac{\nu^B}{\|\nu^B\|} \right) ; \\
\nu^{B\times} &= \begin{bmatrix} 0 & -\nu_3^B & \nu_2^B \\ \nu_3^B & 0 & -\nu_1^B \\ -\nu_2^B & \nu_1^B & 0 \end{bmatrix} ; \\
R_{\nu^B} &= I_3 - \left[ \nu^{B\times} \right] \sin(\Theta_{\text{tilt}}) + \left[ \nu^{B\times} \right]^2 (1 - \cos(\Theta_{\text{tilt}})) ; \\
\text{else} \\
R_{\nu^B} &= I_3 ; \\
\text{end} \\
R^a = R_{\nu^B}^\top R_v^b ; \\
\Theta_{\text{twist}} = \cos^{-1}(R_v^a(3,:)(R_v^d(3,:))^\top) ; \\
\Theta_{\text{sign}} = \cos^{-1}(R_v^a(2,:)(R_v^d(2,:))^\top) ; \\
\text{if } \Theta_{\text{sign}} \leq \frac{\pi}{2} \text{ then} \\
\tilde{\Theta}_x = -\Theta_{\text{twist}} ; \\
\text{else} \\
\tilde{\Theta}_x = \Theta_{\text{twist}} ; \\
\text{end} \\
e = [\tilde{\Theta}_x \tilde{\Theta}_y \tilde{\Theta}_z]^\top
\]
Algorithm 2: REA angle of rotation saturation

input: \( \tilde{\eta}, \tilde{\gamma}, R_s^b \)
output: \( \tilde{\eta}, \tilde{e} \)

if \( \cos^{-1}(\tilde{\eta}_0) > \frac{\pi}{2} \) then
    \( \tilde{\eta} = [\tilde{\eta}_1 \tilde{\eta}_2 \tilde{\eta}_3]^\top \);
    \( \nu = \frac{\tilde{\eta}}{||\tilde{\eta}||} ; \)
    \( \tilde{\eta} = [\cos(\frac{\tilde{\gamma}}{2}) \sin(\frac{\tilde{\gamma}}{2}) \nu^\top]^\top ; \)
else
    \( \tilde{\eta} = \tilde{\eta} ; \)
end
\( \tilde{e} = R_s^b Q2E(\tilde{\eta}) ; \)

Algorithm 3: REA selective saturation

input: \( \tilde{\eta}, \bar{\theta}, R_s^b \)
output: \( \tilde{\eta}, \tilde{e} \)

\( \theta_2 = \sin^{-1}(2(\tilde{\eta}_0 \tilde{\eta}_2 - \tilde{\eta}_1 \tilde{\eta}_3)) ; \)
if \( |\theta_2| > \bar{\theta} \) then
    \( \tilde{\eta} = [\tilde{\eta}_1 \tilde{\eta}_2 \tilde{\eta}_3]^\top ; \)
    \( \nu = \frac{\tilde{\eta}}{||\tilde{\eta}||} ; \)
    \( \tilde{\eta} = \text{REA LimitAngleOfRotation}(\theta_2, \bar{\theta}, \nu) ; \)
else
    \( \tilde{\eta} = \tilde{\eta} ; \)
end
\( \tilde{e} = R_s^b Q2E(\tilde{\eta}) ; \)

Algorithm 4: REA limit angle of rotation

input: \( \theta_2, \bar{\theta}, \nu \)
output: \( \tilde{\eta} \)

if \( \theta_2 > 0 \) then
    \( \tilde{\gamma} = -\cos^{-1}\left(\frac{\sin(\bar{\theta}) + \nu_1 \nu_3}{\sqrt{\nu_2^2 + \nu_1^2 \nu_3^2}}\right) + \text{atan2}(\nu_2, \nu_1 \nu_3) ; \)
else
    \( \tilde{\gamma} = \cos^{-1}\left(\frac{-\sin(\bar{\theta}) + \nu_1 \nu_3}{\sqrt{\nu_2^2 + \nu_1^2 \nu_3^2}}\right) + \text{atan2}(\nu_2, \nu_1 \nu_3) ; \)
end
\( \tilde{\eta} = [\cos(\frac{\tilde{\gamma}}{2}) \sin(\frac{\tilde{\gamma}}{2}) \nu^\top]^\top ; \)
Algorithm 5: REA modified selective saturation

input : \( \vec{\eta}, \bar{\theta}, R^b_s \)
output: \( \vec{\eta}, e \)

\( \vec{\eta} = [\vec{\eta}_1 \vec{\eta}_2 \vec{\eta}_3]^\top \);
\( \nu = \frac{\vec{\eta}}{||\vec{\eta}||} ; \)
\( \theta_2 = \sin^{-1}(2(\vec{\eta}_0 \vec{\eta}_2 - \vec{\eta}_1 \vec{\eta}_3)) ; \)
if \( |\theta_2| > \hat{\theta} \) then
    \( \vec{\eta} = \text{REA LimitAngleOfRotation}(\theta_2, \bar{\theta}, \nu) ; \)
else
    \begin{align*}
    x_1 &= \frac{1+\nu_1\nu_3}{\sqrt{\nu_2^2+\nu_1^2\nu_3^2}} ; \\
    x_2 &= \frac{-1+\nu_1\nu_3}{\sqrt{\nu_2^2+\nu_1^2\nu_3^2}} ; \\
    \gamma &= 2\cos^{-1}(\vec{\eta}_0) ; \\
    \bar{\gamma} &= \infty ; \\
    \text{if } x_1 \leq 1.0001 \text{ then} \\
    \bar{\gamma} &= \text{atan2}(\nu_2, \nu_1\nu_3) ; \\
    \text{else if } x_2 \geq -1.0001 \text{ then} \\
    \text{if } \text{atan2}(\nu_2, \nu_1\nu_3) > 0 \text{ then} \\
    \bar{\gamma} &= \pi + \text{atan2}(\nu_2, \nu_1\nu_3) ; \\
    \text{else} \\
    \bar{\gamma} &= -\pi + \text{atan2}(\nu_2, \nu_1\nu_3) ; \\
    \end{align*}
end
if \( |\gamma| > |\bar{\gamma}| \) then
    \( \vec{\eta}' = \left[ \cos\left(\frac{|\gamma|}{2}\right) \sin\left(\frac{|\gamma|}{2}\right) \nu \right]^\top ; \)
    \( \theta_2' = \sin^{-1}(2(\vec{\eta}_0' \vec{\eta}_2' - \vec{\eta}_1' \vec{\eta}_3')) ; \)
    if \( |\theta_2'| > \hat{\theta} \) then
        \( \vec{\eta} = \text{REA LimitAngleOfRotation}(\theta_2', \bar{\theta}, \nu) ; \)
    else
        \( \vec{\eta} = \vec{\eta}' ; \)
    end
else
    \( \vec{\eta} = \vec{\eta}' ; \)
end
end
\( e = R^b_s Q2E(\vec{\eta}) ; \)
Appendix C. Hardware in the Loop Simulation

Hardware in the loop (HIL) simulators provide an unparalleled ability to test and debug algorithms and code before risking aircraft in flight tests. However, because this was the first time the MAGICC Lab has used the Kestrel 3, none of the MAGICC Lab’s existing HIL capabilities were sufficient.

Jeff Sanders, a MAGICC Lab alumnus, wrote a HIL simulator for the Kestrel 2.4 during his time in the lab \[94\]. It was Jeff’s hope that his HIL simulator would provide a lasting benefit for the MAGICC Lab. While it was used for several years, it was not sufficient for this project for several reasons. First, it was only compatible with the Kestrel 2.4’s communication protocol. This limitation could be overcome, but the underlying structure, shown in Figure C.1, assumed that only the Kestrel 2.4, its ground station Virtual Cockpit (VC), and MATLAB would be used.

Initially, we thought it would be possible to use X-Plane, a commercial flight simulator which estimates the aerodynamics based on the airframe’s geometry, to run the dynamics. While Indriyanto and Jenie were able to add a ducted-fan aircraft to X-Plane we were unable to get the control vanes working properly \[45\]. However, X-Plane can be configured to only act as a graphics engine. In this mode, X-Plane will display the aircraft within accurate real world locations based on the position and attitude that are sent to it.

![Figure C.1: The Kestrel 2.4 hardware in the loop simulator. MATLAB, which runs the dynamics, and, depending on the project, the path planning algorithm, communicates with Virtual Cockpit by using a c++ mex function that creates a TCP/IP connection with Virtual Cockpit. Virtual Cockpit accepts the packets from MATLAB and forwards them to the Kestrel 2.4 over a RS-232 communication link. Virtual Cockpit sends the Kestrel 2.4’s state estimates back to MATLAB over the same TCP/IP socket.](image)

193
Intra-Communication Thread

Interface 1

Object 1

Object 2

... 

Object N

Interface 2

... 

Interface N

Intra Communication Thread

Figure C.2: The MAGICC Interface structure. An interface is responsible for communicating with an outside object, either hardware or software, and creates the correct internal packet type when it receives a packet from its object. These internal packets are sent to the intra-communication thread which forwards them to the other interfaces based on forwarding rules created by the user. The other interfaces then format the internal packets and send them to the objects they connect to.

Because we wanted to be able to use either X-Plane or MATLAB for the dynamics we decided to make the HIL software modular. The structure of the software, which we call MAGICC Interface, is shown in Figure C.2. As can be seen, there are two main parts: interfaces and the intra-communication thread. An interface is in charge of communicating with one object which can be hardware or software. The interface listens for communication from its object within a dedicated thread using TCP/IP, UDP, or serial protocols. If a packet is received, the interface parses the data and creates the appropriate internal packet. There are several possible internal packet types which can contain, among others, true states, state estimates, sensor values, GPS values, or commands. The internal packet is sent to the intra-communication thread which determines which interfaces receive the internal packet based on user defined rules. When an interface receives an internal packet, it parses the data, formats it in a manner that its object can understand, and sends it.

This structure allows an arbitrary number of objects to communicate with each other. For example, MAGICC Interface can handle several instances of MATLAB, one instance of VC, and one instance of X-Plane all communicating with each other with arbitrary rules.
on who receives what type of packets and from whom. In addition, tools to create, view, record, and playback internal packets have been implemented to aid in debugging, testing, and visualizing any algorithms being tested.

Currently three interface types have been created: MATLAB, X-Plane, and Kestrel 3. The MATLAB interface can communicate with MATLAB over UDP by using MATLAB’s Instrument Control Toolbox. This allows MATLAB to run the dynamics and/or the control algorithms. The X-Plane interface communicates with X-Plane over UDP using X-Plane’s built in network interface. X-Plane can only run the dynamics for one airframe but it can act as the display for multiple vehicles. The Kestrel 3 interface can communicate with VC by using VC’s TCP/IP developer server port or the interface can communicate directly with the Kestrel 3 over a RS-232 channel. While the communication through VC is simpler to set up, it requires a fast computer and is limited to roughly 100 Hz with a modern Intel Core i7 processor. Interfaces for the Kestrel 2.4 or Mavlink, which would allow communication with a Px4 or an ArduPilot, could easily be added.

The V-Bat HIL simulator is shown in Figure C.3. MATLAB runs the dynamics and sends the true position, velocity, attitude, angular velocity, and net force to the MAGICC Interface. The MAGICC Interface adds noise to the true states and creates sensor packets, which contain absolute and dynamic pressure, accelerometer measurements, rate gyro measurements, and a magnetic field reading, and GPS packets with the latitude, longitude, and altitude. It then forwards these packets, at a rate of four sensor packets for every GPS packet, to VC which sends them to the Kestrel 3. The Kestrel 3 runs the control and estimation algorithms and sends the aileron, elevator, rudder, and throttle commands to the MAGICC Interface, via VC, which formats them and sends them to MATLAB.
Figure C.3: The Kestrel 3 hardware in the loop structure. MATLAB, which is running the dynamics, communicates with the MAGICC Interface using the UDP Simulink block from the Instrument Control Toolbox. The MAGICC Interface accepts the true states from MATLAB, simulates the noisy sensor measurements, creates sensor and GPS packets, and sends them to Virtual Cockpit over a TCP/IP connection. Virtual Cockpit accepts the packets and forwards them to the Kestrel 3. The Kestrel 3 sends a command packet, which contains the aileron, elevator, rudder, and throttle commands, to Virtual Cockpit for every sensor packet it receives. Virtual Cockpit forwards the command packets to the MAGICC Interface which formats them and sends them to MATLAB.