NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ADAPTIVE CONTROL FOR FIXED WING AIRCRAFT

by

Ryan G. Beall

June 2017

Thesis Advisors:	Oleg Yakimenko
	Vladimir Dobrokhodov
Second Reader:	Fotis Papoulias

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited

ADAPTIVE CONTROL FOR FIXED WING AIRCRAFT

Ryan G. Beall
LT, USN
B.S., United States Naval Academy, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2017

Approved by: Oleg Yakimenko
Thesis Advisor

Vladimir Dobrokhodov
Thesis Advisor

Fotis Papoulias
Second Reader

Ronald Giachetti
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The field of adaptive control offers techniques for increasing performance and robustness in numerous settings and applications. Adaptive control is different than traditional feedback in that it offers a mechanism for adjusting the controller's parameters to reduce plant uncertainty. Traditional feedback control utilizes parameters, which are specified by the engineer to optimize an ideal use case, which often times requires extensive tuning and testing. Adaptive controllers adjust their control parameters using various intelligent mechanisms designed to increase robustness to plant variation or unanticipated disturbances. Adaptive control has many applications in the aerospace domain to include control strategies when aerodynamic coefficients are unknown or are non-constant, actuator failure, airframe damage, etc. This research evaluates fixed wing UAS controller performance and robustness using the $\mathcal{L}_1$ adaptive control architecture.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

DoD	Department of Defense
NPS	Naval Postgraduate School
USN	U.S. Navy
API	Application Program Interface
APM	ArduPilotMega / Multi-Platform Autopilot
CG	Center of Gravity
EKF	Extended Kalman Filter
FIR	Finite Impulse Response
GCS	Ground Control Station
GNC	Guidance Navigation and Control
GUI	Graphical User Interface
IIR	Infinite Impulse Response
IMU	inertial measurement unit
LTI	Linear Time Invariant
MAV	Mirco Aerial Vehcile
MEMS	Microelectromechanical Systems
MIMO	Multiple Input Multiple Output
MRAC	Model Reference Adaptive Control
NED	North East Down

ODE	Ordinary Differential Equation
OS	Operating System
PCHIP	Piecewise Cubic Hermite Interpolating Polynomial
PID	Proportional Integral Differential
PWM	Pulse Width Modulation
RC	Remote Controlled
SISO	Single Input Single Output
SITL	Software in the Loop
TF	Transfer Function
UAS	unmanned aerial system
VTOL	vertical take off and landing
ZOH	zero order hold

Executive Summary

Executive Summary Here!.....

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

I would like to thank.....

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction and History

1.1 Problem Statement

The unmanned aerial system (UAS) has evolved tremendously over the past decade. Miniature autopilots have gotten smaller and cheaper with more sensitive and redundant sensor packages largely due to the cellular phone industry accelerating Microelectromechanical Systems (MEMS) technology. The ability to manufacture these autonomous systems at fractions of the cost enables the advancement in multiple cooperative UAV applications including swarming capability. This ability to mass-produce large quantities of UAS's poses an interesting challenge. Even though the price has gone down and the performance has gone up, there still exists a significant amount of man-hours dedicated to sensor calibration and autopilot control law configuration and tuning for optimal performance.

The Department of Defense (DoD) conducted an analysis of the role of autonomy, which outlined technology gaps and predicted advancements required to meet the growing performance demands [1]. The study amplifies the fact that autonomy is a challenging field and that it is arguably in its infancy. The roadmap attempts to guide decision makers in ensuring capitalization of under-utilized technology and succinct awareness of technical challenges limiting the current state of the art. Figure 1.1 outlines these elements at increasing scope of control comprised of various technology portfolios. The language referenced in the roadmap often recommends that machine learning and/or artificial intelligence is needed for elements such as "Fault Detection." On the lowest level, the roadmap annotates the use of Guidance Navigation and Control (GNC) as neither needing improvement or current technology being underutilized. An in-depth look at the current state of the art with respect to GNC reveals that GNC is still very costly and arguably antiquated. Research of adaptive control as applied to GNC offers new strategies offering improved performance and future reduced cost.

UAS avionics have drastically improved over the past decade, but the fundamental control law algorithms have not changed. The Proportional Integral Differential (PID) architecture

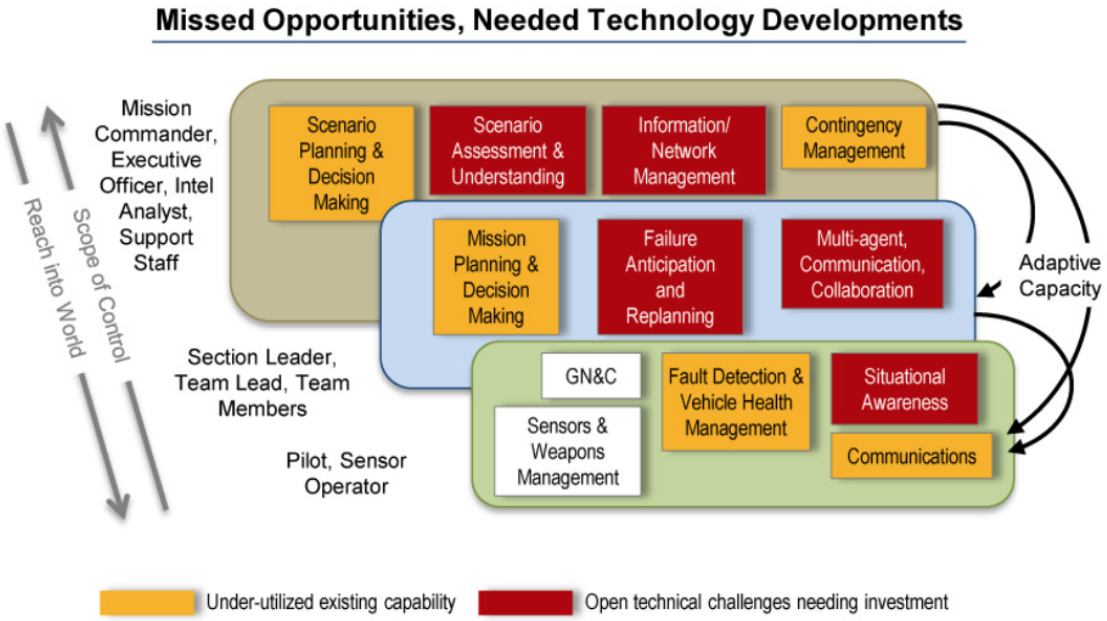

Figure 1.1. DoD Autonomy Roadmap [1]

found its origin in automatic ship steering applications in 1922 [2]. Conventional control law architectures for UAS's predominately still use PID controllers. Conventional control law architectures for UAS's predominately still use PID controllers. Their architecture offers a well understood and predictable behavior. For this reason, it is well suited for the aviation application. The detriment of PID control is that its application is mostly constrained by its use on a linear plant and most aerospace applications are non-linear and time varying. An aircraft's control authority that increases proportionally to dynamic pressure is one example of aerodynamic non-linear control behavior. In this case, the PID controller's robustness to changes in velocity and/or density altitude is not guaranteed and for most aircraft has to be delicately handled with lookup tables produced from hours of flight test.

Tuning the six to ten conventional PID controllers for one airframe is not a trivial task. Swarming systems often will utilize the same airframe assembled by the same manufacturer, and all aircraft still require a tedious quality assurance check. Physical aspects of the airframes such as Center of Gravity (CG), control surface deflection/calibration/speed, and airframe alignment all can drastically vary within the same delivered batch of airframes. Additionally, these miniature UAS's experience hard landings, crashes, and/or damage in transportation, which all can affect aerodynamic handling qualities. In summary, conven-

tional control laws require a moderate to high level of expertise and require significant man-hours to tune properly for every airframe even if identical.

1.2 Adaptive Control History

Adaptive control saw its early debut in the NASA North American X-15 hypersonic rocket-powered X-plane experimental aircraft. The X-15's performance envelope exceeded Mach 6.0 and 300,000 feet [3]. Engineers realized early on that the linear controllers performed well only at one dynamic pressure, but nowhere near the entire flight envelope. Scheduling the controller gains with respect to dynamic pressure (gain scheduling) was one method used to help ensure robustness; the method is still widespread in commercial aviation due to its robustness but requires a significant effort to 'explore' the entire flight envelope. These non-trivial efforts were what encouraged the exploration of the benefits of adaptive control.

The X-15 program started in 1959 and continued to 1968 flying nearly 200 successful flights. It was considered one of NASA's most successful programs. The benefit of adaptive control to the X-15 was that the adaptive controller was supposed to adjust the gain parameters online automatically. If the controller was self-tuning, it could potentially offer increased performance while reducing complexity. The Honeywell implemented the MH-96 adaptive controller in the X-15-3 as a fly-by-wire controller designed to adaptively adjust the damping in pitch and roll with respect to the desired model response. The goal was to achieve consistent aircraft response regardless of dynamic pressure and other variables. During test flights of the MH-96 adaptive control, increased performance was observed especially in the dynamic phases of reentry over that of the linear fixed gain damping system [4]. These early breakthroughs in adaptive control proved the benefits could be viable aerospace solutions. However, on November 15, 1967, there was a fatal accident caused by the adaptive controller. The adaptive controller created an out of control flight situation resulting in dynamic pressures exceeding the structural limits and subsequent breakup of the airframe at 65,000 feet.

The turbulent start of adaptive control, as implemented on the X-15 program, was due in large part to the early naive understanding of robustness. Contemporary robust adaptive control strives to encapsulate these deficiencies of robustness in studies and proofs using Lyapunov stability analysis. In addition to the developments of rigorous stability analysis

tools, a number of unique techniques have also been implemented to increase controller robustness. One such technique utilizes dead band limits on the model adaptation process to avoid system/measurement noise from causing the un-learning of the states [5]. The $\mathcal{L}_1$ adaptive control algorithm utilizes a technique which seeks to decouple the adaptation rate from robustness by 'low pass filtering' the contribution of the fast estimator under the premise that estimating the entire frequency spectrum is overly ambitious and should be limited to the bandwidth of the actuator. Many advances have been made in the adaptive control field over the past few decades, and this research sets out to evaluate a small subset of these techniques in the unforgiving aerospace environment.

CHAPTER 2:
Background and Preliminaries

2.1 Classical Feedback vs Adaptive Control

Control of a system can be broken into two required elements. There is the requirement to control the system from:

1. disturbances which affect the controlled states
2. disturbances which affect the performance of the system as a whole

Classical feedback control seeks to solve the first type of disturbance. This form of control is meticulously tuned to achieve the desired overshoot and settling time for example. The important assumption that is made by classical feedback controllers is that the underlying plant/system performance is not changing. For example, the cruise control that maintains a vehicle's speed assumes that the available horsepower of the car is fixed. This is a fairly good assumption as the horsepower with respect to rpm available at sea level and 5,000 feet for an internal combustion engine is constant enough that a fixed gain feedback controller would perform well at maintaining the speed of the vehicle in both environments. In the case of an airplane, the dynamic pressure is proportional to velocity squared and can drastically change the performance of the aircraft. In this case, the constant system performance assumption can cause a fixed gain classical feedback controller to go unstable at higher dynamic pressures (higher airspeeds). Conversely, adaptive controllers assume that the system performance is unknown and is likely to vary with time. Adaptive control seeks to ensure a system's performance with respect to characteristics, such as damping ratio and settling time, which are kept constant regardless of a plant's dynamics that may be unknown and time varying. For both classical and adaptive control, there exists some form of error which drives the controller. In the case of classical feedback, the error is calculated between the command and the feedback state of the plant. In adaptive control (in general), the error is calculated between the outputs of the desired reference model and real plant's measured performance.

Figure 2.1 outlines the decision making process a controls engineer makes when deciding

the type of controller needed for a given circumstance.

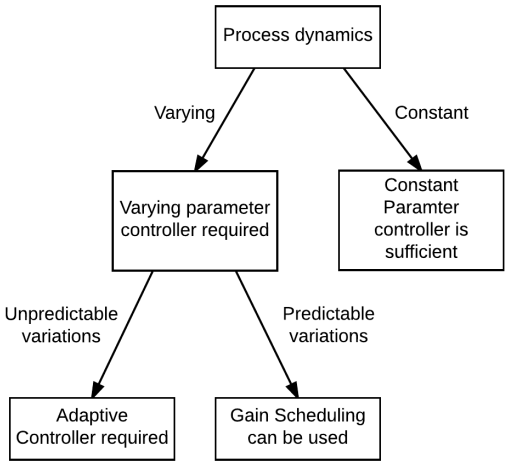

Figure 2.1. Determine if adaptive control should be used

2.2 Model Reference Adaptive Control

Model Reference Adaptive Control (MRAC) establishes the foundation for most of modern, robust adaptive control. Its structure is intuitive in nature and seeks to define a system's response to a command signal with a reference model. Unlike traditional feedback where the error signal is generated with respect to state error, MRAC attempts to achieve a system response with respect to some reference model performance. MRAC assumes that there is some nominal response of the system which can be characterized with a model of unknown parameters. The error between the model response and the system response generates the error for an 'adjustment mechanism' to learn the unknown model parameters.

Figure 2.2 illustrates a topology where a traditional feedback controller is established as an inner loop and the 'Reference Model' and 'Adjustment Mechanism' is established as an outer loop. The outer loop attempts to minimize the error between the reference model output and the plant output. Using this error to learn the system parameters can be done utilizing one of two methods: gradient descent or stability theory.

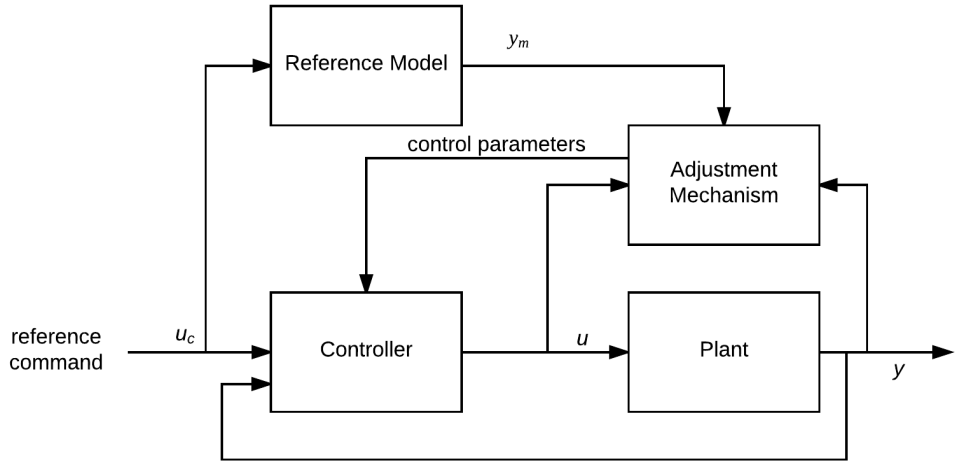

Figure 2.2. Traditional MRAC architecture

2.2.1 MIT Rule - Gradient Decent

One of the first approaches to MRAC controllers was implemented at the Instrument Labs at MIT (now known as Draper Labs). The gradient descent based method was called the 'MIT Rule' for this reason [6]. This method attempts to learn some unknown parameter by descending the gradient of the error between the reference model and the plant output.

Given the simple first order system $G(s)$:

$$G(s) = k_{dc}\frac{1}{s+1} \tag{2.1}$$

where k_{dc} is some unknown feedforward gain. In the case of the MIT rule, k_{dc} is the parameter to be learned and is defined as θ. The first step in the MIT rule is to establish a cost (or loss) function. One example of a cost function $J(\theta)$ is:

$$J(\theta) = \frac{1}{2}e^2 \tag{2.2}$$

$$e = y - y_m \tag{2.3}$$

where e is error, y is the plant output, and y_m is the model output.

In order for the cost function to be minimized, the negative gradient of the cost function is calculated and used to correct the a priori estimate. This method takes the following form where γ is the adaptation gain:

$$\frac{d\theta}{dt} = -\gamma \frac{\partial J}{\partial \theta} = -\gamma e \frac{\partial e}{\partial \theta} \tag{2.4}$$

The stability of this method is very system dependent and heavily relies on trial and error to ensure the adaptation gain (γ) is not too high. This usually requires low adaptation rates for most systems and may not produce adequate results. It should also be noted that this method presupposes there is adequate persistence of excitation. Without a frequency rich error signal being generated by adequate persistence of excitation, the model will fail to adapt. This method also offers no guarantees that the learned parameters will converge to their actual values.

2.2.2 Lyapunov Stability Criteria

Aerospace controllers tend to use linear controllers for their simplicity and well-understood robustness characteristics. This is despite the fact that the applications of these linear controllers are applied to a non-linear dynamical system such as attitude control with varying dynamic pressure. Adaptive Controllers are non-linear and may offer performance benefits to non-linear systems as seen in aforementioned aerospace applications. However, non-linear controller's robustness properties need to be evaluated. The Lyapunov stability criteria offer methods of evaluating these controller's boundedness and robustness behavior.

Aleksandr Lyapunov was a Russian mathematician who's work was published in 1892 [7] concerning the behavior of non-linear systems close to equilibrium without having to rigorously find the unique solutions to difficult differential equations used to model the system. His work was largely overlooked until the Cold War when aerospace solutions required a more rigorous approach to analyzing non-linear control robustness. Modern non-linear control engineers extensively utilize Lyapunov's techniques to design and evaluate non-linear controllers.

Lyapunov Stability Definitions

Lyapunov's methods attempt to evaluate autonomous nonlinear dynamical systems within the bounds of three classifications. In this case, the autonomous system is defined as definable set of ordinary differential equations which are not explicitly dependent upon the independent variable. These classifications can be used to define a nonlinear system as Lyapunov stable, asymptotically stable, or exponentially stable.

Given the following autonomous nonlinear dynamical system:

$$\dot{x}(t) = f(x(t)), \qquad x(0) = x_0 \tag{2.5}$$

where f has equilibrium at x_e :

$$f(x_e) = 0 \tag{2.6}$$

then the equilibrium is said to be:

1. **Lyapunov Stable**
   for every $\epsilon > 0$ there exists a $\delta > 0$ such that, if $\|x(0) - x_e\| < \delta$, then for every $t \geq 0$ we have $\|x(t) - x_e\| < \epsilon$
2. **Asymptotically Stable**
   if the system is Lyapunov stable and there exists a $\delta > 0$ such that if $\|x(0) - x_e\| < \delta$, then $\lim_{t\to\infty} \|x(t) - x_e\| = 0$
3. **Exponentially Stable**
   if the system is asymptotically stable and there exists $\alpha > 0, \beta > 0, \delta > 0$ such that if $\|x(0) - x_e\| < \delta$, then $\|x(t) - x_e\| \leq \alpha \|x(0) - x_e\| \, e^{-\beta t}$, for all $t \geq 0$

Being Lyapunov stable infers that if a system is near equilibrium, then it will indefinitely remain near equilibrium. If the system if found to be asymptotically stable then it eventually will achieve equilibrium as $t \to \infty$ and being exponentially stable implies it reaches equilibrium even faster.

Lyapunov's Second Method

Lyapunov's second proposed method is also known as Lyapunov stability criteria. This method offers a less tenuous method for evaluating mathematically non-ideal systems. Lyapunov analysis of the linearized system around equilibrium can be cumbersome in the case where equilibrium is at the origin, or the eigenvalues are purely imaginary. In this case, the solutions can rapidly depart to infinity or approach zero with little perturbation to the eigenvalues. Lyapunov's second method offers an alternative approach for classifying a system's stability using a concept that is similar to how energy is defined in classical dynamics.

Conceptually, Lyapunov's second method can be compared to evaluating the energy of a vibrating spring mass system. The energy of the unforced spring mass system will dissipate energy due to friction and or damping etc. This trend of energy leaving the system towards some 'attractor' is evidence of the system's stability characteristics and identifies that there will be some stable end state. Likewise, Lyapunov's second method characterizes this with the use of a Lyapunov candidate function $V(x)$. It is important to note that Lyapunov realized that the candidate function could be any function so as long as one candidate function is found in agreement with the stability criteria. It is then said to be Lyapunov stable if any candidate equation is found and meets the stability criteria. This means that it is only incumbent upon the engineer to find one candidate equation to meet the criteria. This can be an iterative process of trying various energy like equations. A common approach is to model the Lyapunov candidate equation as kinetic energy ($\frac{1}{2}u^2$). Lyapunov realized that characterizing the energy of a nonlinear system could be almost impossible for some cases, but this approach could prove stability without the rigorous knowledge of the true system's energy.

Lyapunov's second method defines a system as Lyapunov Stable for a system $\dot{x} = f(x)$ having an equilibrium point at $x = 0$ where the Lyapunov candidate function $V(x) : \mathbb{R}^n \to \mathbb{R}$ such that:

- $V(x) = 0$ if and only if $x = 0$
- $V(x) > 0$ if and only if $x \neq 0$
- $\dot{V}(x) = \frac{d}{dt}V(x) = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} f_i(x) \leq 0$, for all values of $x \neq 0$

if $\dot{V}(x) < 0$ for $x \neq 0$ then system is asymptotically stable.

Additionally, it is required to demonstrate the condition of radial unboundedness to ensure the system is globally stable.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
$\mathcal{L}_1$ Adaptive Control Derivation

Introduction here!!!

3.1 $\mathcal{L}_1$ Adaptive Control

The $\mathcal{L}_1$ adaptive controller is an evolution of the concepts implemented by MRAC. They are similar approaches designed to model a Linear Time Invariant (LTI) system with unknown constant parameters. These parameters are adjusted to achieve the desired outcome of the error between the actual plant (system) and the referenced system model (state predictor) to asymptotically approach zero. Adaptive control attempts to estimate the plant's unknown parameters in situ. Parameter estimation is done using either direct or indirect architecture. The indirect architecture attempts to estimate the system's parameters, which could be considered similar to system identification. Alternately the easier to implement direct architecture estimates the controller parameters explicitly. These architectures can be seen below in Figures 3.1 and 3.2.

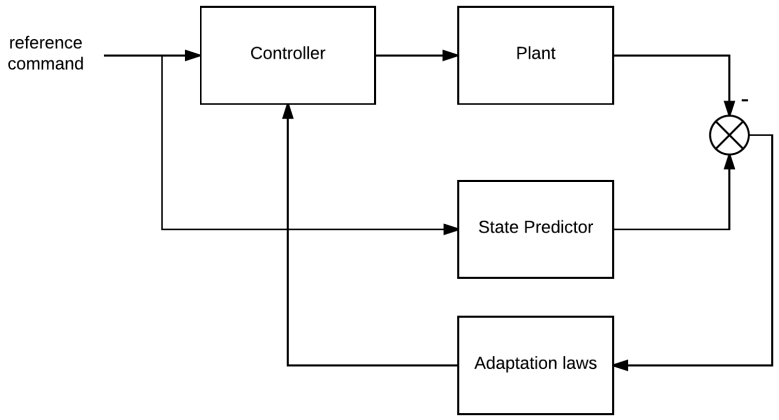

Figure 3.1. Direct MRAC architecture

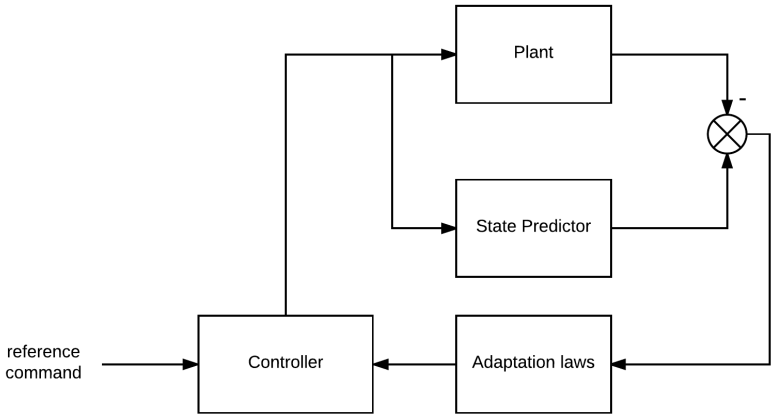

Figure 3.2. Indirect MRAC architecture

3.1.1 Reference Model versus Companion Model

Traditional MRAC controllers often refer to the system objective function as the 'Reference Model.' In this case, the engineer designs a reference model response, and it is from this model response that the error state is calculated directly. Because the $\mathcal{L}_1$ adaptive control implements the use of a low-pass filter in conjunction with a model objective function, the model is often referred to as the 'Companion Model.' This subtle distinction is necessary because the engineer must be aware that the system response will be the with respect to the low-pass filter and the companion model in series. In other words, the plant will not mimic the companion model; it will mimic the companion model plus the low-pass filter.

3.1.2 $\mathcal{L}_1$ Architecture

The $\mathcal{L}_1$ adaptive control algorithm asserts that trying to estimate the plant uncertainties outside of the control actuators' bandwidth is overly ambitious. The system's actuator bandwidth and the slow dynamics of the plant are most commonly the system's limiting factors, and the estimator's robustness/stability could be in question if unmodeled high-frequency content exists in the plant. The $\mathcal{L}_1$ adaptive control constrains the objective function by using a low-pass filter (first or second order) to band the frequency response to meet robustness specifications. This low-pass filter should be tuned to a frequency response commensurate with the actuator's frequency response. Through inspection the low-pass filter placement in the controller topology, it becomes clear that the indirect architecture is

the only candidate as detailed below. Figures 3.4 and 3.5 illustrate the placement of the low-pass filter and its implication on the closed loop model.

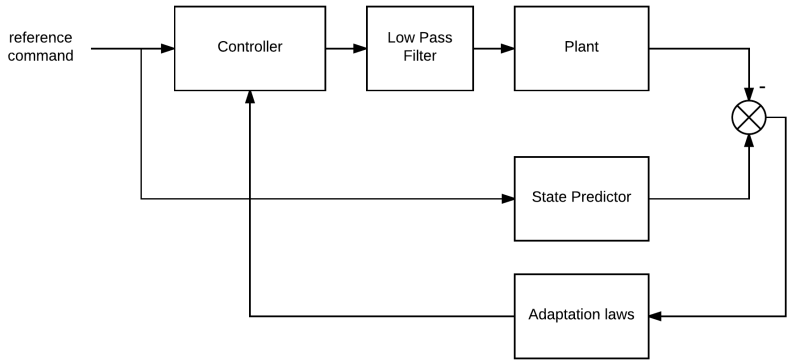

Figure 3.3. Direct MRAC architecture with low-pass filter

It can be seen in figure 3.4 that the direct architecture implementation of the low-pass filter introduces difficulties. The low-pass filter cascades with the plant and effectively creates a new plant that is simply the plant plus a low-pass filter. The aerodynamic and Lyapunov stability underpinnings become non-sensical in this implementation and are non-subtractable when the error state is calculated.

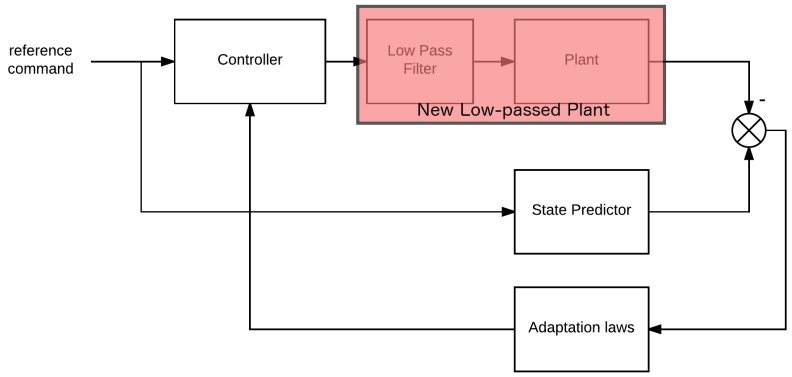

Figure 3.4. Non-Subtractable Lowpass Implementation - Direct Architecture

Conversly, the indirect approach offers an implementation which ensures the low-pass filter is applied to both the companion model (state predictor) and the plant.

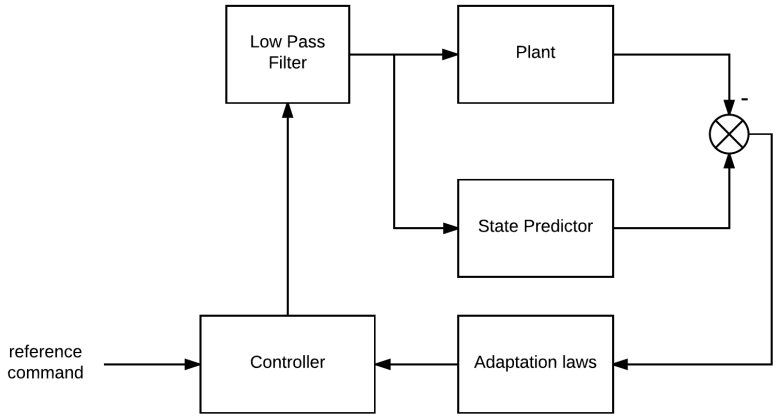

Figure 3.5. Indirect MRAC architecture with low-pass filter

It can be seen that the low-pass filter in the direct architecture inherently changes the structure of the model with the cascading of the low-pass filter and plant block diagrams. This change mathematically is not mirrored in the companion model (state predictor) and therefore is not subtractable. However, in the indirect case, the structure of the model is kept intact, and the low-pass filter is applied to both the plant and the state predictor. This ensures that the low-pass filter is subtractable when calculating the error state and the model's structure is kept intact.

Many slight variations of the $\mathcal{L}_1$ adaptive architectures have been derived for various use cases [8]. Some of the following forms were studied for viability in the fixed wing UAS use case:

- Single Input Single Output (SISO) with constant but unknown state parameters
- SISO with time variant and/or nonlinear unknown state parameters
- Multiple Input Multiple Output (MIMO) with constant but unknown state parameters
- MIMO with time variant and/or nonlinear unknown state parameters

MIMO control algorithms would potentially afford the controller more ability to cope with system coupling if present. A fixed wing UAS would exhibit coupled behavior due to the coupling present in the aerodynamics as seen in equation B.11, but was not chosen due to the added architectural complexity. Unknown state parameters that are assumed to be constant or time invariant are considered matched uncertainty. Unknown state parameters that are

non-constant (time variant) and/or exhibit non-linear behavior are considered unmatched uncertainty. The unmatched uncertainty architecture offers a more appealing solution for fixed wing use cases (asymmetric actuator failure, aerodynamic coefficients scaled by dynamic pressure, etc.), but adds a significant amount of complexity to the architecture. In summary, the SISO architecture with matched uncertainty was chosen for this research.

The SISO controller with matched uncertainty was chosen to control pitch rate (q) and roll rate (p) of the aircraft using two separate, but parallel controllers. This meant that the controller could be generalized to a 'first principles' physical point mass model similar to derivations found in rigid body equations of motion. In this implementation of the $\mathcal{L}_1$ adaptive controller, the desired state x to be controlled was an individual body rate (e.g., q, p).

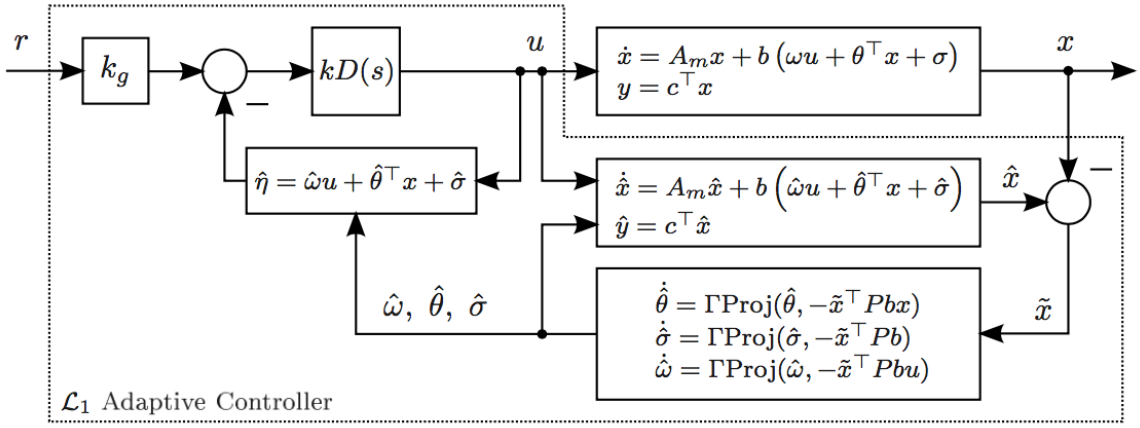

Figure 3.6. $\mathcal{L}_1$ Architecture with Matched Uncertainty Block Diagram [8]

As seen in Figure 3.6, the generalized $\mathcal{L}_1$ architecture in block diagram form and the following elements can be identified:

k_g - feed forward input gain
k - feedback gain
$D(s)$ - user described filter (second order low pass plus integrator)
$\hat{\eta}$ - $\mathcal{L}_1$ controller state
$\dot{x}$ - first order differential equation of state model
$\hat{x}$ - state estimate
$\tilde{x}$ - state error

u - controller output
r - reference objective
A_m - Hurwitz matrix
b - input matrix
$\hat{\omega}$ - unknown input gain coefficient
$\hat{\theta}$ - unknown constant state coefficient
$\hat{\sigma}$ - unknown disturbance estimate
Γ - adaptation gain
Pb - solution to the Lyapunov stability criterion

It should also be noted that the architecture presented in Figure 3.6 includes the use of a projection operator. The parameters for $\dot{\hat{\omega}}$, $\dot{\hat{\theta}}$, and $\dot{\hat{\sigma}}$ are all projection based adaptation laws. This ensures that the adaptation stays bounded around the feasible region of parameter space. The Lyapunov stability proofs for this architecture rely on this method to guarantee stability [8]. More discussion on the specific application of this operator can be found in Appendix D.

One of the main benefits of using the SISO architecture is that the solution to the Lyapunov stability criterion (Pb) used in the projection based adaptation laws is greatly simplified.

In this case, Pb reduces to:

$$Pb = \frac{1}{2\omega_n} \tag{3.1}$$

where ω_n is the natural frequency in rad/s for the first order companion model in discrete recursive form assuming DC gain of 1. The proof for this can be found in Appendix E.4.

3.2 $\mathcal{L}_1$ Parameter Estimation

Three Parameters are estimated in this research with increasing complexity both in architecture and in required stability proofs. The following outlines the nomenclature for the estimation of a system with unknown constant parameters, input/output disturbances, and an unknown system input gain.

The $\mathcal{L}_1$ adaptive control algorithm is primarily used to estimate unknown constant system

parameters. These system parameters are defined as θ as seen in the following model:

$$\dot{x}(t) = Ax(t) + b(u(t) + \theta^{\top}(t)x(t)) \tag{3.2}$$

The second adaptive element is the estimation of the input/output disturbances (σ). This additional parameter is implemented in this research as any unmodeled transient dynamics such as aerodynamic/inertial coupling. This model takes the form:

$$\dot{x}(t) = Ax(t) + b(u(t) + \theta^{\top}(t)x(t) + \sigma(t)) \tag{3.3}$$

The last adaptive element used for this research was the estimation of unknown system input gain (ω). Estimating the unknown system input gain offers the controller the ability to estimate the actuator effectiveness for changes in dynamic pressure or failed control surfaces. This model takes the form:

$$\dot{x}(t) = Ax(t) + b(\omega(t)u(t) + \theta^{\top}(t)x(t) + \sigma(t)) \tag{3.4}$$

The above model in equation 3.4 can be paralleled to the aircraft model derived in equation B.14 as follows for the roll rate model example below:

$$\dot{\hat{p}} = A_p\hat{p} + b_p\left(\hat{\omega}_p\delta_a + \hat{\theta}_p p + \hat{\sigma}_p\right) \tag{3.5}$$

This final model, with the inclusion of all three estimated parameters ($\hat{\omega}, \hat{\theta}, \hat{\sigma}$), established the architecture tested in this research. The fundamental assumption that the aerodynamic and inertial coupling was negligible and set to zero as outlined in equation B.13 is agreeably a gross assumption which may prove to be inadequate for stable flight. These assumptions presume that the estimated input/output disturbance ($\hat{\sigma}$) would be adequate to compensate for these unmodeled dynamics and were validated by successful flight test in section 5.2.

These derivations of the $\mathcal{L}_1$ algorithm guarantees that the error between the model and plant asymptotically approaches zero, but this does not imply the constraint that the estimated parameters are in fact converging to their real values. The algorithm only guarantees that

the parameters are bounded and therefore it is common to observe the parameters never reaching steady-state. The engineer must ensure that the bounds set on the parameters are sufficient for the controlled system to not become unstable. In the discrete form, the algorithm can have numerical floating point instability if these projection operator bounds are too high and not throughly tested in simulation.

3.3 $\mathcal{L}_1$ Filter - $C(s)$

One of the key features of the $\mathcal{L}_1$ adaptive controller is that the robustness of the controller is decoupled from the adaptation rate. This is handled in the filter section of the $\mathcal{L}_1$ architecture. One of the elements that the guaranteed stability depends upon is that $C(s)$ is strictly-proper stable. With the architecture that includes the system input gain (ω), one cannot simply apply a stand alone filter. The inclusion of ω in the architecture block diagram in figure 3.6 slightly modifies the signal output and takes the following form:

$$u(s) = -kD(s)(\hat{\eta}(s) - k_g r(s)) \tag{3.6}$$

where $D(s)$ is the new user defined filter and $C(s)$ now takes the form:

$$\begin{aligned} \omega \in \Omega_0 &\triangleq [\omega_{l_0}, \omega_{u_0}] \\ C(s) &= \frac{\omega k D(s)}{1 + \omega k D(s)} \end{aligned} \tag{3.7}$$

In the case where the user defined function $D(s)$ is a simple integrator, $C(s)$ takes the form:

$$\begin{aligned} D(s) &= \frac{1}{s} \\ C(s) &= \frac{\omega k}{s + \omega k} \end{aligned} \tag{3.8}$$

The feedback gain k should not be assumed to be 1 in this case. As seen in equation 3.8, the $C(s)$ transfer function is a function of k that can have significant influence on the controller output. In actual implementation, k was found to be one of the most influential gains in the architecture and extremely critical in specifying the transition between robustness and performance.

3.4 $\mathcal{L}_1$ Discrete Time Implementation

Implementing any algorithm on actual autopilot hardware will inevitably force some if not all parts of the algorithm to be discretized. Autopilots like the Pixhawk operate at some scheduled loop rate for executing the litany of subprograms that measure sensors, calculate navigation commands, and much more. In the case of the Pixhawk autopilot, the main loop can run up to 400 Hz. At 400 Hz, there is a significant insurance that the vehicle's full frequency domain of importance will be achievable. However, the ArduPilotMega / Multi-Platform Autopilot (APM) flight stack records all logged parameters also at this loop rate and can create log files larger than are reasonably desired. There are a myriad of other reasons why the engineer would not want to run at high loop rates, but successful flight at the lowest (default of 50Hz) is desired if adequate performance of the adaptive control can be achieved. Failures in early adaptive control were largely in part due to a very naive understanding of robustness. Brian Anderson concludes that "it is clear that the identification time scale needs to be faster than the plant variation time scale, else identification cannot keep up" [9].

The $\mathcal{L}_1$ architecture does not require the algorithm to run in sync with the sensor measurements etc. Ideally, the autopilot main loop rate would remain at its default value (50Hz), and the adaptation section of the $\mathcal{L}_1$ controller would run as fast as the CPU would allow. However, the APM architecture does not lend itself well to this scheme in its current configuration without significant modification to the code base. Therefore, the initial tests presumed that the $\mathcal{L}_1$ algorithm is running at the same frequency as the main loop of the autopilot. The fundamental derivation of the $\mathcal{L}_1$ algorithm proves that performance and adaptation gain are maximized as the loop time (dt) approaches zero. Increased performance would come from higher adaptation loop rates, but the primary focus was to implement the $\mathcal{L}_1$ algorithm with minimal or no detrimental effects to the current APM codebase.

3.4.1 Digital Bi-Quad Filter

The $\mathcal{L}_1$ adaptive control algorithm utilizes two specific elements that will require careful discretization; the companion model and the low pass filter. The digital bi-quad filter offers a very versatile and straightforward method for accurately implementing the companion model and the low pass filter discretely using its recursive nature. It is a second order filter which uses a Finite Impulse Response (FIR) front end and an Infinite Impulse Response

(IIR) back end requiring four total memory blocks. This topology allows the designer to create numerous types of filters (low pass, high pass, bandpass, etc.) simply by choosing appropriate coefficients. If a first order filter is needed, then the higher order FIR/IIR terms can be set to zero. Figure 3.7 illustrates this filter's topology where the FIR structure is the left two memory blocks and the IIR structure is the right two memory blocks.

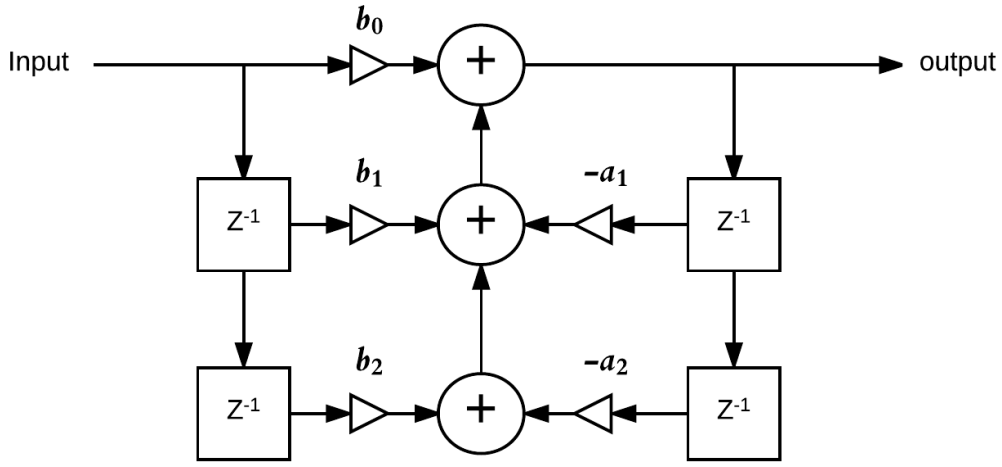

Figure 3.7. Digital Bi-quad Filter Architecture

A bilinear Z transform is used to convert the desired S-domain (continuous time domain) filter/model into the Z-domain (discrete time domain) to determine the structure of the coefficients.

This derivation can be seen below for the second order low pass model:

$$H(s) = \frac{1}{s^2 + \frac{s}{Q} + 1} \tag{3.9}$$

where the bi-linear transform converts s to z via:

$$s = \left(\frac{1}{K}\right)\left(\frac{z-1}{z+1}\right) \tag{3.10}$$

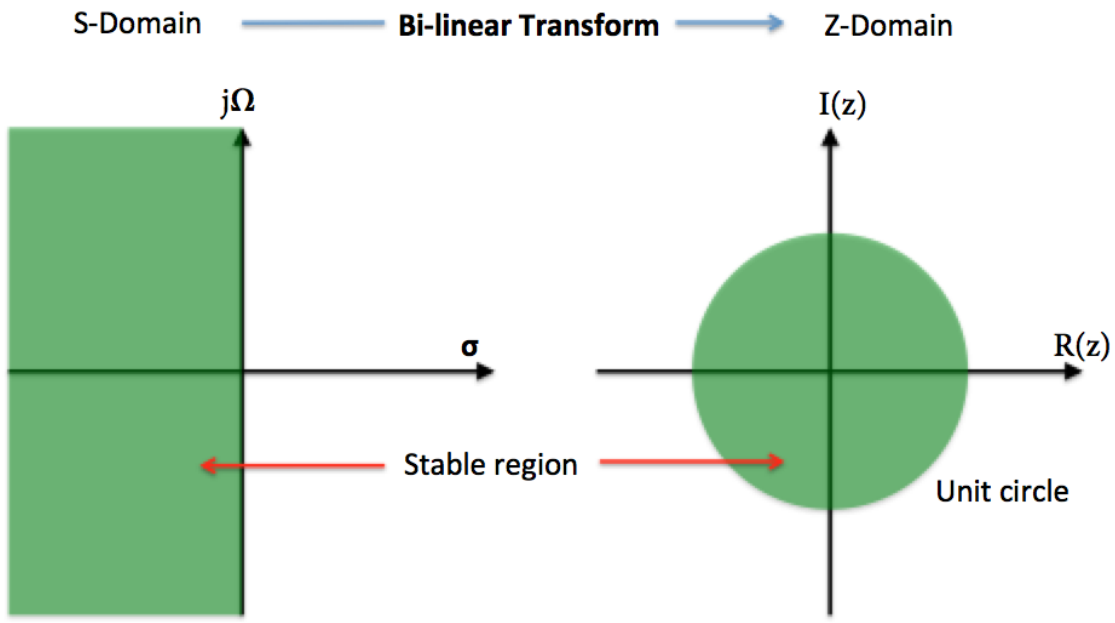

Figure 3.8. Bi-linear Transform

K is the 'pre-warping' factor which accounts for the transition of the vertical s-plane into the circular z-plane as seen in figure 3.8.

where ωT is:

$$\omega T = 2\pi \left(\frac{F_c}{F_s} \right) \tag{3.11}$$

$$\begin{aligned} K &= tan \left(\frac{\omega T}{2} \right) \\ &= tan \left(\pi \frac{F_c}{F_s} \right) \end{aligned} \tag{3.12}$$

F_c is the desired corner frequency of the filter and F_s is the sampling rate (or loop rate of the autopilot). This 'pre-warping' is critical to ensure that the continuous time cutoff frequency desired is correctly established in the discrete implementation. It is the engineer's discretion

if pre-warping is required for the appropriate application, but the general guidance is to pre-warp the Z-domain coefficients if the desired cut-off frequency is close to Nyquist. It was chosen for this application to always pre-warp the coefficients even though the error is small for corner frequencies which are fairly distant from Nyquist. Continuous calculation of the pre-warp coefficient was chosen because calculating the $tan()$ function real time on the CPU adds negligible computational strain but offers ease of tuning for the engineer.

Applying the bi-linear transform to the continuous time second order low pass filter results in:

$$H(z) = \frac{1}{\left[\left(\frac{1}{K}\right)\left(\frac{z-1}{z+1}\right)\right]^2 + \frac{\left(\frac{1}{K}\right)\left(\frac{z-1}{z+1}\right)}{Q} + 1} \tag{3.13}$$

The desired form is:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} \tag{3.14}$$

Reducing equation 3.13 to match the form in equation 3.14 results in the following coefficients:

$$\begin{aligned} a_0 &= 1 \\ a_1 &= \frac{2(K^2 - 1)}{K^2 + \frac{K}{Q} + 1} \\ a_2 &= \frac{K^2 - \frac{K}{Q} + 1}{K^2 + \frac{K}{Q} + 1} \\ b_0 &= \frac{K^2}{K^2 + \frac{K}{Q} + 1} \\ b_1 &= 2b_0 \\ b_2 &= b_0 \end{aligned} \tag{3.15}$$

The bandwidth of the filter Q can be set by the engineer. For example, if the pass-band of the filter is desired to be flat (Butterworth) then Q can be set equal to $\frac{1}{\sqrt{2}}$. For this research, the following C++ code segments were used to explicitly calculate the bi-quad low-pass filter implementation:

```
void DigitalBiquadFilter<T>::compute_params(float sample_freq,
float cutoff_freq, biquad_params &ret) {
    ret.cutoff_freq = cutoff_freq;
    ret.sample_freq = sample_freq;

    float fr = sample_freq/cutoff_freq;
    float K = tanf(M_PI/fr);  //Pre-Warp calculation
    float c = 1.0f+2.0f*cosf(M_PI/4.0f)*K + K*K;

    ret.b0 = K*K/c;
    ret.b1 = 2.0f*ret.b0;
    ret.b2 = ret.b0;
    ret.a1 = 2.0f*(K*K-1.0f)/c;
    ret.a2 = (1.0f-2.0f*cosf(M_PI/4.0f)*K+K*K)/c;
}
```

```
T DigitalBiquadFilter<T>::apply(const T &sample,
const struct biquad_params &params) {

    T delay_element_0 = sample - _delay_element_1 * params.a1
     - _delay_element_2 * params.a2;

    T output = delay_element_0 * params.b0
     + _delay_element_1 * params.b1
     + _delay_element_2 * params.b2;

    _delay_element_2 = _delay_element_1;
    _delay_element_1 = delay_element_0;

    return output;
}
```

This implementation can be used as the $\mathcal{L}_1$ low-pass filter and as the companion model. It can be seen in the above code segment that K, the pre-warp factor, is explicitly calculated

every iteration.

3.4.2 Simplified Bi-quad First Order Model

In the case of the companion model, a first order response may be desired. As described in equations A.2 and A.4, the discrete first order model can be derived from a simplified Bi-quad as seen below in figure 3.9. It can be seen that the first coefficient of the IIR filter is kept from this topology.

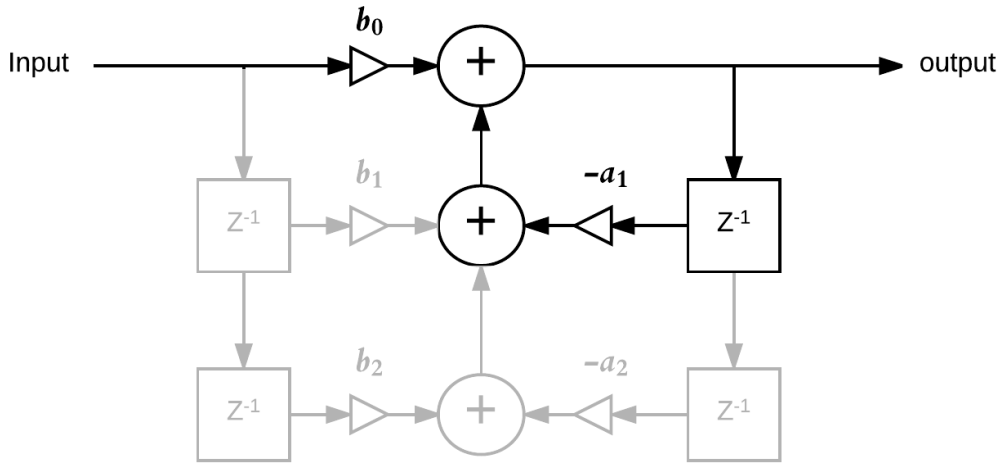

Figure 3.9. Digital Bi-quad Simplified First Order Low-pass Filter

The first order model can be specified by either its time constant (time in seconds to reach 63% of steady-state) or its -3dB corner frequency. The system takes the form as seen in equation 3.16 when defined by its corner frequency.

$$H(s) = \frac{\omega_n}{s + \omega_n} \tag{3.16}$$

therefore the explicit calculation of the Bi-quad coefficients in this case becomes:

$$
\begin{aligned}
a_1 &= e^{\left(\frac{-\omega_n}{F_s}\right)} \\
b_0 &= 1 - a_1
\end{aligned}
\tag{3.17}
$$

where ω_n is the -3dB corner frequency in radians per second and F_s is the sampling frequency in Hz.

Therefore the discrete recursive form of the first order model becomes:

$$
y_{i+1} = a_1 y_{i-1} + b_0 y_i \tag{3.18}
$$

Another form designed to optimize for speed that is commonly seen in software takes the form:

```
float b_0=exp(-f_c/f_s);
float out+=(in-out)*b_0;
```

3.4.3 Euler vs. Trapezoid Rule

The model estimate, as well as the parameter estimates for the $\mathcal{L}_1$ algorithm, are both numerically estimated using discrete integration. The Euler method is a numerical procedure for solving ordinary differential equations. The Euler method as applied to discrete integration is the fundamental method for recursively integrating a digital signal. The algorithm takes the form:
where h is the uniform step size,

$$
y_{i+1} = y_i + hf(t_i, y_i) \tag{3.19}
$$

The recursive trapezoidal method takes the form:

$$
\begin{aligned}
\tilde{y}_{i+1} &= y_i + hf(t_i, y_i) \\
y_{i+1} &= y_i + \frac{h}{2}[f(t_i, y_i) + f(t_{i+1}, \tilde{y}_{i+1})]
\end{aligned}
\tag{3.20}
$$

Comparing the accuracy of the two numerical methods for discretely calculating the integral of $y = e^t$ can be seen in figure 3.10:

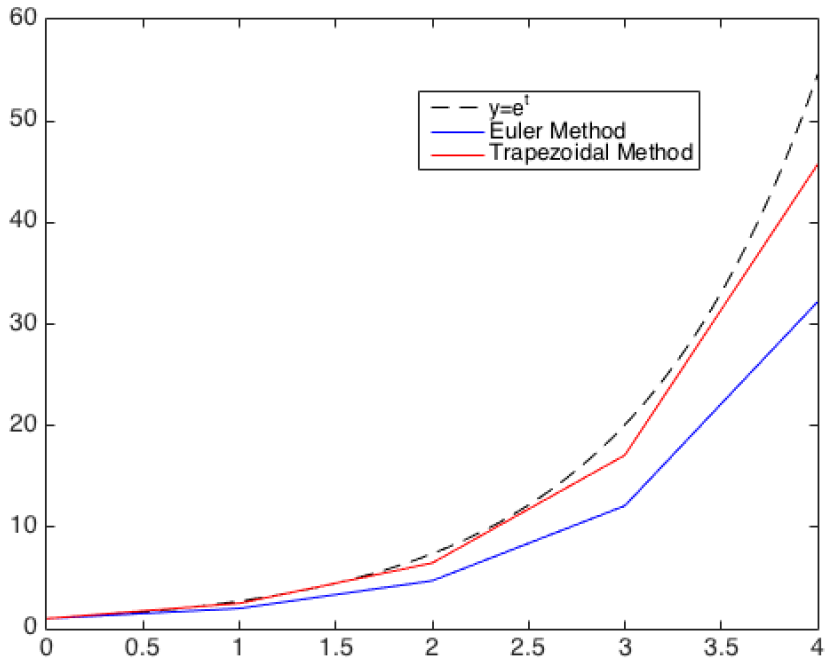

Figure 3.10. Euler vs Trapezoidal Integration error

As seen in equation 3.20, the recursive trapezoidal integration method only adds one more line of complexity to the algorithm for a significant gain in accuracy and therefore will be the chosen method applied for all discrete numerical integration in this research and takes the form:

```
float trap_integration(float y0, float y1_dot, float dt, float &y0_dot)
{
    float y1 = y0 + (dt/2)*(y0_dot+y1_dot);
    y0_dot = y1_dot;

    return y1;
}
```

CHAPTER 4:
Design of Experimental Platform

4.1 Pixhawk Autopilot

The Pixhawk autopilot is a collaborative project among open-source engineers which resulted in a high-performance autopilot which is capable of controlling aircraft, ground vehicles, and many others. The primary reason this autopilot was chosen was because of the vast amount of support in the developer community. The Pixhawk 1 autopilot hardware is effectively obsolete at the time of this writing, but the open-source code base is extremely flexible and continues to be ported to new hardware as it becomes available. This has been the case for various raspberry pi autopilots as well as the Pixhawk 2. The Pixhawk autopilot operates using two flight stacks (code base/operating systems); the PX4 flight stack and the APM flight stack. This research was implemented on the APM flight stack primarily because the author's familiarity with the developer team which offers unparalleled assistance to the academic community. The APM codebase also offers a litany of open-source tools such as a Linux based Ground Control Station (GCS), Software in the Loop (SITL) simulator, log analysis tools, and an Application Program Interface (API) for the RealFlight 7.5 simulator (high fidelity airframe simulation for small aircraft).

4.1.1 Key Features

- 168 MHz / 252 MIPS Cortex-M4F
- 14 PWM / Servo outputs (8 with failsafe and manual override, 6 auxiliary, high-power compatible)
- Abundant connectivity options for additional peripherals (UART, I2C, CAN)
- Integrated backup system for in-flight recovery and manual override with dedicated processor and stand-alone power supply (fixed-wing use)
- Backup system integrates mixing, providing consistent autopilot and manual override mixing modes (fixed wing use)
- Redundant power supply inputs and automatic failover
- External safety switch

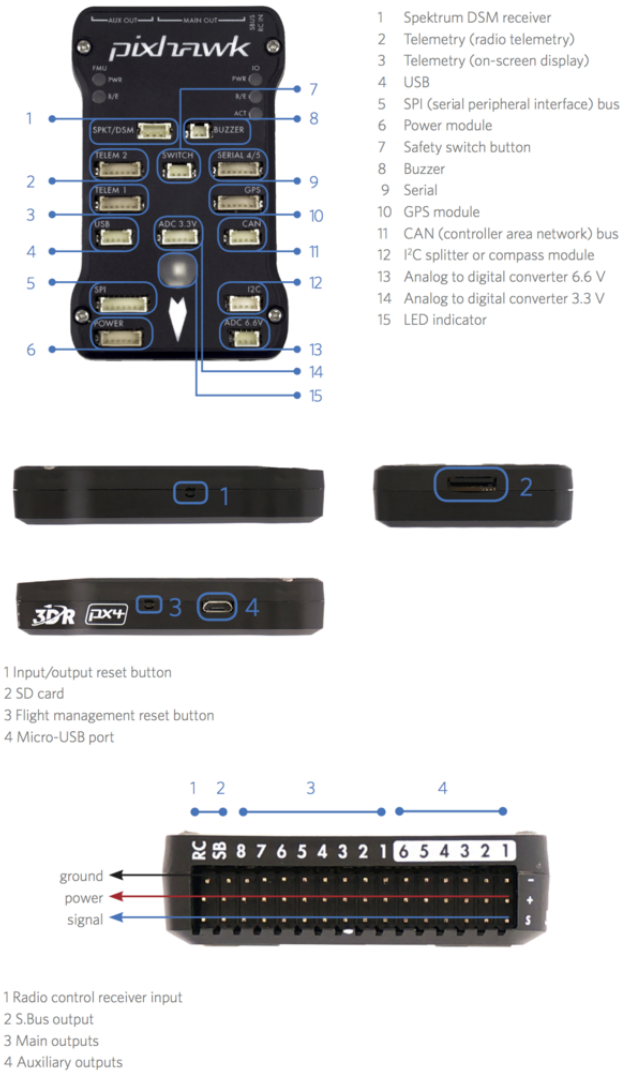

Figure 4.1. Pixhawk 1 Autopilot Connection Diagram

- Multicolor LED main visual indicator
- High-power, multi-tone piezo audio indicator
- microSD card for high-rate logging over extended periods of time

4.1.2 Specifications

Processor

- 32bit STM32F427 Cortex M4 core with FPU
- 168 MHz
- 256 KB RAM
- 2 MB Flash
- 32 bit STM32F103 failsafe co-processor

Sensors

- ST Micro L3GD20H 16 bit gyroscope
- ST Micro LSM303D 14 bit accelerometer / magnetometer
- Invensense MPU 6000 3-axis accelerometer/gyroscope
- MEAS MS5611 barometer

Interfaces

- 5x UART (serial ports), one high-power capable, 2x with HW flow control
- 2x CAN (one with internal 3.3V transceiver, one on expansion connector)
- Spektrum DSM / DSM2 / DSM-X® Satellite compatible input
- Futaba S.BUS® compatible input and output
- PPM sum signal input
- RSSI (PWM or voltage) input
- I2C
- SPI
- 3.3 and 6.6V ADC inputs
- Internal microUSB port and external microUSB port extension

4.2 Ground Control Station - GCS

The GCS used for this research was MAVproxy [10]. It is an open-source python based GCS which provides flexible communication and command with any autopilot utilizing the MAVlink protocol [11]. Even though MAVproxy is written in Python (Operating System (OS) agnostics language), it was found to be cumbersome to operate the GCS on

any other platform other than Linux. This is primarily because the source code updates quite rapidly to support new features and the core developer (Andrew Tridgell) exclusively utilizes MAVproxy in Linux. A significant amount of external libraries are utilized which results in a moderate amount of compatibility debugging for other OS's if desired.

4.2.1 MAVproxy Features

The following are summaries which proved to be extremely useful for this research [12]:

- Command-line, console based application. Plugins included in MAVProxy provide a basic Graphical User Interface (GUI).
- Can be networked and run over any number of computers.
- Portable; capable of running on any POSIX OS with Python, pyserial, and select() function calls, which means Linux, OS X, Windows, and others.
- The light-weight design; runs on small netbooks.
- Tab-completion of commands.

4.3 Simulation

The APM environment offers three versions of SITL simulations. The lowest fidelity SITL is provided by MAVproxy, which is a simple 6-degree of freedom kinematics model with no environment or actuator modeling. This proved to be adequate for initial testing but resulted in poorly tuned algorithms when actual flight tests were conducted. The MAVproxy simulator was used for basic code debugging but nothing else.

The second SITL offered in the APM environment is X-plane 10. This is a much higher fidelity simulation, which includes actuator models and environmental modeling. X-plane 10 is primarily used for simulating full-scale aircraft and therefore is difficult to find models, which accurately represent the dynamics of small fixed-wing UAS. The open-source community provided model called the "maxi-swift" was similar enough to the airframe in this research that it provided adequate SITL modeling which ensured robust flight test.

The last SITL simulation tested under the APM environment was the RealFlight 7.5 API. The RealFlight Remote Controlled (RC) airplane simulator offers some of the industry's highest fidelity simulations for small aircraft. This product requires an API key to hook into

MAVproxy. This capability is not yet on the market as of the time of this research, but the APM core developers were supportive of this research and ran multiple experiments with the RealFlight SITL for early validation.

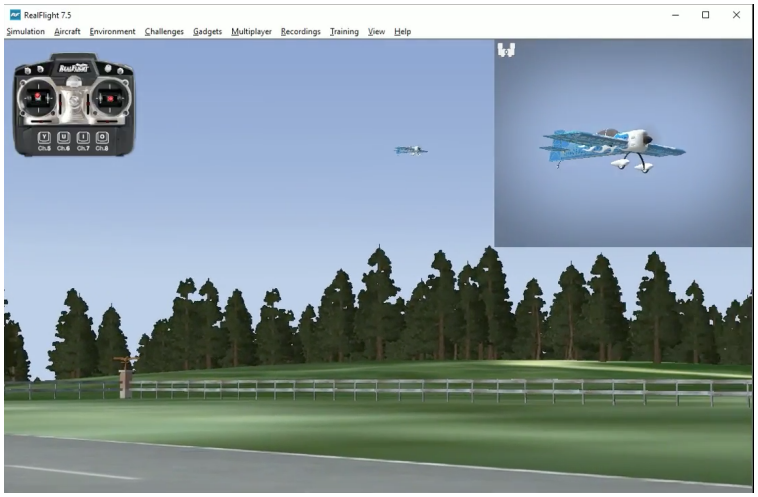

Figure 4.2. High Fidelity RealFlight 7.5 SITL

4.4 Airframe

The aircraft used for this research was the Flitetest Spear [**?**]. The Spear airframe was chosen for its endurance capability of greater than 45 minutes of flight time and its large capacity fuselage. The flying-wing architecture keeps the actuation requirement to a minimum of two servos by utilizing an elevon configuration.

Figure 4.3. Spear Airframe

The large blunt nose provides adequate space for two 2,200 mAh (12.6volts) lithium polymer

batteries wired in parallel. The remaining cargo space was used for accommodating the Pixhawk autopilot.

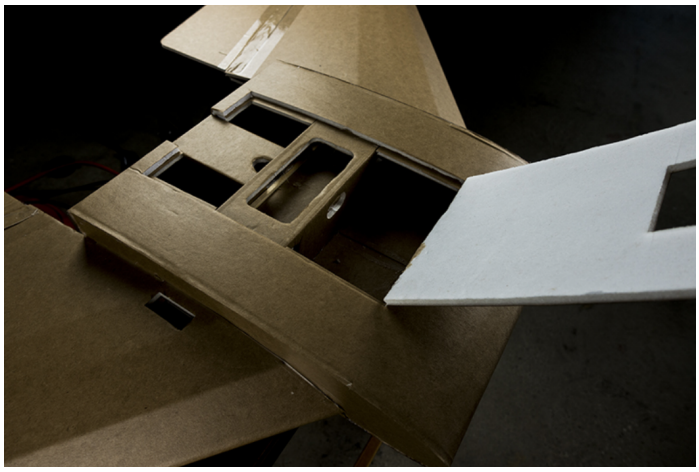

Figure 4.4. Spear Cargo Capacity

This plane was constructed out of craft foam board. The plans were downloaded from flitetest.com [**?**] and converted to CorelDraw vector files for use in a laser cutter. These files were then cut out of four sheets of foam board using the laser cutter. The wing halves were joined with standard box tape and hot glue. This provided a cheap and rapid construction process which was achievable under four hours of build time.

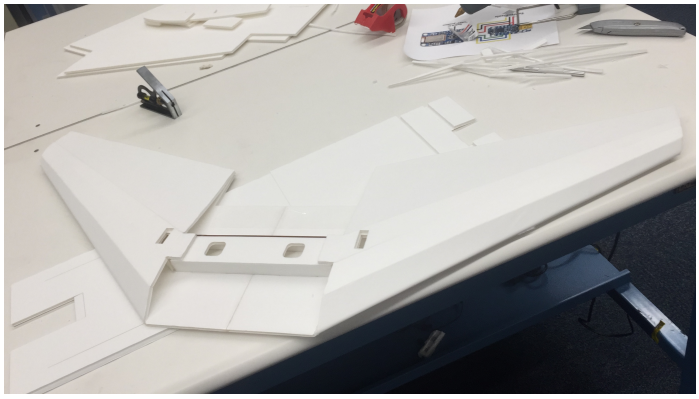

Figure 4.5. Spear Build Process

4.4.1 Specifications

- Weight without battery: 1.45 lbs (658 g)

- Center of gravity: 3 – 3.5” (76 – 89 mm) in front of firewall
- Control surface throws: 16°deflection – Expo 30%
- Wingspan: 41 inches (1041 mm)
- Recommended motor: 425 sized 1200 kv minimum
- Recommended prop: 9 x 4.5 CW (reverse) prop
- Recommended ESC: 30 amp minimum
- Recommended Battery: (2) 2200 mAH 12.6 volt minimum
- Recommended Servos: (2) 9 gram servos

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Flight Testing and Performance Evaluation

5.1 Simulation Results

Simulation results were captured utilizing the APM SITL using X-Plane10. The aircraft model chosen was an open-source flying wing model called the maxi-swift as seen in figure 5.1.

Figure 5.1. Maxi-Swift Flying Wing Model used in X-Plane10

This plane did not afford proper testing of elevon mixing, but the fidelity in the model provided utilizing X-Plane10 was surprisingly accurate with respect to the FT-Spear aircraft used for actual flight test. This drastically increased confidence that anything tested in SITL would have a high probability of success in actual flight test.

5.1.1 Pitch Attitude Performance

It can be seen in figure 5.2 that there exists some steady state error when under PID control. This phenomenon is more pronounced when the desired pitch attitude is negative. This steady state error is due to the increase in lift caused by the increasing airspeed. The

PID controller underperforms in this regime. However, the adaptive controller is able to compensate for these dynamics fast enough to track the desired attitude with no steady state error.

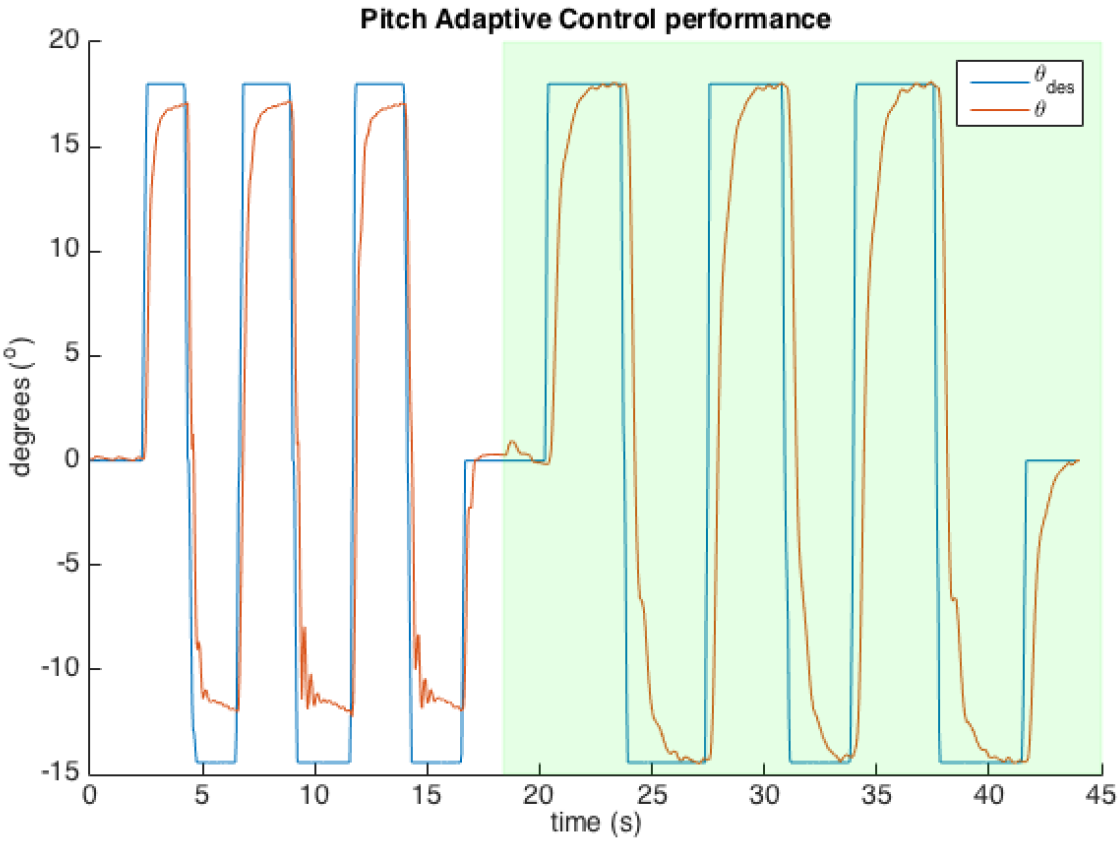

Figure 5.2. PID vs $\mathcal{L}_1$ Adaptive Control Pitch Performance

5.1.2 Roll Induced Pitch Disturbance

When rapidly rolling the maxi-swift aircraft in simulation, there was a noticeable coupling in the pitch axis. The PID controller struggles to correct this discrepancy because the time constant of the integral error simply cannot be increased high enough to achieve satisfactory compensation. As seen in figure 5.3, the $\mathcal{L}_1$ adaptive controller significantly reduced the pitching disturbance due to rapid rolling.

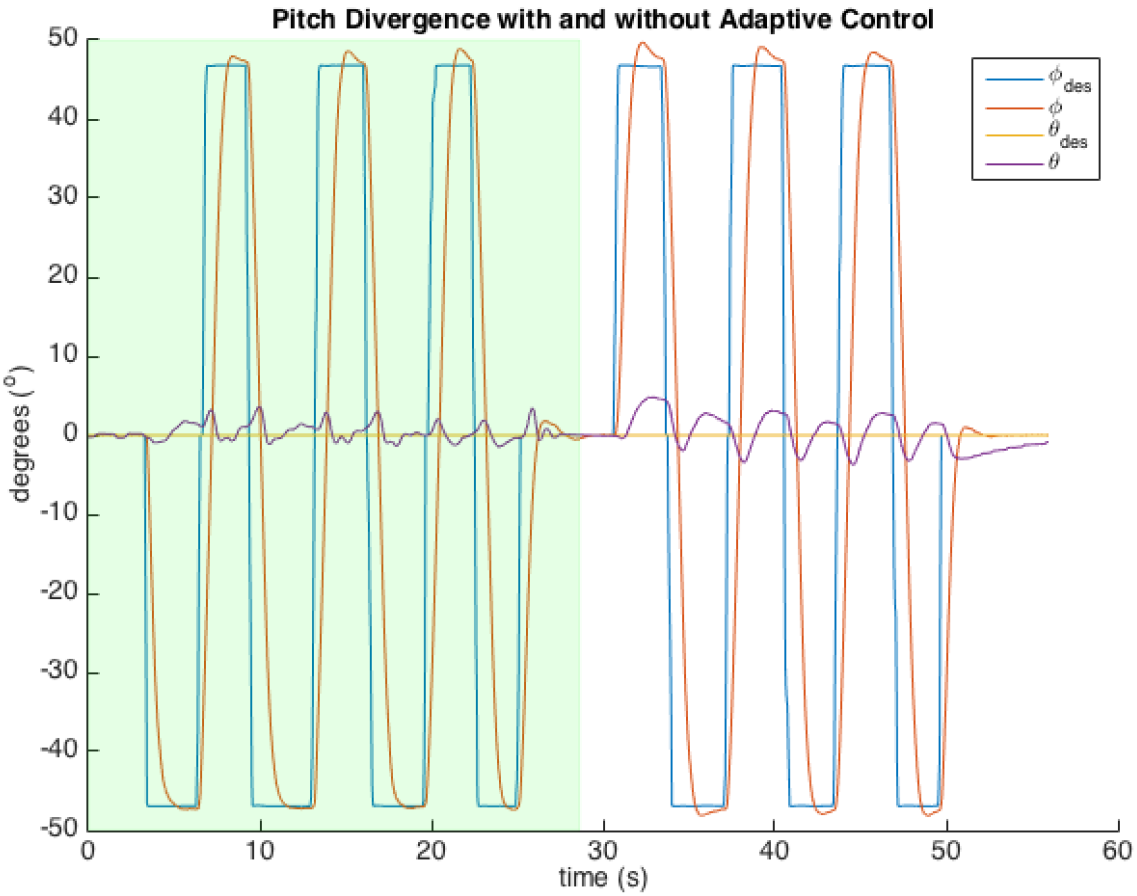

Figure 5.3. PID vs $\mathcal{L}_1$ Adaptive Control Coupled Pitch Resonse Performance

5.1.3 Pitch due to Landing Gear and Flaps

Lowering the landing gear and flaps entering the landing phase of flight causes un-commanded deviation in pitch. The F4U Corsair (see figure 5.4)in X-Plane10 was used to evaluate the attitude hold retention performance. This un-modeled aerodynamics can cause the integrator in a PID controller to saturate or wind up.

The $\mathcal{L}_1$ adaptive controller significantly reduces the attitude excursion due to flaps and landing gear.

5.1.4 Roll Performance

Adaptive control offered little improvement to roll performance with respect to the nominal attitude retention. The roll axis for multiple aircraft was typically seen to have higher cutoff

Figure 5.4. F4U Corsair model - X-Plane10

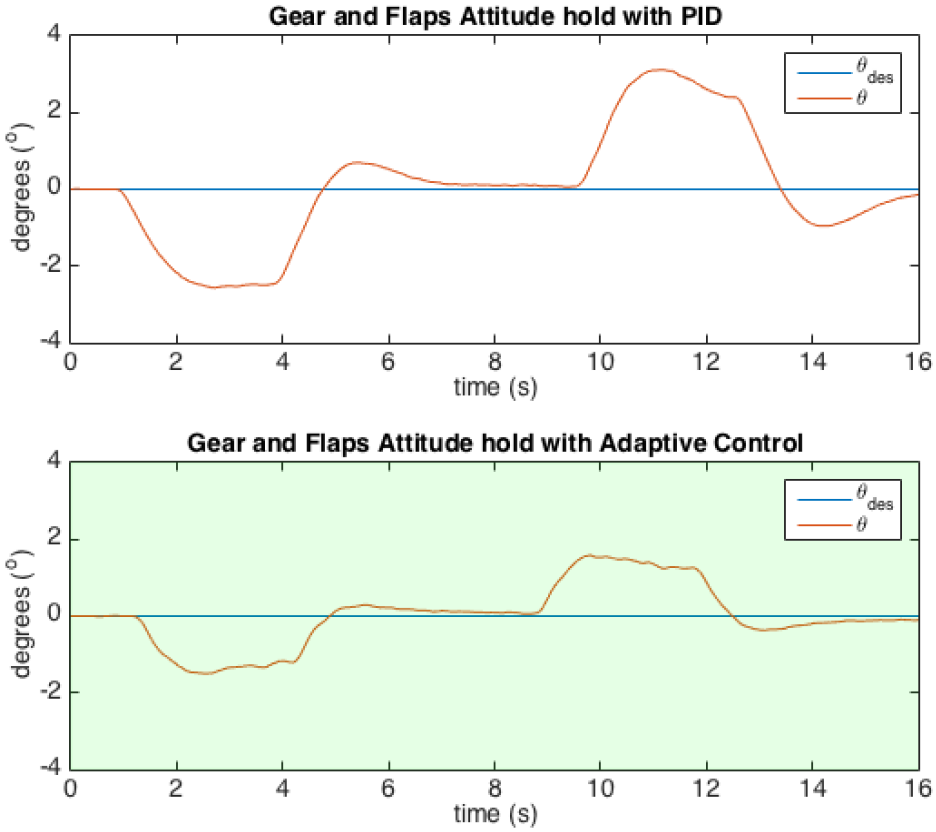

Figure 5.5. Pitch Attittude Response due to Gear and Flaps

frequencies with respect to pitch and therefore required higher values for the $D(s)$ cut off frequencies.

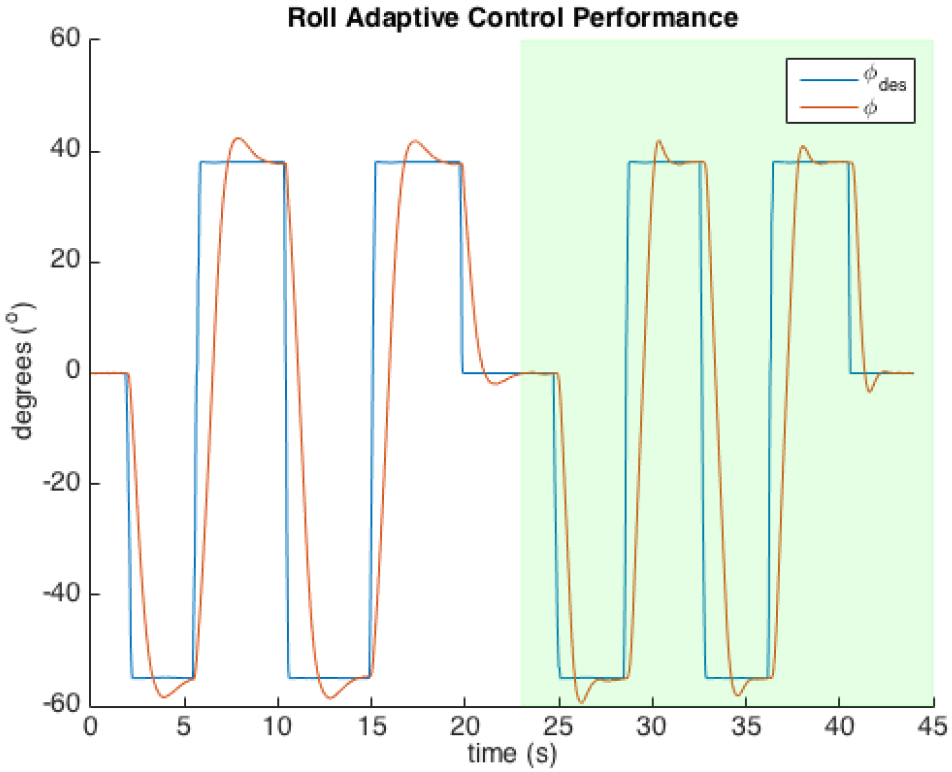

Figure 5.6. Roll Attittude Performance

5.1.5 Yaw to Roll Coupling

The roll dynamics are coupled to the yaw dynamics as seen in equation B.11. In the case of an actuator/servo hardover in the yaw channel, the coupling causes unwanted rolling moments. Figure 5.7 compares a left and right yaw servo hard over for both PID and the $\mathcal{L}_1$ controller. The adaptive controller significantly outperforms the PID controller in maintaining the desired roll. It could be argued that the PID controller's integrator time constant could be re-tuned to be comparable. It was evident for each of the simulation tests that tuning the PID for every scenario would likely have been possible individually to achieve similar performance to the $\mathcal{L}_1$. However, the $\mathcal{L}_1$ was not tuned between each of these tests and outperformed PID in most, if not all cases.

Figure 5.7. Roll Response to Rudder Servo Hardover/Failure

5.1.6 Actuator Miscalibration

The $\mathcal{L}_1$ adaptive control provided fast learning of airframe actuator miscalibrations. The autopilot parameter SERVOX_TRIM was used to offset the ailerons by 15% of their travel to evaluate how quickly the controller was capable of adapting to the new offset. This was tested first on the PID controller and then on the $\mathcal{L}_1$ as seen in figure 5.8.

It can be seen in figure 5.8 and 5.9 that the new trim value achieves the reference command in about 0.5 seconds for the $\mathcal{L}_1$. The $\mathcal{L}_1$ is slightly faster than PID, but the $\mathcal{L}_1$ exhibits some overshoot. It is important to note that this rate of learning is perfectly adequate for learning the miscalibrations in flight, but is insufficient for learning on take off. In the takeoff circumstance, the algorithm saturates the actuator if biases exist between desired and achieved. Because the aircraft has no dynamic pressure, the desired cannot be achieved. This causes controller saturation, which cannot be unwound fast enough for take-off (specifically for tail-dragger configuration). This has to be handled with ad-hoc heuristics very delicately. One could choose to speed scale the learned parameters to help with learning rate, but it was

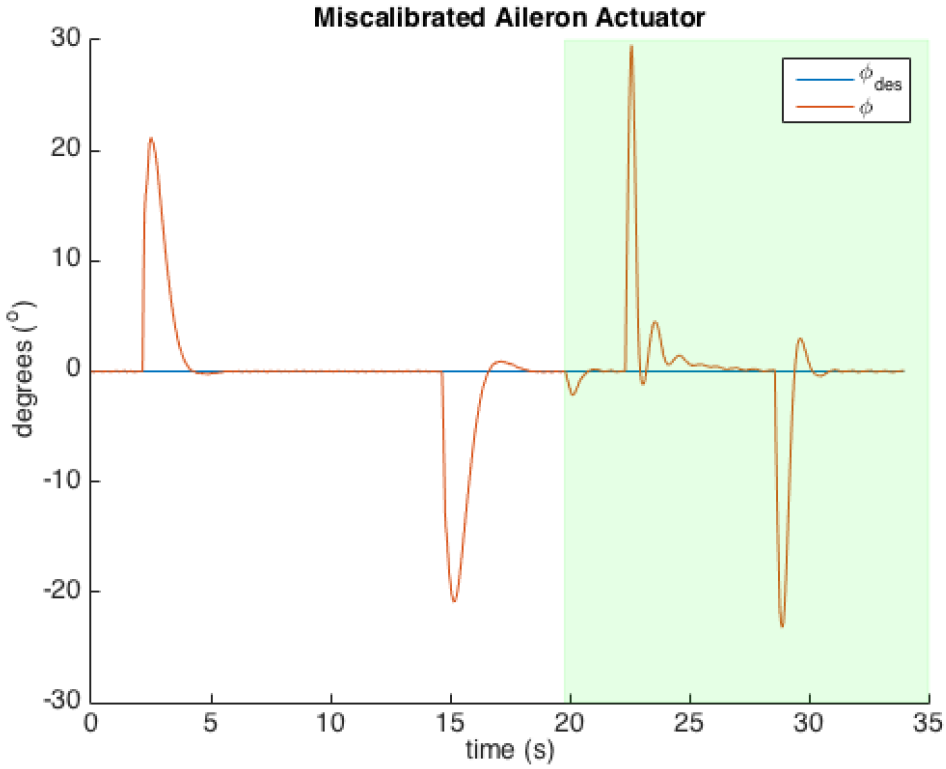

Figure 5.8. Roll Response to Miscalibrated Aileron Servo

not chosen for this research because this controller is also utilized in tail-sitter configurations where the zero airspeed would cause the controller to refuse to learn when in vertical take off and landing (VTOL) mode.

5.2 Flight Test Results

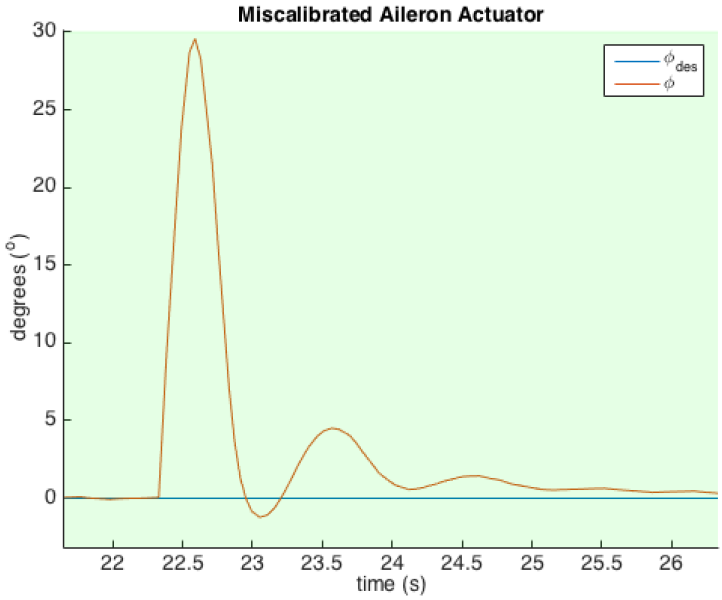

Figure 5.9. $\mathcal{L}_1$ Fast Adaptation to Unknown Miscalibration

CHAPTER 6:
Recommendation

6.1 $\mathcal{L}_1$ Adaptive Control Algorithm Tuning

The primary goal of this research was to reduce the complexity of tuning expertise required to get an unknown airframe airborne successfully. The amount of time required to get the adequate airframe performance using the $\mathcal{L}_1$ algorithm is significantly reduced. Three primary features need tuning: the adaptation gain, the controller filter, and the companion model.

6.1.1 Tuning the Adaptation gain

Tuning the adaptation gain is fairly intuitive. The adaptation gain is a function of the loop frequency at which the filter is run so it may only need to be tuned for a given autopilot at a given loop rate and not for every specific aircraft. Anecdotally, the adaptation gain of 10,000 was used on the Pixhawk1 autopilot running the scheduled loop rate at 100Hz across multiple aircraft without needing to modify. The primary feedback to the user for tuning this gain resides in the desired movement of the estimated states. The adaptation gain was set low (1-10) while watching the parameters (θ, ω, σ) adapt. The adaptation gain is slowly increased until the desired rate of adaptation as seen in the real time monitoring of the parameters is adequate. Another approach for tuning the adaptation rate is to increase the gain until parameters start to oscillate between the bounds and then reduce it. This bang-bang response in the parameter adaptation will show up in the performance of the controller as increased peaks away from zero error between desired and achieved. In the case of this research, this noise can be hard to identify in the rate control itself and therefore is why monitoring the attitude error helps find the appropriate adaptation gain which is extremely high but still not injecting attitude error spikes.

6.1.2 Tuning the Controller Filter

The adaptation feedback gain (k), as discussed in Section 3.3 was by far the most influential gain to tune. The simplicity of tuning the $\mathcal{L}_1$ algorithm resides in the fact that the majority of

lay users could adequately tune an airframe with this stand-alone gain. Adequate values for this gain ranged from 0.3 for responsive aircraft and 2.0 for very sluggish aircraft. This value was set for both the roll and pitch axis independently. As seen in equation 3.8, this value is establishing the cutoff frequency of the control filter that separates the bandwidth limited control channel output and the high bandwidth adaptation. The default value assigned in the source code was 0.45, which proved to be a good starting point. If the default value for k is not correct, then the control channel will produce either low-frequency oscillations or high-frequency oscillations. The low-frequency oscillations are produced because the bandwidth of the control is too low and there should be a perceptible lag between the desired state and the achieved state. High-frequency oscillations occur when the control filter bandwidth is set higher than the plant's bandwidth, and the aircraft is incapable of achieving the desired rates. Unlike PID control, the $\mathcal{L}_1$ control never exhibited unstable performance with incorrect gains. The controller simply oscillates with extremely poor performance. This was a remarkable feature because poorly tuned PID gains can cause an aircraft to depart controlled flight rapidly. Whereas, the $\mathcal{L}_1$ controller maintained bounded flight performance as the theory suggests.

6.1.3 Tuning the Companion Model Cutoff Frequency

System identification was conducted as seen in Appendix C in order to ascertain the bandwidth of various airframes and their actuators. Figure 6.1 is an example of second order models for two aircraft's roll dynamics compared to second order models of RC actuators (servos). As to be expected, the bandwidth for the actuators is slightly higher than the airframe dynamics.

These rough approximations were used to then place the cutoff frequencies of the companion model with the expectation that the companion model cannot achieve higher bandwidths than that of the airframe. Conservatively, the companion model cutoff frequency was typically set 2-5 radians per second lower than the expected max performance of the airframe. After the other algorithm gains are tuned, and satisfactory performance is achieved, the companion model cutoff frequency can then be increased to achieve higher performance.

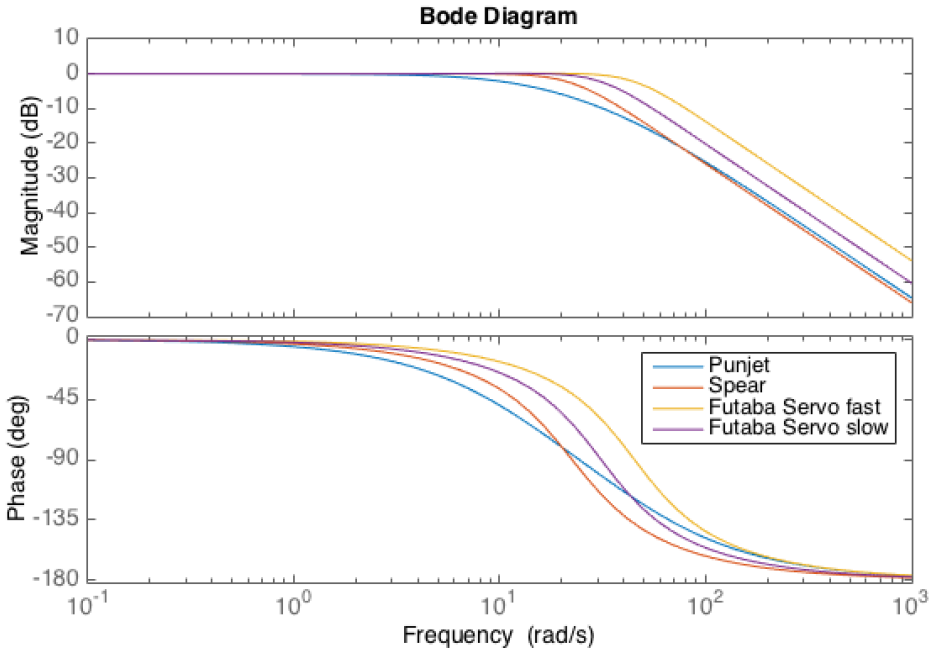

Figure 6.1. Aircraft Frequency Analysis

6.2 Improved Recursive Architecture

The speed of adaptation and accuracy of the discretized $\mathcal{L}_1$ algorithm is drastically improved with increased loop frequency. The algorithm was written as one recursive architecture that updates at the Pixhawk's scheduled loop rate. As previously discussed, the scheduled loop rate ideally should run at lower frequencies to prevent excessive log file size and added strain on the CPU. However, the $\mathcal{L}_1$ architecture only requires the adaptation loop be run at faster rates. With this specific performance enhancement in mind, the APM architecture could be modified to accept an independent loop specifically for the $\mathcal{L}_1$ adaptation update. The sensors measurements and Extended Kalman Filter (EKF) update can run significantly lower with only increased performance of the algorithm. The higher adaptation loop would enable higher adaptation gains (Γ) and consequently produce faster adaptation of the system.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 7:
Conclusion

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A:
Transfer Functions

A.1 Transfer Functions

This research utilizes the Transfer Function (TF) representation of aircraft flight dynamics that is typical of LTI systems. A transfer function is a very useful approach to describe the relationship between inputs and outputs of LTI systems. Both analytically and numerically, the TF approach has significant benefits in continuous and discrete time domains as its construct is based on well-developed properties and primitives of polynomials. These polynomial representations of various aerodynamic, flight dynamics, and control properties of aircraft are well-developed. What is unknown or partially known a priori, are the numerical values of coefficients for those polynomials. Therefore the tools from the areas of online estimation such as regression are utilized to solve for them.

Transfer functions take the form:

$$H(s) = \frac{Y(s)}{X(s)} \tag{A.1}$$

where:

Y(s) is the Laplace transform of the output
X(s) is the Laplace transform of the input

Standard physics models of first and second order form are well understood and seen in many model derivations. The first order model takes the form:

$$H(s) = \frac{k_{dc}}{\tau s + 1} = \frac{\omega_n}{s + \omega_n} \tag{A.2}$$

where:

k_{dc} is the DC gain

τ is the system time constant (time in seconds to reach 63% of steady state)

Similarly the standard form for a second order system takes the form:

$$H(s) = \frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2} \tag{A.3}$$

where:

ω_0 is the system natural frequency in radians per second
ζ is the system damping ratio

The modeling of a system can also be converted to a system of first order differential equations also known as state-space modeling. The following nomenclature will be used to illustrate the modeling of first order systems where $\dot{x}$ is the time derivative of the state, A is the Hurwitz matrix, B is the input matrix, and u is the input vector.

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{A.4}$$

APPENDIX B:
Fixed Wing Aircraft Dynamics Model

B.1 Fixed Wing Aircraft Dynamics Model

The following is the nomenclature that will be used to describe the kinematic equations. Euler angles for pitch (θ), roll (ϕ), and yaw (ψ) will have the units of radians. The following Figure B.1 illustrates the North East Down (NED) reference frame definitions used for body rotational rates about the x axis (p), y axis (q), and the z axis (r) as well as the body velocities in the x axis (u), y axis (v), and the z axis (w).

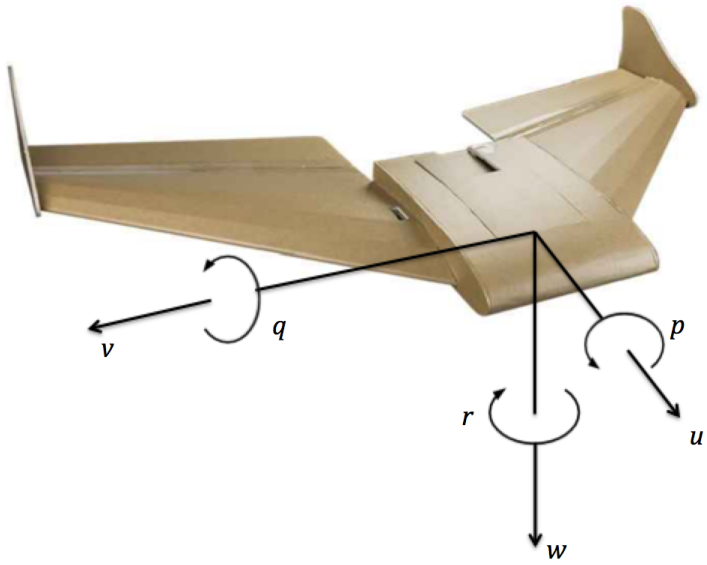

Figure B.1. Reference frame - body rates and velocities

The primary goal of this research is to implement an algorithm which controls fixed wing attitude. Therefore, the focus of the following kinematic and dynamics equations will primarily concentrate on deriving only rotational motion from first principles.

Newton's second law as it pertains to rotational motion can be stated as

$$\tau = J\frac{d\omega}{dt_i} \tag{B.1}$$

where τ is the torques applied to the body, J is the moment of inertia, and $\frac{d\omega}{dt_i}$ is the angular acceleration of the body with respect to the inertial frame.

Equation B.1 can be rewritten in the body reference frame as follows:

$$\tau^b = J\dot{\omega}^b_{b/i} + \omega^b_{b/i} \times \left(J\omega^b_{b/i}\right) \tag{B.2}$$

The expression $\dot{\omega}^b_{b/i}$ is the angular acceleration in the body frame as viewed in the body frame:

$$\dot{\omega}^b_{b/i} = \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} \tag{B.3}$$

The equation can then be rewritten with respect to $\dot{\omega}^b_{b/i}$:

$$\dot{\omega}^b_{b/i} = J^{-1}\left[-\omega^b_{b/i} \times \left(J\omega^b_{b/i}\right) + \tau^b\right] \tag{B.4}$$

J can be defined as the inertia matrix as follows

$$J = \begin{pmatrix} J_x & -J_{xy} & -J_{xz} \\ -J_{xy} & J_y & -J_{yz} \\ -J_{xz} & -J_{yz} & J_z \end{pmatrix} \tag{B.5}$$

The moments of inertia, or the diagonal terms, must be non-zero. The products of inertia, or the off-diagonal terms, are terms which describe the coupling between axis. For a traditional fixed wing aircraft, the natural symmetry will simplify the inertia matrix in the off-diagonal

terms as follows:

$$J = \begin{pmatrix} J_x & 0 & -J_{xz} \\ 0 & J_y & 0 \\ -J_{xz} & 0 & J_z \end{pmatrix} \tag{B.6}$$

The inverse of J can be found to be

$$J^{-1} = \begin{pmatrix} \frac{J_z}{\Gamma} & 0 & \frac{J_{xz}}{\Gamma} \\ 0 & \frac{1}{J_y} & 0 \\ \frac{J_{xz}}{\Gamma} & 0 & \frac{J_x}{\Gamma} \end{pmatrix} \tag{B.7}$$

where,

$$\Gamma = J_x J_z - J_{xz}^2 \tag{B.8}$$

Aircraft nomenclature for torques are defined $\tau \triangleq (l, m, n)^T$ and therefore the combined equations derived from first principles take the form:

$$\begin{aligned} \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} &= \begin{pmatrix} \frac{J_z}{\Gamma} & 0 & \frac{J_{xz}}{\Gamma} \\ 0 & \frac{1}{J_y} & 0 \\ \frac{J_{xz}}{\Gamma} & 0 & \frac{J_x}{\Gamma} \end{pmatrix} \left[\begin{pmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{pmatrix} \begin{pmatrix} J_x & 0 & -J_{xz} \\ 0 & J_y & 0 \\ -J_{xz} & 0 & J_z \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} + \begin{pmatrix} l \\ m \\ n \end{pmatrix} \right] \\ &= \begin{pmatrix} \frac{J_z}{\Gamma} & 0 & \frac{J_{xz}}{\Gamma} \\ 0 & \frac{1}{J_y} & 0 \\ \frac{J_{xz}}{\Gamma} & 0 & \frac{J_x}{\Gamma} \end{pmatrix} \left[\begin{pmatrix} J_{xz}pq + (J_y - J_z)qr \\ J_{xz}(r^2 - p^2) + (J_z - J_x)pr \\ (J_x - J_y)pq - J_{xz}qr \end{pmatrix} + \begin{pmatrix} l \\ m \\ n \end{pmatrix} \right] \\ &= \begin{pmatrix} \Gamma_1 pq - \Gamma_2 qr + \Gamma_3 l + \Gamma_4 n \\ \Gamma_5 pr - \Gamma_6(p^2 - r^2) + \frac{1}{J_y} m \\ \Gamma_7 pq - \Gamma_1 qr + \Gamma_4 l + \Gamma_8 n \end{pmatrix} \end{aligned} \tag{B.9}$$

where,

$$
\begin{aligned}
\Gamma_1 &= \frac{J_{xz}(J_x - J_y + J_z)}{\Gamma} \\
\Gamma_2 &= \frac{J_z(J_z - J_y) + J_{xz}^2}{\Gamma} \\
\Gamma_3 &= \frac{J_z}{\Gamma} \\
\Gamma_4 &= \frac{J_{xz}}{\Gamma} \\
\Gamma_5 &= \frac{J_z - J_x}{J_y} \\
\Gamma_6 &= \frac{J_{xz}}{J_y} \\
\Gamma_7 &= \frac{J_x(J_x - J_y) + J_{xz}^2}{\Gamma} \\
\Gamma_8 &= \frac{J_x}{\Gamma}
\end{aligned}
\tag{B.10}
$$

The aerodynamic torques (excluding propulsive torques) can be found to be:

$$
\begin{pmatrix} l \\ m \\ n \end{pmatrix} = \frac{1}{2}\rho V_a^2 S \begin{pmatrix} b\left[C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{b}{2Va}p + C_{l_r}\frac{b}{2V_a}r + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r\right] \\ c\left[C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{c}{2V_a}q + C_{m_{\delta_e}}\delta_e\right] \\ b\left[C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2Va}p + C_{n_r}\frac{b}{2V_a}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r\right] \end{pmatrix}
\tag{B.11}
$$

Substituting the aerodynamic torques found in equation B.11 into equation B.9 results in [13]:

$$
\begin{aligned}
\dot{p} &= \Gamma_1 pq - \Gamma_2 qr + \frac{1}{2}\rho V_a^2 Sb\left[C_{p_0} + C_{p_\beta}\beta + C_{p_p}\frac{bp}{2V_a} + C_{p_r}\frac{br}{2V_a} + C_{p_{\delta_a}}\delta_a + C_{p_{\delta_r}}\delta_r\right] \\
\dot{q} &= \Gamma_5 pr - \Gamma_6(p^2 - r^2) + \frac{1}{2}\rho V_a^2 Sc\frac{1}{J_y}\left[C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{cq}{2V_a} + C_{m_{\delta_e}}\delta_e\right] \\
\dot{r} &= \Gamma_7 pq - \Gamma_1 qr + \frac{1}{2}\rho V_a^2 Sb\left[C_{r_0} + C_{r_\beta}\beta + C_{r_p}\frac{bp}{2V_a} + C_{r_r}\frac{br}{2V_a} + C_{r_{\delta_a}}\delta_a + C_{r_{\delta_r}}\delta_r\right]
\end{aligned}
\tag{B.12}
$$

Simplifying equation B.12 assuming no inertial or aerodynamic coupling results in:

$$\begin{aligned}
\dot{p} &= \frac{1}{2}\rho V_a^2 Sb\left[C_{p_{\delta_a}}\delta_a + C_{p_p}\frac{bp}{2V_a} + C_{p_0}\right] \\
\dot{q} &= \frac{1}{2}\rho V_a^2 Sc\frac{1}{J_y}\left[C_{m_{\delta_e}}\delta_e + C_{m_q}\frac{cq}{2V_a} + C_{m_0}\right] \\
\dot{r} &= \frac{1}{2}\rho V_a^2 Sb\left[C_{r_{\delta_r}}\delta_r + C_{r_r}\frac{br}{2V_a} + C_{r_0}\right]
\end{aligned} \tag{B.13}$$

The equations in equation B.13 are then slightly modified to fit the first order Ordinary Differential Equation (ODE) model as described in the primary literature [8].

$$\begin{aligned}
\dot{\hat{p}} &= A_p\hat{p} + b_p\left(\hat{\omega}_p\delta_a + \hat{\theta}_p p + \hat{\sigma}_p\right) \\
\dot{\hat{q}} &= A_q\hat{q} + b_q\left(\hat{\omega}_q\delta_e + \hat{\theta}_q q + \hat{\sigma}_q\right) \\
\dot{\hat{r}} &= A_r\hat{r} + b_r\left(\hat{\omega}_r\delta_r + \hat{\theta}_r r + \hat{\sigma}_r\right)
\end{aligned} \tag{B.14}$$

where the hat nomenclature ($\hat{}$) recognizes that the parameters are estimates as follows,

$\hat{\omega}$ - estimated input gain coefficient
$\hat{\theta}$ - estimated constant state coefficient
$\hat{\sigma}$ - estimated disturbance estimate

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C:
System Identification

C.1 System Identification

C.1.1 Data Collection

The Pixhawk autopilot was used to capture roll and pitch rates ($\dot{p}, \dot{q}$) for the test vehicle as well as the pilot's command inputs. These outputs and inputs were the essential building blocks for creating pitch rate and roll rate models for the test vehicle. The autopilot is capable of logging data at 50-400 Hz and therefore is a discrete time domain signal. This data should ultimately be manipulated into the s-domain. The mathematics for this procedure are well defined, and numerous tools can be used to simplify this process.

It is crucial to ensure there is sufficient frequency content in the data recorded. Exciting multiple frequencies in the time domain ensures the regression techniques have an adequate sample space to search for polynomial coefficients. Exciting adequate frequency inputs is analogous to only sampling at one independent variable and expecting to get a regression fit from a non-changing dependent variable.

To ensure sufficient frequency content was obtained from the aircraft, a linear chirp was chosen and implemented into the Pixhawk source code as follows:

$$x(t) = sin\left[\phi_0 + 2\pi\left(f_0 t + \frac{k}{2}t^2\right)\right] \tag{C.1}$$

where:

ϕ_0 is the initial phase of the chirp at t=0 (nominally zero degrees)
f_0 is the initial frequency at t=0
k is the chirp rate
t is time in seconds

An example of this formulation can be seen below:

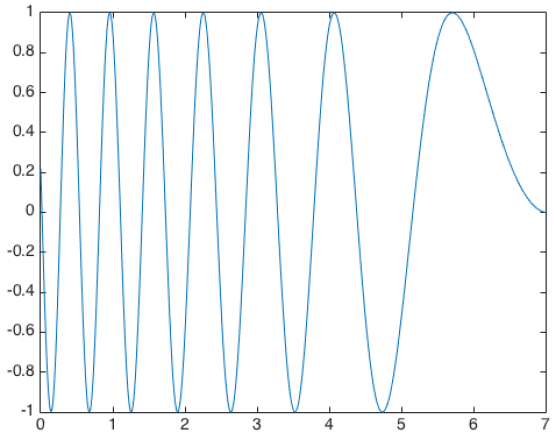

Figure C.1. Reverse Linear Chirp Example

C.1.2 z-Domain to s-Domain

The logged input and output data in discrete form requires shaping to convert cleanly into an s-domain representation. The first step is ensuring that the data is of constant sampling rate. In other words, the time between samples is uniform from sample to sample. The data provided from the Pixhawk autopilot does not have a uniform sampling rate. The sample rate is a user-defined rate (50-400Hz) but has a slight amount of jitter (±0.1%). Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) interpolation was used to interpolate the data into a uniform sampling rate.

After the data is shaped correctly with a uniform sample rate, the discrete data is transformed into a continuous approximation using a zero order hold (ZOH) technique. Taking the Laplace transform of the continuous input/output data will convert it into the s-domain, and finally, a non-linear least squares minimization algorithm can be run to find the polynomial coefficients which best fit the data.

The order of the regression (number of polynomials to estimate) is at the discretion of the engineer and their intuition of system's physical representation. Higher order models will better fit the data, but in most cases tend to overfit the data if they cannot be justified by physical principles. Most aircraft models assume that the system is LTI and second order. These fundamental aerodynamics models divide the modeling into longitudinal and lateral

dynamics. Each axis of the aircraft is assigned two, second order responses. Pitch, for example, has a second order response in the pitch damping mode (also known as the short period) and also has a second order response in the transition of kinetic energy to potential energy (also known as the long-period or phugoid). This would imply that the collected body rate data $(\dot{p}, \dot{q})$ collected by the autopilot should be modeled as first order systems because body rate is the first derivative of attitude. Assuming a first order system for the collected data in this experiment keeps the originally derived physical meaning but may be insufficient upon critical analysis. Both first order and second order model were estimated for comparison sake.

Results were collected from two flight test events. The first flight test was data collected prior to implementing the chirp, and the pilot attempted to increase frequency of the input signal manually. The second set of data collected was via the reverse linear chirp method previously described.

The manually piloted acquired data was expected not to have as adequate of frequency content in the signal but still provided adequate results for modeling the aircraft.

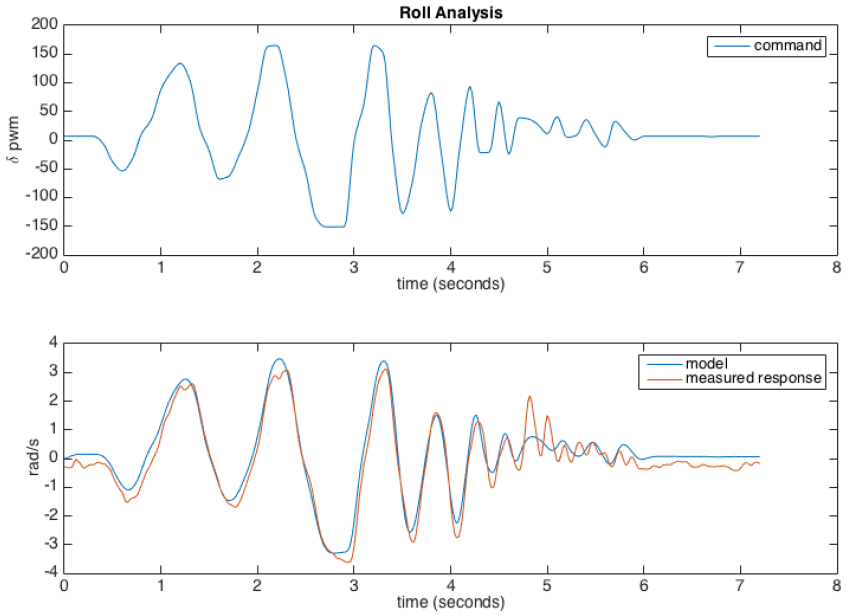

Figure C.2. Roll Model Regression with Manual Inputs

The above result demonstrates the utility of this technique even with poorly structured data

from manual pilot inputs. It can be seen that the second order model starts to misrepresent the data at higher frequencies. The lower frequency validity of this model showed potential and most of the high-frequency response may able to be neglected upon further review. The following figure with the chirp results highlights the actual issue with the high-frequency modeling issues.

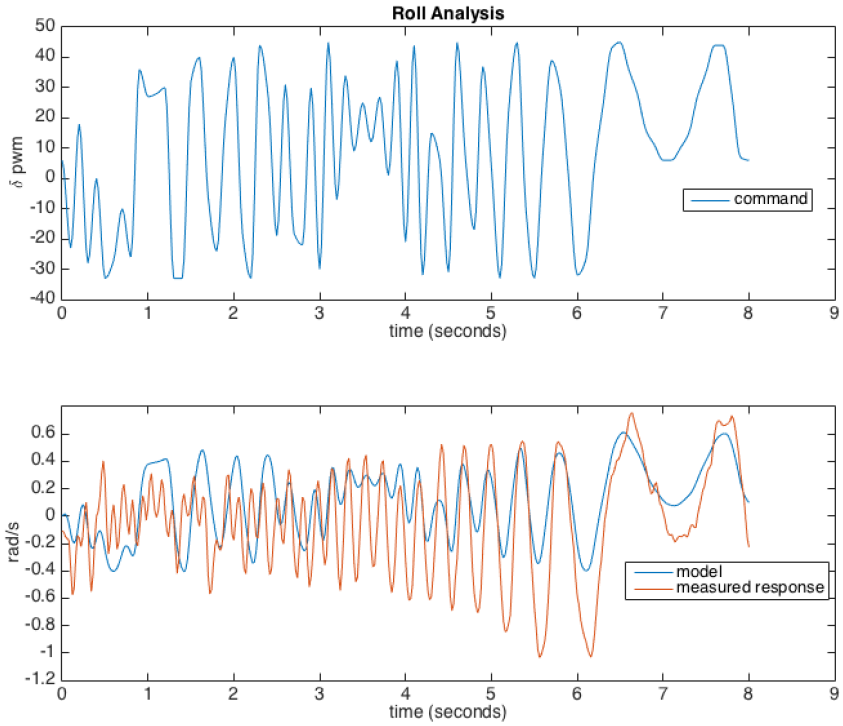

Figure C.3. Roll Model Regression with Reverse Linear Chirp

The reverse linear chirp, starting at high frequency and chirping down, clearly illustrates that this regression method has an underlying problem that was not evidently clear in the previous example. In the chirp analysis, it is clear that the high-frequency modeling is in error. After reviewing this result, it was clear that the sampling rate of the input channel was aliased. The chirp response was physically observed on pre-flight, in actual flight, and in the actual body rate of the aircraft. However, the aliased input channel was arbitrarily biasing the regression result. The data was logged at 50 Hz with the assumption that 50 Hz was ten times higher than the expected natural frequency of the aircraft and at least five times higher than the Nyquist criterion. The Nyquist criterion is the theoretical minimum frequency ($2f_0$) to sample a signal and recover a given frequency. The autopilot is capable

of logging data at 400Hz, but in this case, the servo output loop is still at 50Hz due to it being the hardware bandwidth limit of standard RC servos. Some high-performance servos are capable of higher input frequencies, but this still wouldn't solve the specific issue found when running this analysis. The peculiar aliasing issue is hardware specific to the Pixhawk 1 autopilot in how the main CPU sends servo commands to the auxiliary I/O CPU. The most recent version of firmware at the time of this test improperly logs the Pulse Width Modulation (PWM) through an aliased prone signal path. The main and I/O CPU both run at 50Hz with some appreciable clock drift. This generates a noticeable beat frequency and delay when the actual values in registry are saved for PWM values are sent back round trip to the main CPU. The implication of logging the PWM values at the very end of the digital transmission line seems valuable in principle because the values being logged are the undeniable values being sent to the actuators. However, the cost of logging these values in this manner on the Pixhawk architecture incurs significant aliasing at almost all frequencies. Logging the commanded PWM values prior to being sent to I/O CPU solved the aliasing discrepancy and produced very frequency rich models.

The manually piloted acquired data provided viable data source for the models even though it is a very simplistic approach. There were two separate manual tests run on the same aircraft on the same flight, and the following are the results using this regression technique to model a second order system.

$$H(s) = \frac{10.39}{s^2 + 31.26s + 504.9} \tag{C.2}$$

and

$$H(s) = \frac{10.61}{s^2 + 29.77s + 498.7} \tag{C.3}$$

Converting to standard form as described in equation A.3:

$$H(s) = \frac{0.0206 * 22.47^2}{s^2 + 2 * 0.69 * 22.47s + 22.47^2} \tag{C.4}$$

and

$$H(s) = \frac{0.0213 * 22.33^2}{s^2 + 2 * 0.67 * 22.33s + 22.33^2} \tag{C.5}$$

It is important to note that this system identification technique run on separate sets of data has produced two models with very similar values for ω_n and ζ.

This produces the average values of:

$\omega_n = 22.4 rad/s$
$k = 0.0209$
$\zeta = 0.681$

With the aliasing removed from the chirped input command signals as previously described, the model is drastically improved and produces the following results:

$$H(s) = \frac{4.409}{s^2 + 27.11s + 430.6} \tag{C.6}$$

and

$$H(s) = \frac{3.295}{s^2 + 18.82s + 2.96.5} \tag{C.7}$$

Converting to standard form as described in equation A.3:

$$H(s) = \frac{0.0102 * 20.75^2}{s^2 + 2 * 0.65 * 20.75s + 20.75^2} \tag{C.8}$$

and

$$H(s) = \frac{0.0111 * 17.21^2}{s^2 + 2 * 0.54 * 17.21s + 17.21^2} \tag{C.9}$$

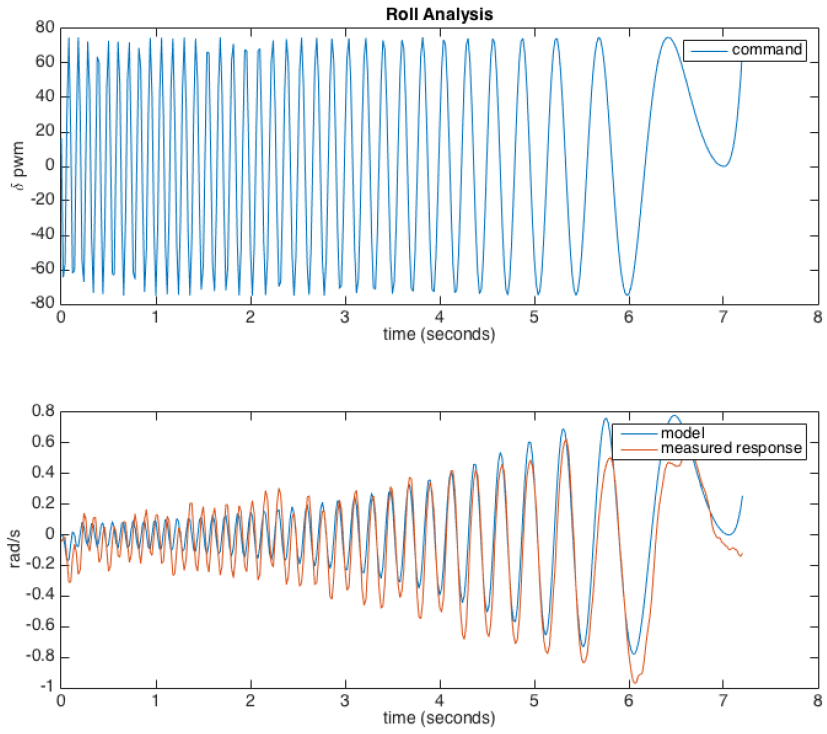

Figure C.4. Non-Aliased Reverse Chirp model example

This produces the average values of:

$\omega_n = 18.98 rad/s$
$k = 0.010$
$\zeta = 0.598$

In the author's experience, these values are reasonable values for this size and weight of airframe. This regression technique has shown potential to create realistic models from actual flight test data. The data must be properly shaped. The chirp method has the potential to increase the fidelity of the high-frequency response of the aircraft if the aliasing issue can be resolved on the command input channel.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D:
Projection Operator

D.1 $Proj(\theta, y)$ Derivation

In this section, the projection operator will be defined. Adaptive controllers often use the projection operator in their adaptive laws to ensure uniform boundedness of the system error. This can aid in faster adaptation and ensure controller closed-loop stability. The projection operator attempts to mathematically achieve two objectives; ensure the Lyapunov function time derivative remains negative semi-definite, and to keep the estimated parameters uniformly bounded. Using the projection operator ensures that θ is locally Lipchitz continuous even though the input y is piecewise continuous. The projection operator as utilized in this research is defined as:

$$Proj(\theta, y) \triangleq \begin{cases} y & \text{if } f(\theta) > 0, \\ y & \text{if } f(\theta) \leq 0 \text{ and } \nabla f^\top y \geq 0, \\ y - \frac{\nabla f}{\|\nabla f\|} \left\langle \frac{\nabla f}{\|\nabla f\|}, y \right\rangle f(\theta) & \text{if } f(\theta) \leq 0 \text{ and } \nabla f^\top y < 0. \end{cases} \tag{D.1}$$

where $\epsilon > 0$,

$$f(\theta) = -\frac{\theta^2 - \theta_{\max}^2}{\epsilon \theta_{\max}^2} \tag{D.2}$$

$$\nabla f^\top = -\frac{2\theta}{\epsilon \theta_{\max}^2} \tag{D.3}$$

The projection function chosen for this research is parabolic and has inflection points at user defined maximum/minimum bands. Equation D.2 is plotted in figure D.1 with example maximum/minimum of 0.65 and various values for ϵ. The engineer must set the value for ϵ to achieve the desired slope of the projection at the maximum values. This slope should be steep enough to capture the highest expected error given one recursion through the algorithm.

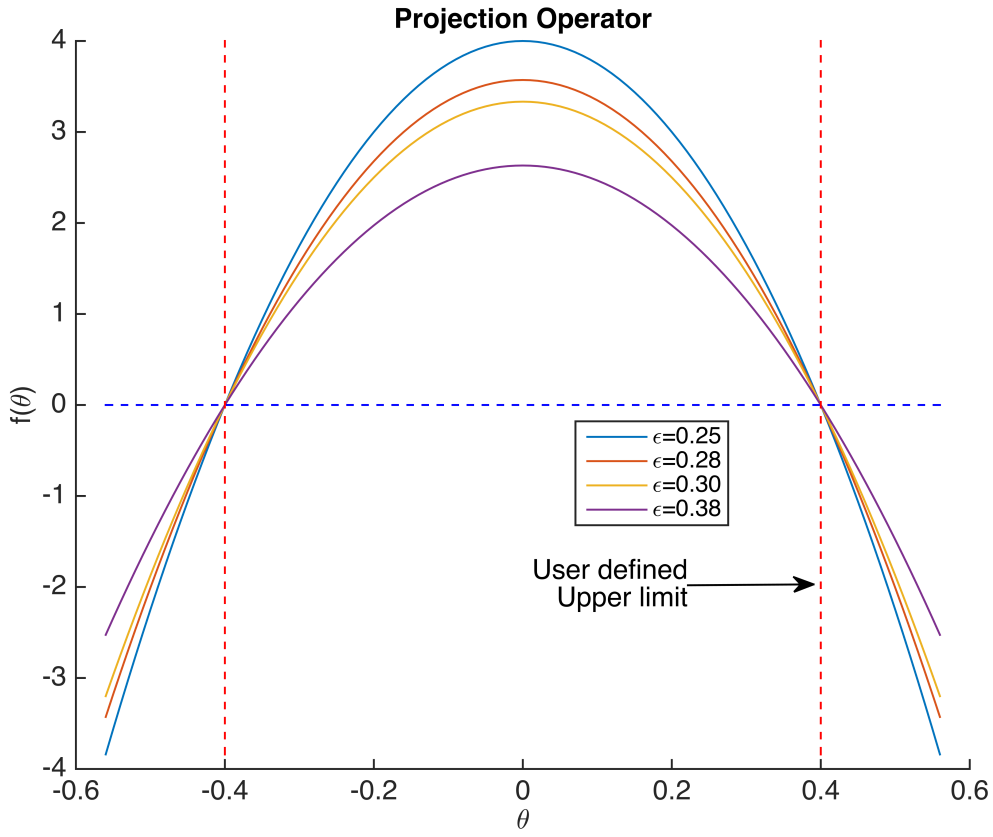

Figure D.1. Projection Operator - $f(\theta)$

There may exists systems which need to have projection bounds which are not symmetrical about zero. This was found to be the case for this research and therefore the following projection function was used to assist the engineer tuning the algorithm:

where $\epsilon > 0$,

$$f(\theta) = -\frac{4(\theta_{\min} - \theta)(\theta_{\max} - \theta)}{\epsilon(\theta_{\max} - \theta_{\min})^2} \tag{D.4}$$

$$\nabla f^\top = \frac{4(\theta_{\min} + \theta_{\max} - 2\theta)}{\epsilon(\theta_{\max} - \theta_{\min})^2} \tag{D.5}$$

Equation D.4 is plotted in figure D.2 with example maximum of 0.65, minimum of 0.25, and various values for ϵ.

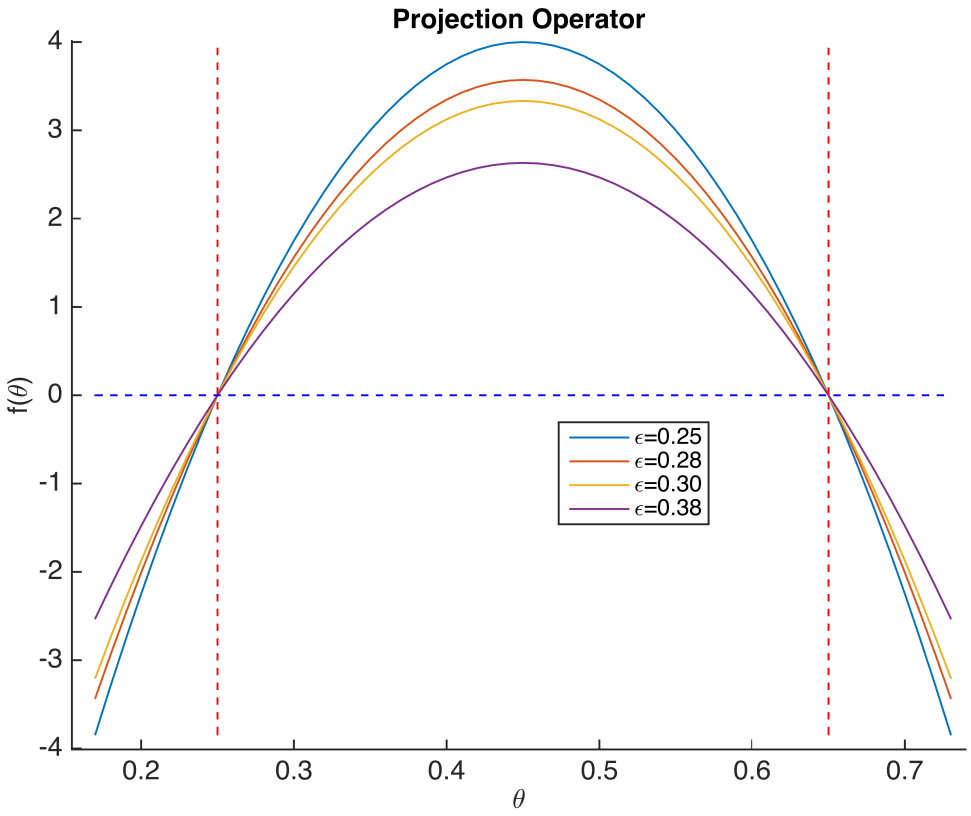

Figure D.2. Projection Operator with offset limits

D.2 C++ Implementation

```
float projection_operator(float theta,float y,float epsilon,
float theta_max,float theta_min)
{

        // Calculate convex function
        float f_theta = (-4*(theta_min-theta)*(theta_max-theta))
        /(epsilon*(theta_max-theta_min)^2);
        float f_theta_dot = (4*(theta_min+theta_max-(2*theta)))
        /(epsilon*(theta_max-theta_min)^2);

        float projection_out = y;

        if (f_theta <=0 && f_theta_dot*y < 0)
        {
```

```
                projection_out = y*(f_theta+1); // y-(y*-f_theta);
        }

        return projection_out;
}
```

APPENDIX E:
Matlab Code

E.1 Euler vs Trapezoidal Method

```
clear, format long, clc, close all
dt = 1;

t0 = [0:.01:4];

y0 = exp(t0);

%Euler integration

t1 = [0:4];
y1 = exp(0);
for i=1:4
    y1(i+1)=y1(i) + dt*exp(i-1);
end

%Trapezoidal integration
t2 = [0:4];
y2 = exp(0);

for i=1:4
    y2(i+1)=y2(i) + dt*exp(i-1);
    y2(i+1)=y2(i) + (dt/2)*(exp(i-1)+y2(i+1));
end

plot(t0,y0,'k--',t1,y1,'b',t2,y2,'r')
legend('y=e^t','Euler Method','Trapezoidal Method')
```

E.2 SISO Lyapunov Solution Proof

```
clear,clc, format compact, close all
%% Find Pb
wn = [5:0.1:10]; %rad/s
```

```
for i=1:length(wn)
    b = [wn(i)];
    a = [1,wn(i)];
    [A,B,C,D] = tf2ss(b,a)

    Pb(i) = lyap(A,1);
end
Pb_test = 1./(2*wn);

plot(wn,Pb_test,'o',wn,Pb)

%in this simple 1x1 matrix case Pb is easy to calc by hand in code
% Pb ends up being: Pb=1/2*wn;
```

E.3 Reverse Linear Chirp

```
clear, format compact, clc, close all

fs = 1/200;
t = [0:fs:7];
f0= 0.01;%Hz
f1= 10; %Hz
k = (f1-f0)/(t(end));

phi_0 = 0.0;

sample_delay = 0.02;

out = sin(phi_0+2*pi*(f0*(7-(t+sample_delay))+(k/2).*(7-(t+sample_delay)
    ).^2));
%out = 1500 + out*10;

out = out*10;

plot(t,out)
```

E.4 Projection Operator Example Plots

```
clear, clc, format compact, close all
```

```
2
3  %% Plot Projection
4  epsilon = [0.25,0.28,0.3,0.38];
5  theta_max = 0.65;
6  theta_min = 0.25;
7  center = (theta_max+theta_min)/2;
8
9  span = (theta_max-theta_min)*1.4;
10 theta = [center-(span/2):0.01:center+(span/2)];
11
12 for j=1:length(epsilon)
13     for i=1:length(theta)
14         %f_theta(i,j) = -(theta(i).^2-theta_max^2)/(epsilon(j)*theta_max
               ^2);
15         %f_theta_dot(i,j) = -(2*theta(i))/(epsilon(j)*theta_max^2);
16         f_theta(i,j) = -(theta_min-theta(i))*(theta_max-theta(i))./(
               epsilon(j));
17         f_theta_dot(i,j) = (theta_min+theta_max-(2*theta(i)))/(epsilon(j
               ));
18     end
19 end
20
21 figure
22 for j=1:length(epsilon)
23 hold on
24 plot(theta,f_theta(:,j))
25 end
26 line([min(theta),max(theta)],[0,0],'color','blue','linestyle','--')
27 line([theta_min,theta_min],[min(f_theta(:,1)),max(f_theta(:,1))],'color'
       ,'red','linestyle','--')
28 line([theta_max,theta_max],[min(f_theta(:,1)),max(f_theta(:,1))],'color'
       ,'red','linestyle','--')
29 legend('\epsilon=0.25','\epsilon=0.28','\epsilon=0.30','\epsilon=0.38')
30 ylabel('f(\theta)')
31 xlabel('\theta')
32 title('Projection Operator')
33 hold off
```

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] D. D. of Defense), "The role of autonomy in dod systems," July 2012.

[2] N. Minorsky, "Directional stability of automatically steered bodies," *Naval Engineers Journal*, vol. 32, no. 2, 1922.

[3] D. R. Jenkins, "Hypersonics before the shuttle: A concise history of the x-15 research airplane," 2000.

[4] Z. T. Dydek, A. M. Annaswamy, and E. Lavretsky, "Adaptive control and the nasa x-15-3 flight revisited," *IEEE Control Systems*, vol. 30, no. 3, pp. 32–48, 2010.

[5] E. Lavretsky and K. A. Wise, "Robust adaptive control," in *Robust and Adaptive Control*. Springer, 2013, pp. 317–353.

[6] K. J. Åström and B. Wittenmark, *Adaptive control*. Courier Corporation, 2013.

[7] A. M. Lyapunov, "The general problem of motion stability," *Annals of Mathematics Studies*, vol. 17, 1892.

[8] N. Hovakimyan and C. Cao, *L1 adaptive control theory: guaranteed robustness with fast adaptation*. Siam, 2010, vol. 21.

[9] B. D. Anderson *et al.*, "Failures of adaptive control theory and their resolution," *Communications in Information & Systems*, vol. 5, no. 1, pp. 1–20, 2005.

[10] [Online]. Available: https://github.com/ArduPilot/MAVProxy

[11] [Online]. Available: https://github.com/mavlink/mavlink/

[12] [Online]. Available: http://ardupilot.github.io/MAVProxy/html/index.html

[13] R. W. Beard and T. W. McLain, *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California