Modeling and Adaptive Control of Indoor Unmanned Aerial Vehicles

by

Bernard Michini

B.S., Aeronautics and Astronautics
Massachusetts Institute of Technology (2007)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2009

Author .....
Department of Aeronautics and Astronautics
August 28, 2009

Certified by .....
Jonathan P. How
Professor
Thesis Supervisor

Accepted by .....
Prof. David L. Darmofal
Associate Department Head
Chair, Committee on Graduate Students

Modeling and Adaptive Control of Indoor Unmanned Aerial Vehicles

by

Bernard Michini

Submitted to the Department of Aeronautics and Astronautics
on August 28, 2009, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

The operation of unmanned aerial vehicles (UAVs) in constrained indoor environments presents many unique challenges in control and planning. This thesis investigates modeling, adaptive control and trajectory optimization methods as applied to indoor autonomous flight vehicles in both a theoretical and experimental context. Three types of small-scale UAVs, including a custom-built three-wing tailsitter, are combined with a motion capture system and ground computer network to form a testbed capable of indoor autonomous flight.

An $\mathcal{L}_1$ adaptive output feedback control design process is presented in which control parameters are systematically determined based on intuitive desired performance and robustness metrics set by the designer. Flight test results using a quadrotor helicopter demonstrate that designer specifications correspond to the expected physical responses. Multi-input multi-output (MIMO) $\mathcal{L}_1$ adaptive control is applied to a three-wing tailsitter. An inner-loop body rate adaptation structure is used to bypass the non-linearities of the closed-loop system, producing an adaptive architecture that is invariant to the choice of baseline controller. Simulations and flight experiments confirm that the MIMO adaptive augmentation effectively recovers nominal reference performance of the vehicle in the presence of substantial physical actuator failures.

A method for developing a low-fidelity model of propeller-driven UAVs is presented and compared to data collected from flight hardware. The method is used to derive a model of a fixed-wing aerobatic aircraft which is then used by a Gauss pseudospectral optimization tool to find dynamically feasible trajectories for specified flight maneuvers. Several trajectories are generated and implemented on flight hardware to experimentally validate both the modeling and trajectory generation methods.

Thesis Supervisor: Jonathan P. How
Title: Professor

Acknowledgments

I'd like to extend a sincere thank you to my advisor, Professor Jonathan How, for his patient guidance over the past two years. His continual feedback, insights, and suggestions have driven me achieve and learn much more than would have been possible on my own.

Thanks to everyone in the Aerospace Controls Lab for their support and friendship. Specific thanks go to Josh Redding, Dan Levine, Brett Bethke, Brandon Luders, Sameera Ponda and Cam Fraser for all of their help throughout my graduate career, to Karl Kulling for his invaluable guidance with my flight training experience, and to Ben Chiel for his timely and diligent work that contributed directly to this thesis. I'd especially like to extend my gratitude to Frant Sobolic, from whom I learned a great deal over the course of an enjoyable two years. And, of course, thanks to Kathryn Fischer for somehow managing to keep this lab operational despite our best efforts.

Finally, I want to thank my family and friends for the support and love that is so essential to any success I may find. To my mom, dad, and brother who have been there for me throughout my entire life, who have backed me up unconditionally no matter what kind of trouble I find myself in, and to whom I owe more than I can ever hope to repay: you've made me all that I am. Thank you.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

1.1 Motivation

Unmanned aerial vehicles (UAVs) have become increasingly prominent in a variety of aerospace applications. While most commercially available UAVs are designed to operate outdoors in large, open environments, the rapid evolution of computer, sensing, and estimation technology brings the prospect of useful indoor UAVs closer every day. Such vehicles could open the door for an entirely new set of unique mission capabilities.

However, with the possibility of indoor flight comes many associated challenges in vehicle planning and control. For example, position constraints become tighter with more severe consequences for violation. Obstacles in a cluttered indoor environment make the task of dynamic trajectory planning essential. These examples illustrate the requirement for more than conventional planning and control methods. Advanced adaptive control and trajectory planning algorithms may very well be needed to address these challenges.

Despite the reasons necessitating the use of adaptive control and trajectory planning in the future of unmanned flight, implementation of such algorithms on hardware has been limited. The increased potential to improve performance typically comes along with an increased risk of failure. Thus it is difficult to justify trading the risk of losing an expensive airframe and avionics system for performance benefits that may

not be analytically guaranteed.

This thesis explores the use of both adaptive control and trajectory generation techniques in the context of indoor autonomous flight. With a testing facility that has the unique capability to enable such flight in a low-risk, controlled indoor environment, advanced algorithms are tested in both simulation and experimental implementation.

1.2 Literature Review

1.2.1 Control and Flight Testing of UAVs

Advances in autopilot hardware have enabled research institutions to conduct flight experiments testing a variety UAV control and coordination algorithms. Several notable outdoor testbeds have been developed. Stanford's STARMAC testbed utilizes a custom autopilot to control a fleet of quadrotors for multi-agent control research [1]. Brigham Young University's MAGICC Lab has developed a broad range of outdoor testing capabilities for a variety of research topics in flight control, coordination, and planning [2]. UAV testbeds have also been used to investigate the potential of adaptive control algorithms to improve flight performance and robustness [3, 4].

Few research facilities have the ability to perform *indoor* flight experiments with small-scale, agile UAVs. The Realtime Autonomous Vehicle test ENvironment (RAVEN) at the Massachusetts Institute of Technology Aerospace Controls Laboratory is capable of performing such indoor tests [5, 6]. RAVEN bypasses the challenge of onboard sensing and control with the use of a motion capture system and ground computers (details in Section 2.5). This enables the rapid-prototyping and testing of experimental control algorithms in a low-risk, controlled indoor environment [7].

1.2.2 $\mathcal{L}_1$ Adaptive Control

Model Reference Adaptive Control (MRAC) has been thoroughly researched for the task of recovering nominal performance in the presence of uncertainties, but can be

particularly susceptible to time delays [8]. A filtered version of MRAC, termed $\mathcal{L}_1$ adaptive control, was developed to address these issues and offer a more realistic adaptive solution. [9, 10]. The main advantage of $\mathcal{L}_1$ adaptive control over other adaptive control algorithms such as MRAC is that $\mathcal{L}_1$ cleanly separates performance and robustness. The inclusion of a low-pass filter not only guarantees a bandwidth-limited control signal, but also allows for an arbitrarily-high adaptation rate limited only by available computational resources. This parameterizes the adaptive control problem into two very realistic constraints: actuator bandwidth and available computation.

In Chapter 4 of this thesis the *output feedback* version of $\mathcal{L}_1$ described in [11] is considered. One consequence of using this output feedback (as opposed to full-state feedback) form of $\mathcal{L}_1$ is that the expected closed-loop response becomes somewhat complex. Whereas in full-state form the reference model sets the desired system behavior, with output feedback $\mathcal{L}_1$ it is not immediately clear how to choose the design parameters to achieve some desired response. Previous design methodologies have focused on norm minimization and time delay optimization via modification of only the low-pass control signal filter [9, 10, 12]. In [13], more than just the low-pass filter is considered, but the analysis again relies on a system norm as the only performance metric. Metrics are validated through extensive flight testing in [3], but these metrics are not considered in an *a priori* control design process. None of the approaches listed comprises a systematic design process that considers both transient performance and robustness simultaneously. Such a design process would be a key step for further application of $\mathcal{L}_1$ adaptive output feedback control in real-world applications including indoor autonomous flight.

A multi-input multi-output (MIMO) form of $\mathcal{L}_1$ adaptive control is developed in Refs. [14] and [15]. Full-state (as opposed to output) feedback is assumed, avoiding the design issues described above. However, the MIMO form requires a *linear* reference model. This prevents the applicability of MIMO $\mathcal{L}_1$ to systems with non-linear plant or control models. Also, there are no published experimental MIMO $\mathcal{L}_1$ flight results at the time of this writing.

1.2.3 UAV Modeling and Trajectory Generation

While the problem of finding accurate dynamic models for flight vehicles has long been a topic of intense research, many techniques result in models that are too complex to be used practically for the purposes of UAV planning and control. Several works address the problem of finding models for UAVs that are less complex yet still capture the most essential aerodynamic effects. In [16], a blade-element propeller model is combined with an aerodynamic panel method to create a model that is used for multi-objective airframe optimization. A similar propeller model is used in [17] but with a numerical vortex aerodynamic approach.

Trajectory planning for agile maneuvers such as those that may be required for indoor autonomous flight requires close consideration of the vehicle dynamics. Several notable works have addressed this problem for autonomous helicopter maneuvers. In [18], the Maneuver Automaton is introduced as a way of specifying and finding agile trajectories in a general mathematical context. A supervised learning approach to aerobatic trajectory generation is used in [19], whereby training data from a human operator provides a starting point from which the controller can begin to iteratively learn a maneuver. In [20], a vehicle dynamics model for a small fixed-wing UAV (the same as is used later in this thesis) is constructed largely from measured wind tunnel data . A Lyapunov backstepping technique is then applied to determine the control commands require to achieve a desired trajectory.

1.3 Contributions

In Chapter 4, a design process is developed that addresses the difficulties of applying $\mathcal{L}_1$ adaptive output feedback control in realistic flight control scenarios. While previous efforts focus more on non-tangible performance metrics, the design process presented allows the user to specify meaningful and intuitive characteristics of both the transient response and robustness of the controlled system. The process is verified with implementation on an indoor autonomous quadrotor, demonstrating that variations in the specified cost function produce the expected and desired physical

responses.

A MIMO $\mathcal{L}_1$ adaptive controller is applied to a three-wing tailsitter in Chapter 5. The approach differs from previous publications in that the adaptation must be applied about the inner-most body rate loop of the system due to a non-linear plant and baseline controller. Also, there have been no published implementations of MIMO $\mathcal{L}_1$ control to date, and the thesis proposes a simple adaptive anti-windup scheme that is found to be essential for reliable flight testing of the algorithm.

In Chapter 6, a previously-developed propeller model is combined with flat-plate aerodynamic theory to provide a low-fidelity model of propeller-based UAVs. The model is unique from previous research in its ability to capture the important first order effects dominating vehicle dynamics with very little computational complexity.

The vehicle model is used in Chapter 7 for a trajectory generation technique based on the Gauss pseudospectral optimization method. As opposed to those previously-published, the technique presented enables the specification of desired vehicle trajectories in an intuitive cost function format. In addition, state and control constraints are easily modified and the trajectories generated are guaranteed to be dynamically feasible if a solution exists. Finally, the technique is successfully tested on a fixed-wing aerobatic aircraft in a closed-loop manner for two-dimensional agile maneuvers.

Possibly the most important area of contribution throughout the thesis is the implementation and flight testing of the techniques explored. Oftentimes controls algorithms are tested only in simulation, giving no indication as to whether or not they would actually be useful in a real-world situation. While analytical proofs and simulation results are, of course, absolutely essential in the development process, implementation is often neglected as the *final* means by which a method's usefulness will be measured. The difficulties involved in implementation also serve to provide insights into the algorithms themselves that may never have been realized through simulation alone. Special care is given throughout the thesis to ground any analytical findings firmly in the applicability to physical flight vehicles.

1.4 Overview

The thesis is structured as follows. Chapter 2 introduces the three indoor autonomous vehicles used for flight testing throughout the thesis. The RAVEN testbed is also introduced as a facility enabling indoor autonomous flight. Chapter 3 describes the quaternion-based attitude controller for the flight vehicles as well as the outer-loop horizontal velocity controller. Chapter 4 proposes a systematic design process for the use of $\mathcal{L}_1$ adaptive output feedback control in realistic flight control applications. In Chapter 5, multi-input multi-output $\mathcal{L}_1$ adaptive control is applied to a novel three-wing aircraft with a non-linear baseline controller. Chapter 6 presents a low-complexity modeling procedure for propeller-driven UAVs. In Chapter 7, a trajectory optimization method is presented that makes use of the model and generates dynamically feasible trajectories via a user-specified cost function. Flight test results complement analytical findings throughout the work. Finally, Chapter 8 serves to summarize the thesis and outline future directions of research.

Chapter 2

Flight Vehicles

2.1 Introduction

This section gives a brief overview of the various UAVs used in the chapters to follow. The three main vehicles used in flight testing are a quadrotor helicopter, a custom-built three-wing tailsitter, and a fixed-wing aerobatic aircraft. Each has its own unique properties and control challenges as outlined below. Table 2.1 summarizes key physical parameters for each vehicle. The section also describes the RAVEN testbed used to carry out indoor flight experiments.

2.2 Quadrotor Helicopter

The Draganflyer VTi Pro quadrotor, shown in Figure 2-1 (left), is a small-scale (< 1 kg) quadrotor helicopter well-suited for indoor autonomous flight. The vehicle uses four independently controlled, fixed pitch, counter-rotating propellers to achieve

Table 2.1: Summary of key physical parameters for flight test vehicles.

Vehicle	Mass (g)	Max. Thrust (N)	Prop. Diameter (m)	Wing Span (m)	Batt. Type
Quadrotor	580	6.9	0.31	n/a	11.1v/2Ah
Three-wing	145	2.9	0.20	0.52	7.4v/0.4Ah
Fixed-wing	165	2.9	0.20	0.78	7.4v/0.4Ah

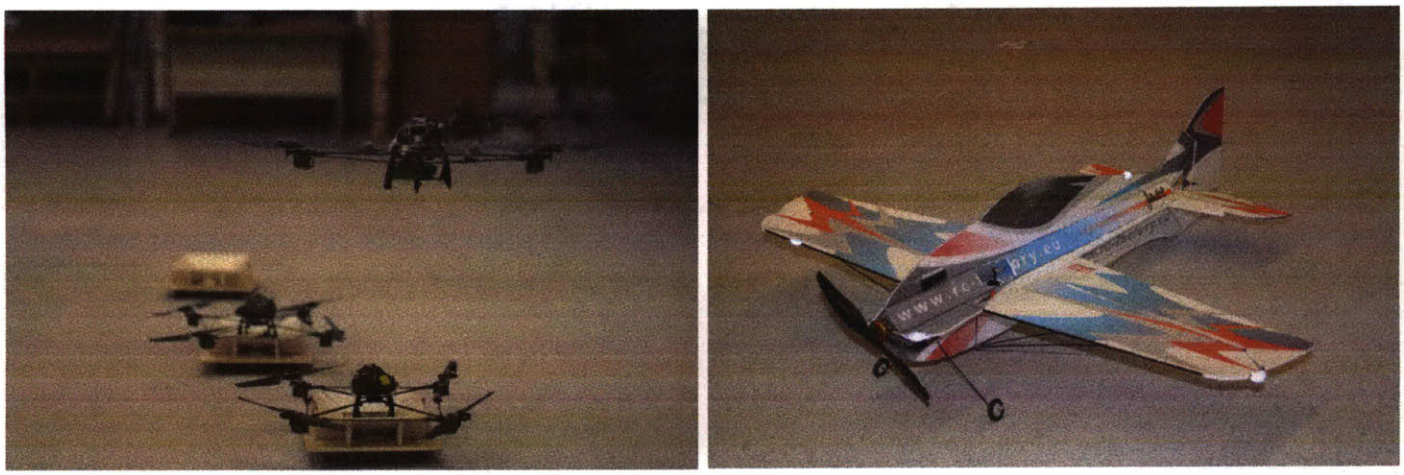

Figure 2-1: Quadrotor helicopters (left) and fixed-wing aerobatic aircraft (right).

control about all three axes. As such, a complex variable-pitch swashplate system is not required which greatly increases the physical robustness of the aircraft. Onboard rate gyros stabilize the angular rates allowing a human pilot to fly the vehicle with a relatively small amount of practice. The typical Lithium Polymer batteries used allow for flight times of 7-10 minutes.

Lack of aerodynamic surfaces gives the vehicle relatively predictable dynamic characteristics. However, one control challenge to be addressed is the susceptibility of the vehicle to actuator variation and partial failure. The brushed DC motors used for propulsion are subject to slow failure over the lifetime of the vehicle. Also, the rudimentary motor controllers onboard do not account for change in battery performance over the course of the flight, for which a 12% drop in voltage is typical.

2.3 Three-Wing Tailsitter

In an effort to test adaptive algorithms on more dynamically complex flight vehicles, a custom three-wing tailsitter was designed and built. As shown in Figure 2-2, the vehicle has three symmetric, triangular wings with independent control surfaces at the bottom of each. A carbon-fiber skeleton gives the vehicle structural solidity and allows for an extremely light-weight airframe (~145g) with a high thrust-to-weight ratio (~2.0). The vehicle uses standard RC commercial-off-the-shelf servos, motor,

Figure 2-2: Three-wing tailsitter (left and center) and its RC flight hardware (right).

motor controller, and radio.

The three control surfaces and the wings themselves are almost fully immersed in propeller wash, giving rise to a complex aerodynamic situation. This and the fact that a single control surface effects a moment about all three CG axes make the vehicle dynamics very difficult to model accurately. The response grows more complex as the vehicle translates out of the hover configuration and free stream velocity becomes a factor. This gives rise to a difficult control problem that may be addressed with adaptive control algorithms.

2.4 Fixed-Wing Aerobatic Aircraft

The Clik F3P competition plane is shown in Figure 2-1 (right). It is an extremely light ($<$ 200g) airframe designed for aggressive aerobatic maneuvering. The Clik's high thrust-to-weight ratio ($\sim$1.75) give it the ability to hover in a "prop-hang" configuration and transition smoothly to forward flight. The vehicle uses the same RC COTS hardware as the three-wing tailsitter.

This aircraft provides several relevant control challenges. The dynamics within a single flight regime (e.g. during the hovering prop-hang) can be well-approximated with standard aerodynamic modeling techniques. However, as the aircraft approaches forward flight and its translational speed increases, the dynamics change substantially. Also, it is not immediately obvious how to design trajectories for this aircraft that achieve some meaningful goal. For instance, to command a step in position, the aircraft would start in hover, smoothly transition to forward flight, then tran-

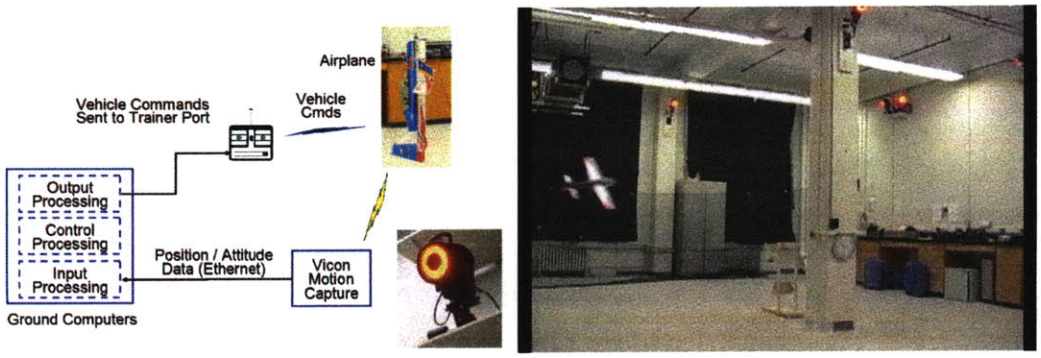

Figure 2-3: RAVEN system architecture (left) and the RAVEN flight space (right).

sition back to hover. Given the varying dynamics, some type of trajectory generation/optimization tool must be used to select a dynamically feasible path to the goal.

2.5 The RAVEN Testbed

Figure 2-3 presents the control architecture used for all of the flight experiments in this thesis. The system can support cooperative missions of up to ten (10) vehicles flying simultaneously [5] as well as switched-mode aerobatic flight [21]. A key feature of RAVEN is the high-precision motion capture system that can accurately track all vehicles in the room in real-time. The current architecture can utilize either a Vicon MX system or a Motion Analysis Raptor system. With lightweight reflective balls attached to each vehicle's structure, the Vicon motion capture system can measure the vehicle's position and attitude information at rates up to 120 Hz, with approximately a 10 ms delay, and sub-mm accuracy [6].

Flight control commands are computed using ground-based computers at rates that exceed 50 Hz and sent to the vehicles via standard Radio Control (R/C) transmitters. An important feature of this setup is that small, inexpensive, essentially unmodified, radio-controlled vehicles can be used. This enables researchers to avoid being overly conservative during flight testing. The computer configuration is shown in Figure 2-3 (left), with input (vehicle and environment state estimation), planning and control, and output (conversion to R/C commands) processing all done in linked

ground computers, as if it were being done onboard.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Quaternion Attitude Controller

3.1 Introduction

This chapter describes the baseline attitude and velocity controllers developed for the flight vehicles listed in Chapter 2. An important consideration in designing an aircraft attitude controller is the choice of attitude representation. Use of Euler angles (yaw, pitch, roll) is widespread in aerospace due in large part to the physical intuition of the system. However, Euler angles suffer from a set of singularities known in flight control as "gimbal lock", making them impractical for use in tracking arbitrary 3D attitudes [22]. Quaternions offer a singularity-free solution and have special properties that can be exploited for the design of a simple, generalized attitude controller. The algorithms presented in this chapter form the core baseline controllers for all three flight vehicles and have been thoroughly tested through many hours of indoor flight.

The chapter proceeds as follows. Section 3.2 serves as a brief primer on quaternion rotations, Section 3.3 outlines the development of a general 3D quaternion attitude controller, and Section 3.4 describes the outer-loop horizontal velocity controller used to generate attitude commands.

3.2 Quaternion Rotations

A quaternion can be thought of as a complex number with one real (scalar) part and three imaginary parts that form a vector:

$$\vec{q} = [\ \underbrace{q_0}_{\text{scalar}},\ \underbrace{q_x\hat{\mathbf{i}},\ q_y\hat{\mathbf{j}},\ q_z\hat{\mathbf{k}}}_{\text{vector}}\ ] \tag{3.1}$$

Quaternions are used widely as computationally efficient descriptors of three-dimensional rotations. The following is intended as a brief summary of quaternion rotations, a more thorough treatment can be found in [23]. A quaternion essentially represents an axis-angle rotation. Consider a rotation of θ radians about an axis described by the unit vector $[r_x, r_y, r_z]$. The quaternion describing the rotation would simply be:

$$\vec{q_r} = \left[\ \cos\left(\frac{\theta}{2}\right),\ r_x \sin\left(\frac{\theta}{2}\right)\hat{\mathbf{i}},\ r_y \sin\left(\frac{\theta}{2}\right)\hat{\mathbf{j}},\ r_z \sin\left(\frac{\theta}{2}\right)\hat{\mathbf{k}}\ \right] \tag{3.2}$$

Note that by definition $\|\vec{q}\|_2 = \sqrt{q_0^2 + q_x^2 + q_y^2 + q_z^2} = 1$. Sequential rotations are achieved by quaternion multiplication (denoted herein by $\otimes$) which is a simple algebraic operation similar to the cross product [23]. Consider rotating by a first quaternion $\vec{q_1}$, then again by a second quaternion $\vec{q_2}$ (where the second rotation is relative to the *previously rotated* frame). The quaternion representation of this set of rotations is:

$$\vec{q}_{total} = \vec{q_2} \otimes \vec{q_1} \tag{3.3}$$

The quaternion rotation operation is more efficient than matrix methods (like the direction cosine matrix) due to its small number of arithmetic operations. The inverse of a quaternion rotation is the quaternion's complex conjugate $\vec{q^*} = [q_0 - q_x\hat{\mathbf{i}} - q_y\hat{\mathbf{j}} - q_z\hat{\mathbf{k}}]$. Thus $\vec{q_1} \otimes q_1^*$ corresponds to no rotation.

3.3 Quaternion-Based Attitude Control

The sequential rotation properties of quaternions can be used to devise a generalized attitude controller for flight vehicles. The algorithm presented here is similar to that in [4]. Consider some desired aircraft attitude that is specified as a quaternion rotation from the global frame $\vec{q_d}$. This desired rotation can be constructed intuitively with an axis and an angle as in (3.2). The actual *measured* attitude of the aircraft can also be represented as a quaternion rotation from the global frame $\vec{q_a}$. Now consider that the desired attitude $\vec{q_d}$ can be constructed as the measured attitude $\vec{q_a}$ sequentially rotated by some error quaternion $\vec{q_e}$. Since quaternion rotations are sequential, the error quaternion represents a rotation from the *aircraft frame*:

$$\underbrace{\vec{q_d}}_{\text{global frame}} = \underbrace{\vec{q_e}}_{\text{aircraft frame}} \otimes \underbrace{\vec{q_a}}_{\text{global frame}} \tag{3.4}$$

The error quaternion can then be solved for explicitly using the conjugate of the measured attitude quaternion:

$$\underbrace{\vec{q_e}}_{\text{aircraft frame}} = \underbrace{\vec{q_d}}_{\text{global frame}} \otimes \underbrace{\vec{q_a^*}}_{\text{global frame}} \tag{3.5}$$

The error quaternion $\vec{q_e}$ represents the rotation required to get from the measured attitude to the desired attitude. Since the rotation is *from the aircraft frame*, the x, y and z components of $\vec{q_e}$ correspond to the rotations needed about the x, y and z body axes of the aircraft. Thus the three components correspond directly to the required aileron, elevator, and rudder commands without any transformation. The scalar part of $\vec{q_e}$ represents the angle through which the aircraft must be rotated and is thus proportional to the amount of control effort required.

The following procedure outlines the implementation of a quaternion attitude controller based on the above result:

1. Set the desired attitude $\vec{q_d}$, which can be done by some higher-level control loop.
2. Obtain the aircraft's measured attitude $\vec{q_a}$ from either an onboard IMU or mo-

tion capture system.

3. Calculate the error quaternion $\vec{q_e} = \vec{q_d} \otimes \vec{q_a^*}$
4. From $\vec{q_e}$, compute the error angle $\theta_e = 2\arccos(q_{e_0})$ and the error rotation axis components:

$$r_x = \frac{q_{e_x}}{\sin(\theta_e/2)} \tag{3.6a}$$

$$r_y = \frac{q_{e_y}}{\sin(\theta_e/2)} \tag{3.6b}$$

$$r_z = \frac{q_{e_z}}{\sin(\theta_e/2)} \tag{3.6c}$$

5. Split the errors into the required aileron, elevator, and rudder commands δ_a, δ_e and δ_r:

$$\delta_a = K_{p_a}\ \theta_e\ r_x \tag{3.7a}$$

$$\delta_e = K_{p_e}\ \theta_e\ r_y \tag{3.7b}$$

$$\delta_r = K_{p_r}\ \theta_e\ r_z \tag{3.7c}$$

where the K_p's are positive proportional gains.

6. The proportional commands above are augmented with derivative terms using the measured body rates p, q, and r to form the complete proportional-derivative (PD) attitude controller:

$$\delta_a = K_{p_a}\ \theta_e\ r_x - K_{d_a}\ p \tag{3.8a}$$

$$\delta_e = K_{p_e}\ \theta_e\ r_y - K_{d_e}\ q \tag{3.8b}$$

$$\delta_r = K_{p_r}\ \theta_e\ r_z - K_{d_r}\ r \tag{3.8c}$$

where the K_d's are positive derivative gains.

While denoted above as aileron, elevator, and rudder, the control commands are easily generalizable to any actuator set (*e.g.*, the three-wing tailsitter or quadrotor helicopter) so long as the commands are mapped to the appropriate control axes.

Along with its applicability to any generic flight vehicle, another key feature of this attitude controller is its ability to track arbitrary 3D attitudes without singularities of any kind (such as those inherent in Euler angle control methods).

3.4 Outer-Loop Horizontal Velocity Controller

With the attitude controller in place, a simple outer-loop horizontal velocity controller can be added to generate the desired attitude $\vec{q_d}$. The controller is based about the aircraft in a hover configuration, where thrust is being generated vertically to counteract the acceleration of gravity. Tilting the vehicle in any direction will generate a horizontal component of thrust and the vehicle will translate as a result. For example, if a positive x-velocity is desired, the aircraft should be tilted in the positive x-direction. This corresponds to a rotation about the positive y-axis.

To calculate the desired quaternion $\vec{q_d}$ given the desired x- and y-velocities $[v_{d_x}, v_{d_y}]$ and the actual measured x- and y-velocities $[v_{a_x}, v_{a_y}]$, the velocity errors are calculated:

$$v_{e_x} = v_{d_x} - v_{a_x} \tag{3.9a}$$

$$v_{e_y} = v_{d_y} - v_{a_y} \tag{3.9b}$$

The errors are then used to determine the desired quaternion rotation $\vec{q_d}$, with integral action added to the controller:

$$\vec{q_d} = \left[1.0,\ \left(-K_{p_{v_y}} v_{e_y} - K_{i_{v_y}} \int v_{e_y} dt\right)\hat{\mathbf{i}},\ \left(K_{p_{v_x}} v_{e_x} + K_{i_{v_x}} \int v_{e_x} dt\right)\hat{\mathbf{j}},\ 0\hat{\mathbf{k}}\right] \tag{3.10}$$

Finally, the quaternion is normalized to ensure that $\|\vec{q_d}\|_2 = 1$. The combined proportional-integral (PI) horizontal velocity controller and quaternion attitude controller forms the baseline controller for the quadrotor, three-wing, and Clik aerobatic aircraft used throughout this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

$\mathcal{L}_1$ Adaptive Output Feedback

This section provides a brief overview of $\mathcal{L}_1$ adaptive output feedback control [11, 24], which is the adaptive algorithm used herein. Derivation of the predicted time-delay margin is presented, as well as the expected closed-loop system response. Challenges in choosing the $\mathcal{L}_1$ control parameters $C(s)$ and $M(s)$ are discussed as a motivation for the proposed design process.

4.1 Overview

Figure 4-1 shows the block diagram for the adaptive controller. The disturbance $d(s)$ is used to represent any type of non-linear disturbance, and thus can represent not only external disturbances but also changes in the plant $A(s)$ due to parameter variations or actuation failures. Note that if $C(s) = 1$, a simple single-input single-output MRAC controller is recovered. If this were the case, $\hat{\sigma}$ (the adaptive signal) would simply be "whatever it takes" to make the output of $A(s)$ match the output of $M(s)$. The τ block represents a typical sensor measurement time delay, and will be used later in the characterization of robustness metrics.

Adding the low-pass filter $C(s)$ does two important things. First, it limits the bandwidth of the control signal u being sent to the plant. This prevents high-bandwidth oscillatory control signals (as are often seen with fast-adapting MRAC controllers) from being commanded. Second, the portion of $\hat{\sigma}$ that gets sent into

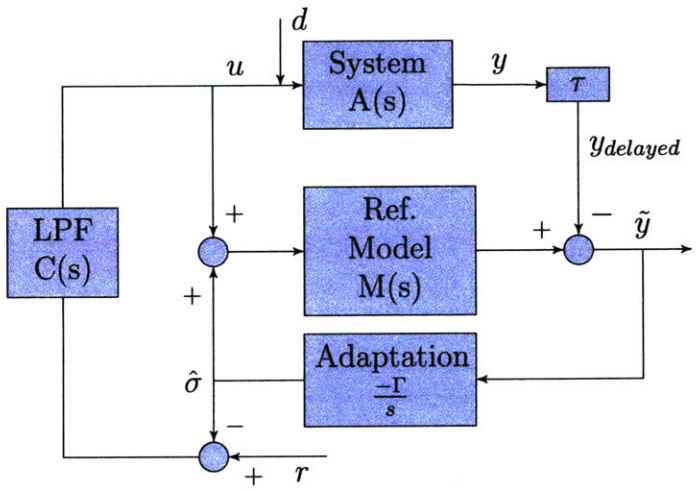

Figure 4-1: $\mathcal{L}_1$ adaptive output feedback control block diagram. Note that if $C(s) = 1$, a simple single-input single-output MRAC controller is recovered.

the reference model is the *high-frequency portion* (note that the low-pass version is subtracted from the full signal before being sent to the reference model $M(s)$). This signal, in a sense, corresponds to the portion of the disturbance $d(s)$ that can not be canceled given the limited actuator bandwidth. The fact that this is sent into the reference model $M(s)$ along with the reference signal implies that the output of $M(s)$ is the *achievable* system output, a realistic goal that the system should be able to match given its bandwidth constraints.

4.1.1 Control Parameters

The user-specified parameters of the $\mathcal{L}_1$ controller are the low-pass filter $C(s)$, the reference model $M(s)$, and the adaptation rate Γ. It is clear that $C(s)$ should be chosen such that its bandwidth does not exceed that of the available actuators. The adaptation rate Γ is essentially the gain of the adaptive estimator, and since the control signal is low-pass filtered a very large value can be used (for example, $\Gamma = 10000$ in the flight tests in Section 4.4). Since the controller must still be implemented in real-time on a computer, Γ is limited in practice by the stability of the numerical integration which is determined largely by available computational capabilities. As will be discussed below, the choice of $M(s)$ is not so straightforward since it does not

act in the same way as the reference model in MRAC.

4.1.2 Predicted Time Delay Margin

It is helpful to be able to predict the adaptive controller's margin to a time delay on the output measurement $y(s)$ (corresponding to some known sensor delay). While this analysis has previously been done for the general $\mathcal{L}_1$ control setup [12], it has not been shown explicitly for the output feedback case considered here. Since the output feedback system is comprised of SISO linear blocks, the time delay margin of the system can be calculated as the ratio of phase margin to cross-over frequency of the appropriate system. From Figure 4-1, the system of interest is the system whose input is y_{delayed} and whose output is y, assuming that $r = 0$ and $d = 0$. It is easiest to analyze this system using state space techniques instead of transfer functions. Let $[A_A, B_A, C_A, 0]$, $[A_C, B_C, C_C, 0]$, and $[A_M, B_M, C_M, 0]$ be the state space representations of $A(s)$, $C(s)$, and $M(s)$, respectively. It can be verified that the system with input y_{delayed} and output y has the following state space representation:

$$A = \begin{bmatrix} A_A & 0 & B_A C_C & 0 \\ 0 & A_M & B_M C_C & B_M \\ 0 & 0 & A_C & -B_C \\ 0 & -\Gamma C_M & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \Gamma \end{bmatrix} \quad C = \begin{bmatrix} C_A & 0 & 0 & 0 \end{bmatrix} \quad D = [0] \tag{4.1}$$

The time delay margin is then calculated by taking the ratio of the phase margin to the cross-over frequency, both of which can be deduced from a Bode plot of the system. Determination of the time-delay margin using this method has been confirmed both in simulation and experiment.

4.1.3 Expected Closed-loop Response

An excellent analysis of the $\mathcal{L}_1$ output feedback system is provided in [[13]] where it is shown that, if the disturbance is known exactly (i.e. Γ is sufficiently large such that the adaptive estimator is doing its job perfectly), the expected closed-loop response

becomes:

$$y(s) = \underbrace{H(s)C(s)r(s)}_{\text{Response to reference } r(s)} - \underbrace{H(s)(1 - C(s))d(s)}_{\text{Response to disturbance } d(s)} \tag{4.2}$$

where

$$H(s) = \frac{A(s)M(s)}{C(s)A(s) + (1 - C(s))M(s)} \tag{4.3}$$

Recall that in MRAC the expected closed-loop response is simply $y(s) = M(s)r(s)$. Now, even if the disturbance is known perfectly, the expected closed-loop response is a complicated function of $A(s)$, $C(s)$, and $M(s)$.

4.1.4 Selection of $C(s)$ and $M(s)$

As seen from Eq. 4.2, the inclusion of a low-pass filter in the control path obscures the expected closed-loop dynamics. Since $M(s)$ no longer acts as a reference model, it is not obvious how its choice affects the system response. In other words, while it is a design parameter of the $\mathcal{L}_1$ controller, it cannot simply be chosen as the reference model like in MRAC. While the bandwidth of $C(s)$ should be upper-bounded by the bandwidth of the corresponding actuator, there is no intuitive lower bound.

This brings about the need for a systematic method of choosing the filters $C(s)$ and $M(s)$ based on desired performance and robustness metrics. Performance measures may include transient response characteristics such as rise time, overshoot, or cross-over frequency. Robustness measures may include metrics such as time delay margin and disturbance rejection.

4.2 Design Process

In this section, a design process is proposed by which $C(s)$ and $M(s)$ are selected in a systematic way based on desired performance and robustness metrics. This is achieved by performing multi-objective optimization on a weighted cost function comprised of these metrics, which also requires basic system identification of the baseline plant $A(s)$.

4.2.1 Specifying Performance and Robustness Metrics

The first step in this design process is to identify performance and robustness metrics relevant to the control task at hand. For the dynamic control of indoor autonomous flight vehicles, three important metrics are considered. **Transient performance** characteristics such as rise time and overshoot provide intuitive measures of the closed-loop system response. For instance, these can be chosen to set the aggressiveness of the controller based on mission requirements. For application to real-world physical systems, **time delay margin** is a very important measure of robustness. Some minimum time delay margin must be achieved based on known delays in the sensing, estimation, computation, and actuation of the physical system. Finally, the general goal of adaptive control is to maintain nominal performance in lieu of disturbances such as actuation failures or plant parameter variations. Thus, it is important to somehow specify a desired degree of **disturbance rejection**.

Transient Performance

As shown in Eq. 4.2, the expected closed-loop response $y(s)$ to the reference input $r(s)$ is given by the transfer function $H(s)C(s)$. Thus, the design process should shape the transient response of this system. The response can be affected by weighting a combination of standard step-response characteristics which may include rise time, overshoot, or settling time. Frequency-domain metrics may also be used such as the bandwidth of $H(s)C(s)$, which in this case would roughly represent the frequency range through which the output $y(s)$ matches the reference input $r(s)$. Calculation of these metrics is fairly straight-forward either by simulating the closed-loop system (more time-consuming) or via second-order approximation (less accurate).

Time Delay Margin

The time delay margin of the adaptive controller can be calculated as in Section 4.1.2. As will be seen, $A(s)$ will often represent the nominal *closed-loop* system which may include internal feedback loops from the baseline controller. It should be noted that,

for analytical simplicity, the margin considered here only considers a time delay on the output signal $y(s)$ being sent back to the adaptive controller, and not on any of the feedback signals internal to $A(s)$. Even so, this time delay margin still provides a good indication of how robust the adaptive controller will be to feedback delays in general. In addition, since only the adaptive controller is being designed here, use of this measure also offers analytical separation between the effect of time delay on the adaptive controller and the potentially complex effects on the nominal system.

Robustness to Disturbances

As shown in Eq. 4.2, the expected closed-loop response $y(s)$ to the disturbance $d(s)$ is given by the transfer function $H(s)(1 - C(s))$. Note that $d(s)$ can be an arbitrary signal (restricted only in its Lipschitz norm [13]), thus it can represent many types of disturbances including actuation failures, parameter variations, and exogenous factors. To minimize the effects of these disturbances, the norm of $H(s)(1 - C(s))$ should be minimized. The $\mathcal{L}_1$ norm has typically been used, [9, 10] so the measure of disturbance rejection used here is chosen to be $\|H(s)(1 - C(s))\|_{\mathcal{L}_1}$, a smaller norm corresponding to increased robustness to disturbances.

4.2.2 System Identification

Since the expected closed-loop response from Eq. 4.2 includes the nominal system $A(s)$, some knowledge of this system is required to design the adaptive controller. As will be seen in Section 4.3, $A(s)$ can be taken to be the nominal closed-loop system, which includes the baseline controller. Since this system is typically stable and well-behaved by design, system identification from experimental data is relatively easy.

Take, for example, the quadrotor helicopters used in Section 4.4. The system $A(s)$ in this case represents the closed-loop response of the system from x-velocity reference command to x-velocity measured output. Using data from routine flight tests, the

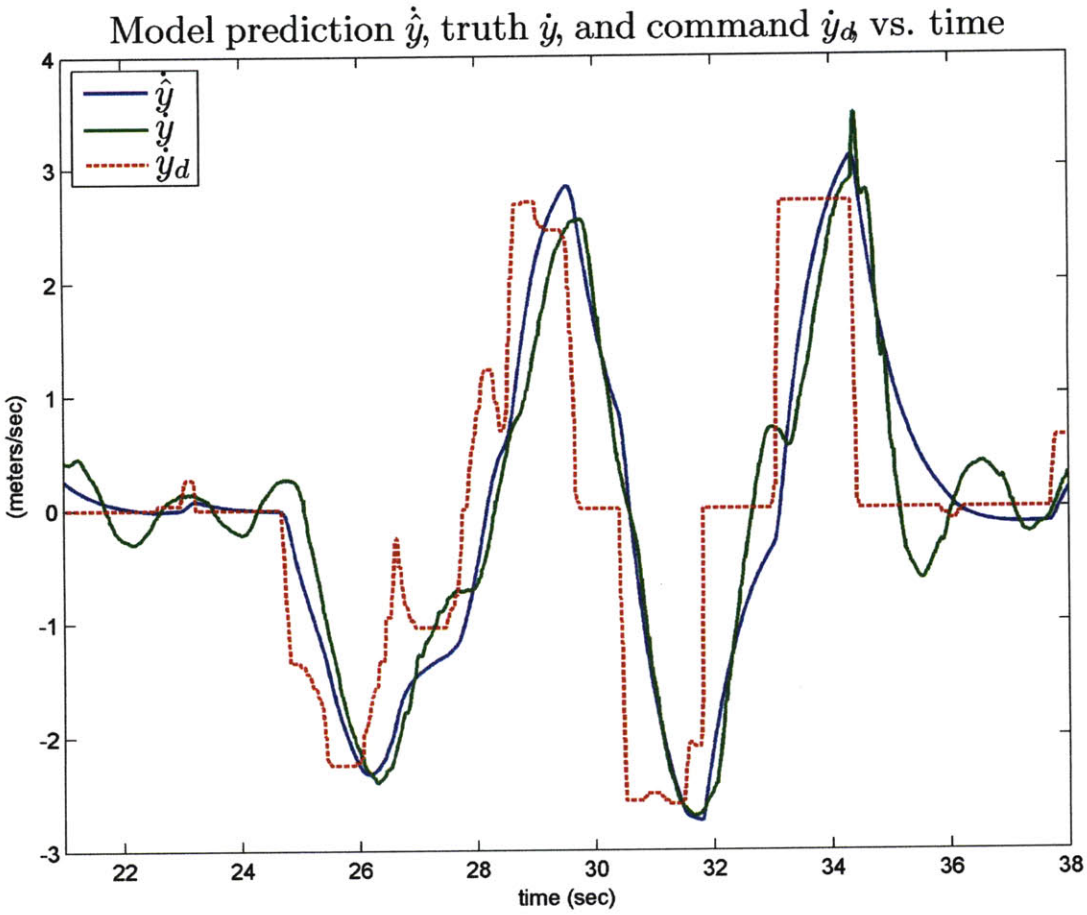

Figure 4-2: System identification results for quadrotor closed-loop velocity system. User-generated reference command (red), measured output (green), and predicted output based on system ID (blue).

following second-order system model was fit using an ARMAX-type regression:

$$\frac{V_{x_{meas}}}{V_{x_{ref}}}(s) = \frac{k(s - z_1)}{(s - p_1)(s - p_2)} = \frac{1.80(s + 0.44)}{(s + 0.89)(s + 0.89)} \tag{4.4}$$

As can be seen in Figure 4-2, this simple second-order model adequately characterizes the closed-loop response.

4.2.3 Multi-Objective Optimization

The desired performance and robustness metrics are combined into a cost function that can be minimized in an effort to calculate $C(s)$ and $M(s)$. The optimization process must be provided with the cost function, relevant constraints (like system

stability), a model of $A(s)$ from system identification, and some parametrization of $C(s)$ and $M(s)$. Figure 4-3 shows a general diagram of the optimization process.

Cost Function and Constraints

The cost function considered here is a simple weighted combination of the performance and robustness metrics listed above. For example, a possible cost function might be:

$$J = \alpha_1(\text{Rise Time}) + \alpha_2(\text{Overshoot}) + \alpha_3\left(\frac{1}{\text{TD margin}}\right) + \alpha_4\left(\|H(s)(1-C(s))\|_{\mathcal{L}_1}\right) \quad (4.5)$$

Since the cost function is to be minimized, α_i represents the *penalty* on the associated metric. Two constraints must be considered. First, the bandwidth of $C(s)$ must be limited to the bandwidth of the associated actuator. In the case of the quadrotor velocity controller, the "actuator" is the closed-loop system given by (4.4), thus $C(s)$ must be limited in bandwidth to that of (4.4). The second constraint to be considered is the stability of the expected system response $H(s)C(s)$. One simple way to implement this constraint is to augment the cost function with a term that is arbitrarily large if the system is unstable. The test for stability used in the experimental validation below is simply a check of whether the roots of $H(s)C(s)$ are strictly negative.

Filter Parametrization

$C(s)$ and $M(s)$ must be parameterized such that the cost function can be minimized over these parameters. A particularly straightforward parametrization is:

$$C(s,\hat{c}) = \frac{\hat{c}}{s+\hat{c}} \quad (4.6)$$

$$M(s,\hat{m}) = \frac{\hat{m}}{s+\hat{m}} \quad (4.7)$$

Using this parametrization, each transfer function has only one parameter associated with it. While this may limit performance in some ways, it greatly simplifies the optimization process and provides for an intuitive initial design iteration. Use of

higher-order filters is an open topic of research [13]. It will be shown in Section 4.4, however, that these simple filter parameterizations are intuitive, easy to implement, and work quite well in practice.

Solution Methods and Limitations

Once the cost function, constraints, and parameterized solution forms are specified, the designer is free to use any constrained optimization technique to solve for $C(s)$ and $M(s)$. One obvious drawback to this process, though, is that the cost function presented is almost certainly non-convex with respect to the parameterized transfer functions. This is not surprising considering that characteristics like overshoot and time-delay margin are summed in the same function. Typical solvers, like Matlab®'s `fmincon`, work well for a small number of parameters, but exhibit poor convergence properties as the parametrization complexity increases.

Using filters in the form of (4.7) is thus advantageous since typical solvers usually converge to a global minimum. It is easy to then verify this minimum if necessary using exhaustive search methods when there are only two parameters. Furthermore, the cost as a function of the two parameters ($\hat{c}$ and $\hat{m}$) can be visualized using a contour plot, making the process more intuitive to the designer. Ongoing work (see Section 4.5) aims to address the non-convexity issue by attempting to cast each performance and robustness metric as a linear matrix inequality (LMI) constraint and performing the optimization with much more efficient search methods. This has been done for the general $\mathcal{L}_1$ adaptive control formulation [25], but has not been extended to the output feedback case considered here.

4.3 Experimental Setup

The quadrotor helicopter described in Section 2.2 is used to experimentally validate the above design process. While the dynamics of quadrotor can be modeled reasonably well, its four motors are subject to performance variations and partial failures. The goal of adaptive control in this case is to make quadrotor flight more robust to these

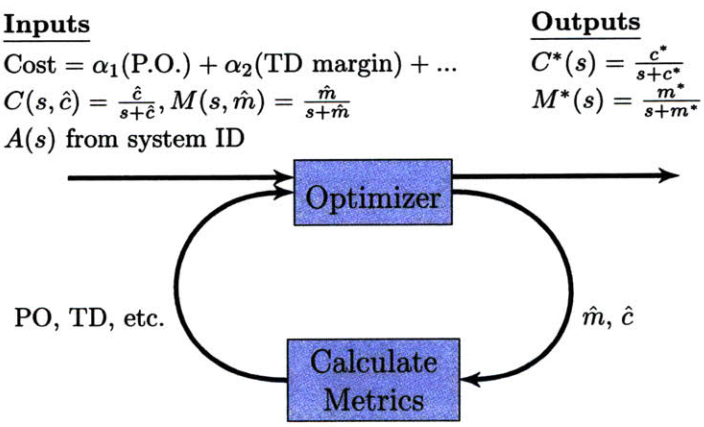

Figure 4-3: Multi-objective optimization diagram

types of actuator failures. Since the vehicles are controlled autonomously, failures can be simulated mid-flight by scaling the control commands to individual rotors. The RAVEN testbed described in Section 2.5 is used to measure the position, attitude, and rates of the quadrotor in realtime.

The nominal controller for the quadrotor consists of an outer-loop velocity controller wrapped around an inner-loop attitude controller. The vehicle's translational velocity in the X-Y (horizontal) plane is affected by commanding an appropriate attitude. For example, if a positive x-velocity is desired, the outer-loop velocity controller will command an attitude that "tilts" the vehicle in the x-direction, and the inner-loop attitude controller will attempt to track this commanded attitude. Independent controllers for x- and y-velocity are used, and attitude commands are combined and sent to a single inner-loop attitude controller. The velocity controller is linear Proportional+Integral, while the attitude controller is quaternion-based linear Proportional+Derivative similar to that presented in [[4]].

Figure 4-4 shows the baseline x-velocity controller for the quadrotor helicopter augmented with an $\mathcal{L}_1$ adaptive controller. The system in the dashed box represents $A(s)$, the baseline closed-loop system that takes a velocity reference input and produces some measured velocity as an output as described above. This is the system identified in (4.4). Note that the $\mathcal{L}_1$ controller is completely external to the nominal closed-loop system, augmenting the velocity reference command sent to the baseline

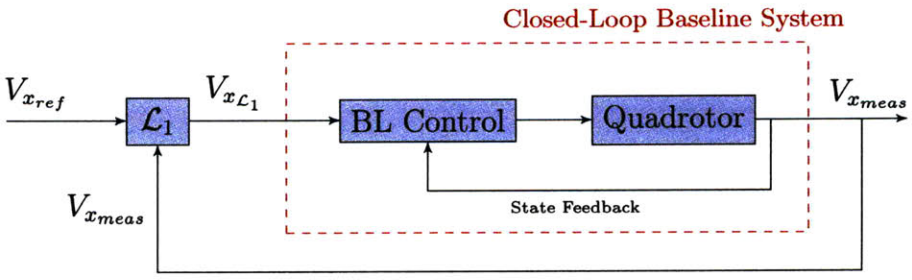

Figure 4-4: Setup for the quadrotor x-velocity controller with $\mathcal{L}_1$ adaptive augmentation. The system in the dashed lines represents $A(s)$, the baseline closed-loop system.

controller.

4.4 Experimental Results

The design process described in Section 4.2 would normally be carried out by weighting the appropriate performance and robustness metrics in accordance with mission requirements. However, to validate the design process experimentally, flight results are shown that vary *only one* parameter of the cost function at a time. In doing so, modifications to individual components of the cost function can be clearly verified to have the expected result. The cost function chosen is:

$$J = \alpha_1(\text{Rise Time}) + \alpha_2\left(\frac{1}{\text{TD margin}}\right) + \alpha_3(\|H(s)(1-C(s))\|_{\mathcal{L}_1}) \tag{4.8}$$

Thus α_1 penalizes slow transient performance, α_2 penalizes small time delay margin, and α_3 penalizes poor disturbance rejection. Three flight scenarios are used to test these metrics with the quadrotor: nominal response to a step command, response to a step command with time-delayed sensor measurements, and response to a single-rotor actuator failure to 50% of its original capacity.

Note that while the $\mathcal{L}_1$ adaptive controller is acting on the velocity controller of the baseline system, a simple proportional position controller has been added as an outer-loop. This is done primarily due to space constraints of the testing area, since step commands in position are more predictable than steps in velocity. Position

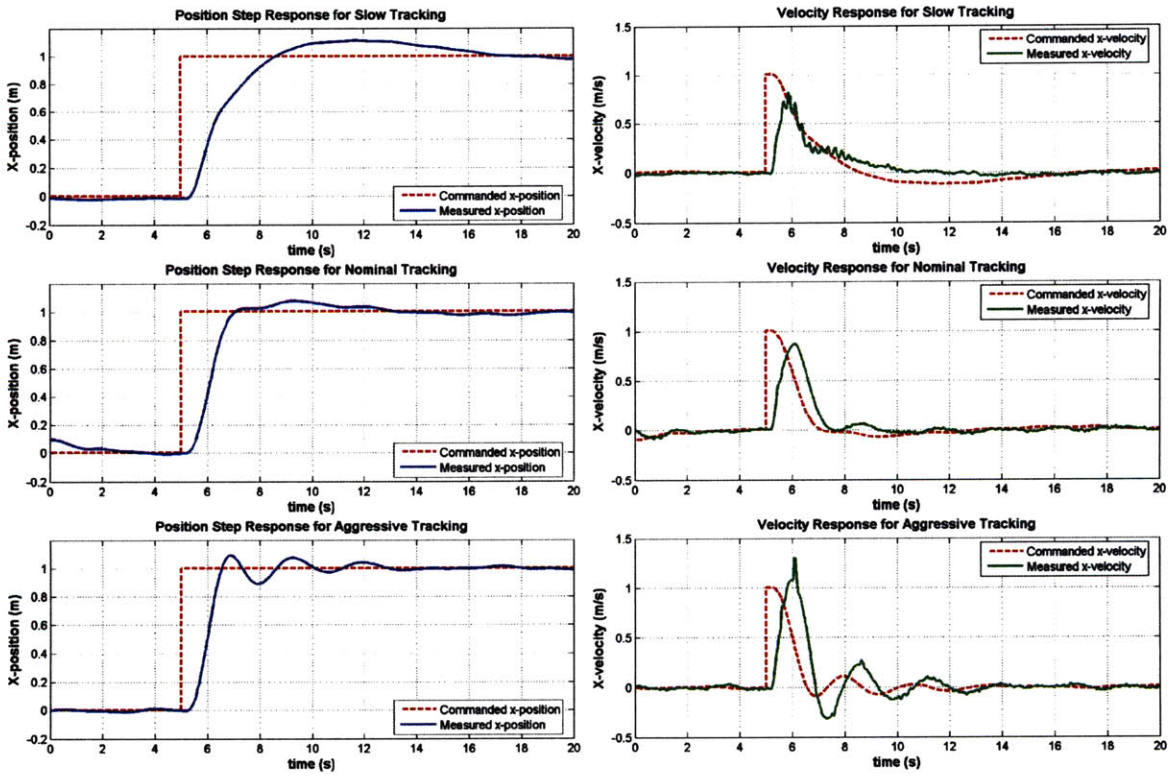

Figure 4-5: Nominal transient performance flight results, position response (left) and velocity response(right). Parameters chosen for slow tracking (top), nominal tracking (middle), aggressive tracking (bottom).

response also provides insight as to how adaptation on the velocity controller impacts position control. As a result, the following will show not only the velocity response of the system, but also the position response.

4.4.1 Flight Results: Nominal Transient Performance

To test nominal transient performance, α_1 was increased while keeping α_2 and α_3 constant, corresponding to an increasing penalty on rise time. Figure 4-5 shows that the expected behavior is achieved. As the penalty is increased, the position and velocity responses show faster rise times at the cost of more overshoot and less damping.

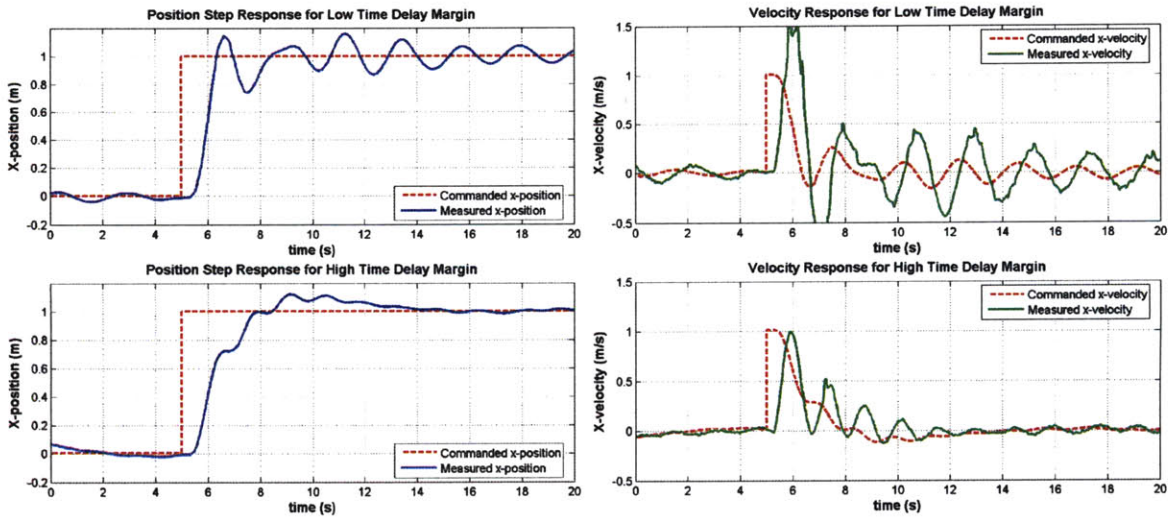

Figure 4-6: Measurement time delay flight results, position response (left) and velocity response(right). Parameters chosen for less time delay margin (top), and more time delay margin (bottom).

4.4.2 Flight Results: Robustness to Time Delay

To test robustness to time delay, α_2 was increased while keeping α_1 and α_3 constant. As can be seen in Figure 4-6, the increased penalty yields a slower, more well-damped response in lieu of a 90ms sensor measurement delay while the smaller penalty yields a marginally stable response. It should be noted that measurement delay is applied to *all* feedback loops (including the inner-loop controllers) as would be the case for realistic time-delayed systems. Recall from Section 4.1.2 that the time delay analysis for the $\mathcal{L}_1$ controller only considered delay on the output feedback signal, not the internal feedback loops. The fact that the expected result is still achieved here implies that the $\mathcal{L}_1$ time delay analysis yields the correct trend even when all of the feedback loops are delayed.

4.4.3 Flight Results: Robustness to Actuator Failure

To test robustness to disturbances, α_3 was increased while keeping α_1 and α_2 constant, corresponding to an increased penalty on poor disturbance rejection. In Figure 4-7, it is clear that the increased penalty yields a stable response after a 50% single-rotor failure, while the lower penalty leads to an unstable response in which the

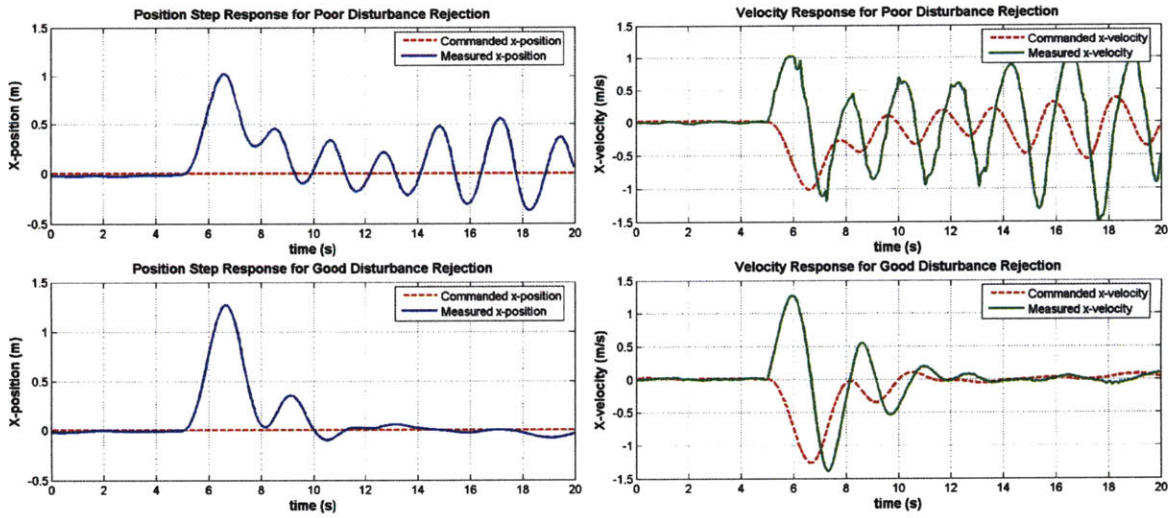

Figure 4-7: Actuator failure flight results, position response (left) and velocity response(right). Parameters chosen for poor disturbance rejection (top), and good disturbance rejection (bottom).

vehicle eventually crashes. While the increased penalty yields stability, it comes at the cost of a slower and more conservative response (note that it initially has a greater maximum error in position). This result highlights the fundamental tradeoff between performance and robustness.

4.4.4 Comparison to Baseline Controller

As a useful benchmark, flight tests are performed comparing the baseline controller to the augmented $\mathcal{L}_1$ adaptive system. To provide a fair comparison, the adaptive controller is tuned so that the nominal step response for both systems is sufficiently similar (see Figure 4-8 top). The same adaptive controller is then compared to the baseline controller in the measurement delay (Figure 4-8 middle) and actuator failure (Figure 4-8 bottom) scenarios. In the time delay case, the adaptive controller shows noticeably improved tracking in both position and velocity. The same can be said for the actuator failure, as the adaptive controller yields a smaller maximum error in position as well as improved damping in the transient response for both position and velocity. These results confirm that adaptive control can offer useful performance improvements in the context of indoor autonomous flight.

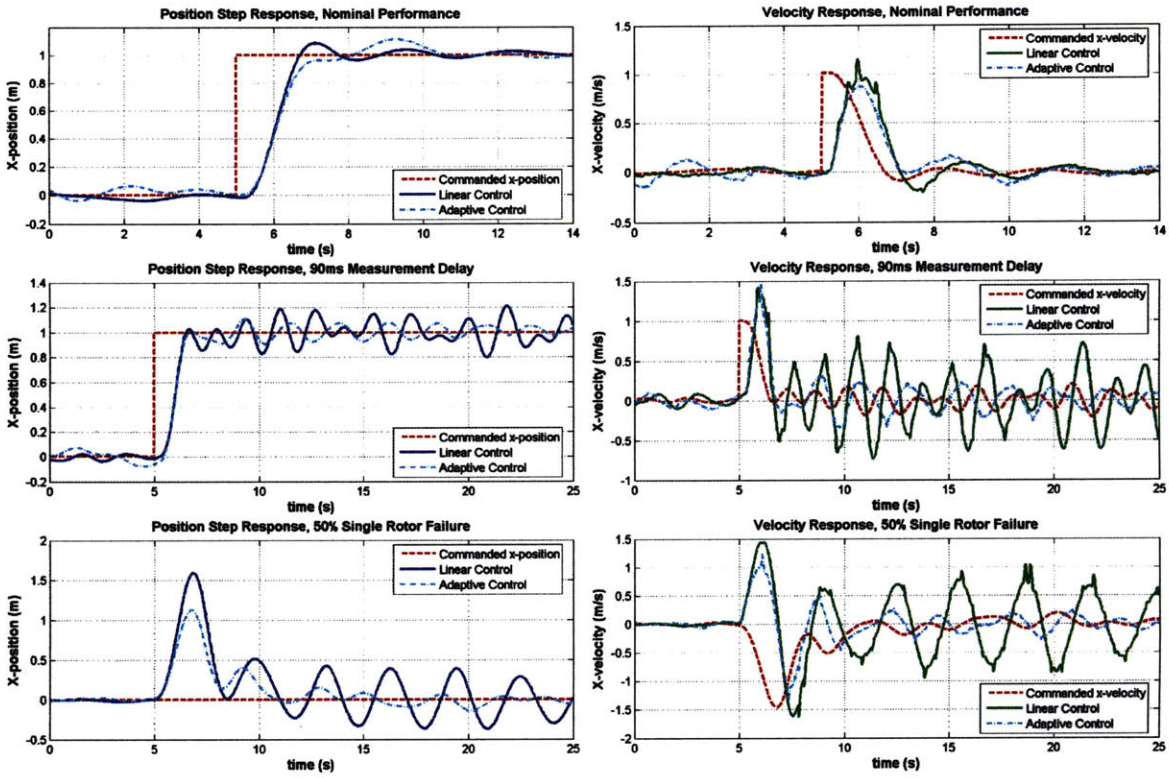

Figure 4-8: Flight test comparison of baseline linear controller to $\mathcal{L}_1$ adaptive controller, position response (left) and velocity response (right). While both show similar nominal performance (top), the $\mathcal{L}_1$ adaptive controller shows improved performance for both a 90ms measurement delay (middle) and a 50% single-rotor failure (bottom).

4.5 Summary

This chapter attempts to provide a systematic design process for the use of $\mathcal{L}_1$ adaptive output feedback control in realistic flight control applications. The proposed method provides the control designer with an intuitive method linking relevant performance and robustness metrics to the selection of the $\mathcal{L}_1$ parameters $C(s)$ and $M(s)$. This design process represents a step in the direction of more readily applying $\mathcal{L}_1$ adaptive control to real-world flight systems and taking advantage of its potential benefits.

Flight test results verify the process for an indoor autonomous quadrotor, demonstrating that variations in the specified cost function produce the expected and desired physical responses. In flight tests comparing it with the baseline linear controller, the augmented $\mathcal{L}_1$ adaptive system shows definite performance and robustness improvements. Also, adaptive augmentation is shown to help enable aggressive flight for

a fixed-wing aerobatic aircraft. Both of these results confirm the potential of $\mathcal{L}_1$ adaptive control as a useful tool for autonomous aircraft.

Chapter 5

MIMO $\mathcal{L}_1$ Adaptive Control

5.1 Introduction

In more complex flight scenarios the need arises for more than just *single*-input *single*-output adaptive augmentation. With more information available and more inputs to the system, it is apparent that *multi*-input *multi*-output (MIMO) adaptive control has the potential to yield increased performance and robustness for complex systems. Chapter 4 describes the application of single-input single-output (SISO) $\mathcal{L}_1$ adaptive control to a quadrotor helicopter. This chapter describes the application of multi-input multi-output (MIMO) $\mathcal{L}_1$ to the three-wing tailsitter, an aircraft with more complex control challenges as described in Section 2.3.

The chapter proceeds as follows. Section 5.2 gives an overview of MIMO $\mathcal{L}_1$ adaptive control and the difficulties of applying it to a non-linear baseline controller, Section 5.3 describes how the reference model is selected from system identification, Section 5.4 lists some modifications that are made to the standard adaptive law, Section 5.5 presents simulation results of the MIMO controller, Section 5.6 presents flight results, and Section 5.7 gives future directions of research.

5.2 Inner-loop Body Rate Adaptation

In Refs. [14] and [15], standard state-feedback $\mathcal{L}_1$ is extended to the MIMO case. In doing so, the authors assume a general form of the system dynamics similar to:

$$\dot{x} = Ax(t) + B\Lambda(t)\left(u(t) + d(t)\right) \tag{5.1}$$

where $x \in \Re^{n\times 1}$, $u \in \Re^{m\times 1}$, d is some disturbance with known Lipschitz bound, and $\Lambda \in \Re^{n\times n}$ is a diagonal "failure matrix" of the form:

$$\begin{bmatrix} \Lambda_1 & 0 & 0 \\ 0 & \Lambda_2 & 0 \\ 0 & 0 & \ddots \end{bmatrix} \tag{5.2}$$

where upper and lower conservative bounds on Λ_i are known. Incorporating both input failures and a time-varying disturbance, the resulting general form is applicable to many realistic aerospace systems.

However, in Refs. [14, 15] the adaptive augmentation encompasses a *linear* closed-loop baseline controller on the example systems. This is necessary because the reference dynamics of the $\mathcal{L}_1$ adaptive controller must be linear. The linearity requirement is an issue if the closed-loop controller is non-linear. As is the case on the three-wing tailsitter, complex aircraft often require complex non-linear and/or ad-hoc control methods to achieve stable flight. In this situation it is very difficult to obtain a linear representation of the entire closed-loop dynamics without extensive MIMO system identification and approximation. The problem becomes how to cleanly augment the system with adaptive control when a non-linear baseline control scheme is in place.

To overcome the problem of augmenting the non-linear closed-loop system with an $\mathcal{L}_1$ adaptive controller, adaptation is instead applied to the inner-loop body rate subsystem. The body rate system inputs are the derivatives of the three control surface deflections $\dot{\delta}_1, \dot{\delta}_2$, and $\dot{\delta}_3$ and whose outputs are the three body angular rates, p, q and r. The system involves only the physical open-loop dynamics of the vehicle,

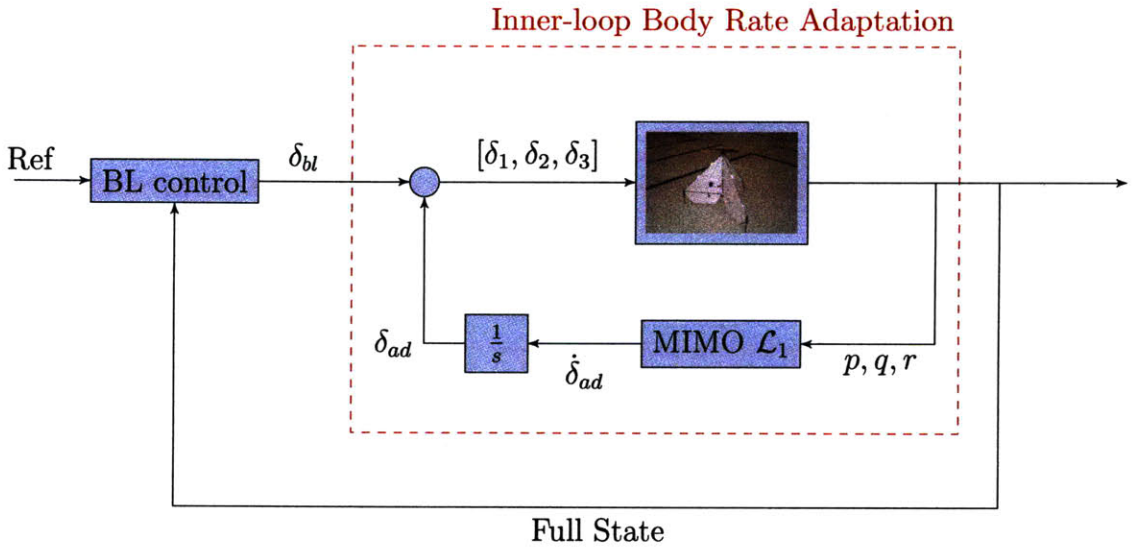

Figure 5-1: Diagram of inner-loop MIMO $\mathcal{L}_1$ body rate adaptation.

and thus is invariant to any specific baseline (BL) controller. The baseline controller still generates the three control deflections, but now the $\mathcal{L}_1$ controller monitors the response from deflection to body angular rate. The $\mathcal{L}_1$ controller then attempts to make the response match some nominal reference model in the case of some sort of physical system variation/failure (more details in the next section). Figure 5-1 shows the block diagram for the inner-loop body rate control structure.

Since it is the control deflections that directly affect the body angular rates, the *derivatives* of the control deflections directly affect the body angular accelerations. As will become more obvious in the next section, this is why the $\mathcal{L}_1$ system uses the derivatives of control deflections and not the deflections themselves. One consequence of using the time derivatives of deflection is that the adaptive output must be integrated before being added to the baseline control signals. The integration brings about the need for a slight modification to the adaptive laws which will be explained further.

5.3 Identifying a Reference Model

The inner-loop $\mathcal{L}_1$ adaptive system requires a reference model which it will use to compare to the measured system dynamics. Note that unlike the output feedback case described in Chapter 4, the MIMO version of $\mathcal{L}_1$ follows the reference model more directly. This greatly simplifies the task of choosing the $\mathcal{L}_1$ parameters as will be seen. In the MIMO case, the reference model will be chosen to simply be the nominal dynamics of the three-wing from $\{\dot{\delta}_1, \dot{\delta}_2, \dot{\delta}_3\}$ to $\{p, q, r\}$.

As mentioned in Chapter 2, the dynamics of the three-wing tailsitter are very difficult to model accurately from first principles due to the complex propeller flow and asymmetric control forces. System identification is instead used to fit a parametrized state-space model of the dynamics. The state-space model is parametrized as follows:

$$\begin{aligned}
\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} &= \underbrace{\begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{bmatrix}}_{A} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \underbrace{\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}}_{B} \begin{bmatrix} \dot{\delta}_1 \\ \dot{\delta}_2 \\ \dot{\delta}_3 \end{bmatrix} \\
\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} &= \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{C} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{D} \begin{bmatrix} \dot{\delta}_1 \\ \dot{\delta}_2 \\ \dot{\delta}_3 \end{bmatrix}
\end{aligned} \tag{5.3}$$

In this parametrized model, several assumptions are implicit. The dynamics matrix A is chosen to be diagonal to ignore coupling of body rates and simplify the identification process. The a_i's thus represent small, negative terms accounting for aerodynamic damping in the system. The input matrix B is chosen to be full in order to generalize the model to any arbitrary actuator geometry, as on the three-wing. The state output matrix C is chosen to be identity so that the state $\{p, q, r\}$ is the output. Finally, the feed-forward matrix D is chosen to be zero to ensure a strictly-proper system that will avoid complications in $\mathcal{L}_1$ implementation.

To identify the parameters of the above model, data is logged for a series of

routine flight tests. The control deflections $\delta_1, \delta_2, \delta_3$ are numerically differentiated using a simple finite difference technique. The differentiated control data $\{\dot{\delta_1}, \dot{\delta_2}, \dot{\delta_3}\}$, the body rates $\{p, q, r\}$, and the parametrized state space model (5.3) were applied to Matlab's prediction-error minimization (`pem`) command to find estimates of the parameters a_i and b_{ij}. The regression method attempts to minimize the mean-square prediction error. Figure 5-2 shows the measured outputs $\{p, q, r\}$ compared to the predicted outputs from the parameter identification using actual flight test data as input. The plot shows adequate prediction accuracy given the measurement noise and numerical differentiation involved. The estimated parameters are:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \underbrace{\begin{bmatrix} -4.6 & 0 & 0 \\ 0 & -7.9 & 0 \\ 0 & 0 & -6.4 \end{bmatrix}}_{A} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \underbrace{\begin{bmatrix} 5.6 & 7.6 & -60.1 \\ 17.2 & -26.8 & -4.6 \\ 47.7 & 49.5 & 87.7 \end{bmatrix}}_{B} \begin{bmatrix} \dot{\delta_1} \\ \dot{\delta_2} \\ \dot{\delta_3} \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{C} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{D} \begin{bmatrix} \dot{\delta_1} \\ \dot{\delta_2} \\ \dot{\delta_3} \end{bmatrix} \tag{5.4}$$

The above system, which represents the nominal open-loop body-rate behavior of the three-wing tailsitter, is used as the reference model for the MIMO $\mathcal{L}_1$ adaptive controller.

5.4 Adaptive Law

Based on Refs. [14, 15], the following elements comprise the MIMO $\mathcal{L}_1$ controller. The state predictor $\hat{x}$ has the following dynamics:

$$\dot{\hat{x}} = A_m \hat{x} + B_m \dot{\delta}_{bl} + B_m(\hat{\Lambda}\dot{\delta}_{ad} + \hat{\sigma} + \hat{k_x}^{\mathrm{T}} x + \hat{k_u}^{\mathrm{T}} \dot{\delta}_{bl}) \tag{5.5}$$

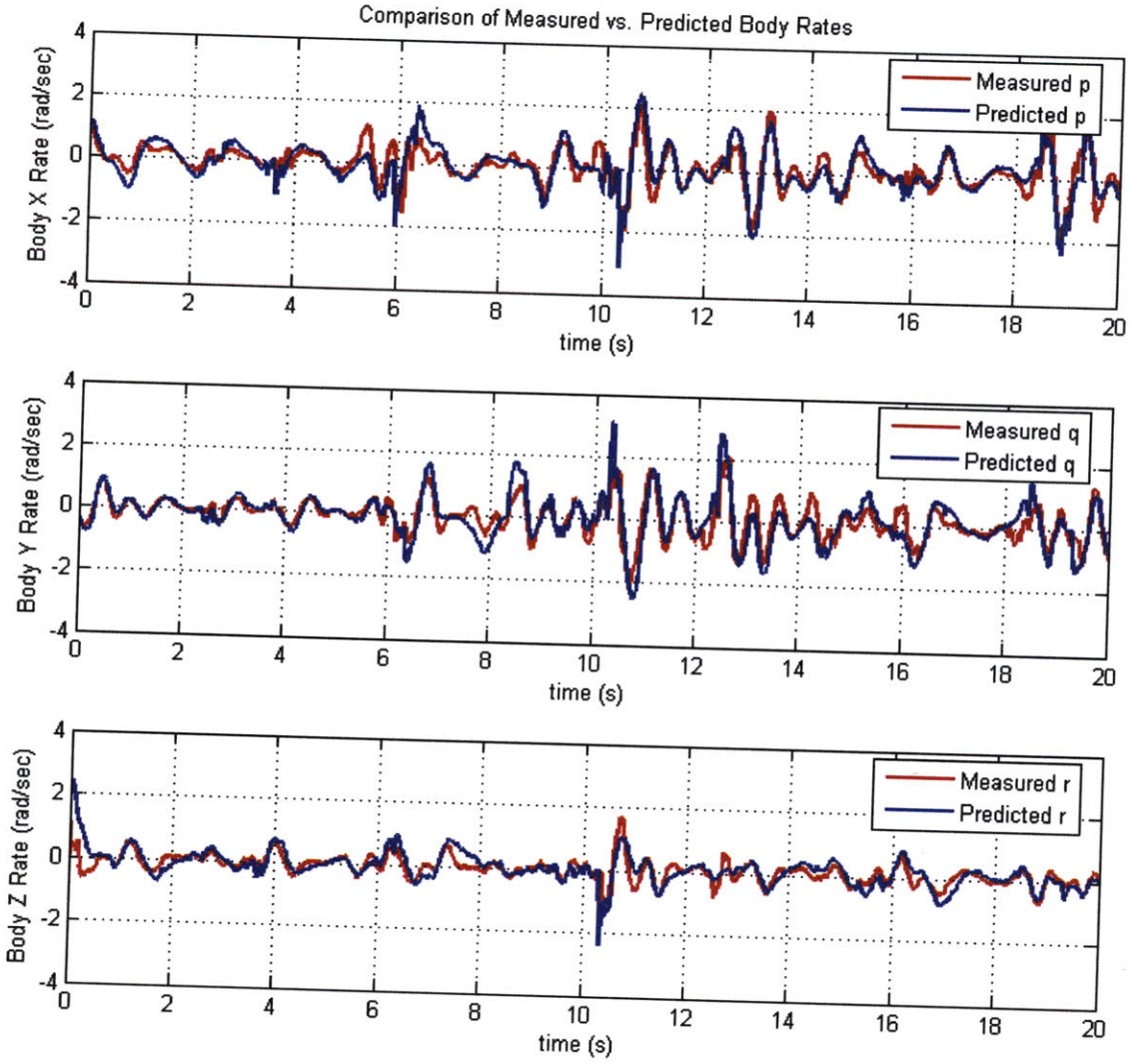

Figure 5-2: Comparison of measured vs. predicted body rates from system identification.

where A_m and B_m are the reference dynamics from (5.4), x is the predictor state $[p, q, r]^{\mathrm{T}}$, $\dot{\delta}_{bl}$ and $\dot{\delta}_{ad}$ are the $\Re^{3\times 1}$ baseline and adaptive control inputs, respectively, to control surface deflection rates, and $\hat{\Lambda}, \hat{\sigma}, \hat{k_x}$, and $\hat{k_u}$ are the adaptive estimates whose dynamics are given by:

$$\dot{\hat{\Lambda}}(t) = -\Gamma \dot{\delta}_{ad}(t)\tilde{x}^{\mathrm{T}}(t)PB_m \tag{5.6a}$$

$$\dot{\hat{\sigma}}(t) = -\Gamma \tilde{x}^{\mathrm{T}}(t)PB_m \tag{5.6b}$$

$$\dot{\hat{k_x}}(t) = -\Gamma x(t)\tilde{x}^{\mathrm{T}}(t)PB_m \tag{5.6c}$$

$$\dot{\hat{k}}_u(t) = -\Gamma \dot{\delta}_{bl}(t) \tilde{x}^{\mathrm{T}}(t) P B_m \tag{5.6d}$$

where $\Gamma > 0$ is the scalar adaptation rate, $\tilde{x}(t) = \hat{x}(t) - x(t)$ is the prediction error, $P = P^{\mathrm{T}} > 0$ is the solution to the algebraic Lyapunov equation, and the projection operator [26] is applied to all derivatives to ensure boundedness. Finally, the adaptive control input is given by:

$$\dot{\delta}_{ad}(s) = -k\chi(s) \tag{5.7a}$$

$$\chi(s) = D(s)\hat{\Lambda}(s)\dot{\delta}_{ad}(s) \tag{5.7b}$$

$$D(s) = \begin{bmatrix} \frac{1}{s} & 0 & 0 \\ 0 & \frac{1}{s} & 0 \\ 0 & 0 & \frac{1}{s} \end{bmatrix} \tag{5.7c}$$

where $k > 0$ sets the control bandwidth since $\dot{\delta}_{ad}$ is in a simple gain-feedback system.

As shown in Figure 5-1, the output $\dot{\delta}_{ad}$ of the adaptive system is integrated to obtain the adaptive control increment δ_{ad}. This gets summed with the baseline control signal δ_{bl} to form the total control signal sent to the control surfaces.

5.5 Simulation Results

To verify the performance of the adaptive augmentation in an actuator failure scenario, simulations are performed using identified system (5.4). The failure matrix from (5.2) is set to:

$$\Lambda = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.4 & 0 \\ 0 & 0 & 1.0 \end{bmatrix} \tag{5.8}$$

which represents extreme failures of 80% and 60% on actuators one and two, respectively. The adaptive parameters chosen are $k = 25$ and $\Gamma = 1000$, and the simulation integration timestep is 0.0005 seconds. Simple sums of sinusoids are given as baseline inputs. While the sinusoids are not a very realistic input to the system, the point is to

prove that the adaptive augmentation does its job in recovering reference performance in the face of actuator failures.

Results of the failure simulation are shown in Figure 5-3. The top three plots show the body rates $\{p, q, r\}$ for the reference system, baseline system, and system with adaptive augmentation. As expected, the baseline system response no longer matches the reference dynamics (due to the failures), and the adaptive augmentation recovers the reference performance for all three body rates. The bottom plot shows the adaptive control increment. Note that the magnitude of the adaptive control inputs relfect the fact that there is a large failure on the first actuator and a slightly smaller failure on the second. The lack of adaptive input on the third actuator indicates that there is no failure on that input. The result confirms that, at least in simulation, the adaptive controller is doing it's job correctly.

5.6 Experimental Results

5.6.1 Implementation on Flight Hardware

To experimentally test the MIMO $\mathcal{L}_1$ scheme described above on the three-wing tailsitter, the algorithm is implemented in the RAVEN testbed (see Section 2.5). Numerical integration on the flight computer is performed at $3000Hz$ and a fourth order Runge-Kutta scheme is used to ensure numerical stability. The $\mathcal{L}_1$ bandwidth parameter is set to $k = 5$, the adaptation rate is $\Gamma = 2500$, and the reference model (5.4) is used.

To test the adaptive controller's ability to recover performance after an actuator failure, roughly 60% of one of the three control surfaces is physically removed as shown in Figure 5-4. Data is obtained comparing the baseline controller with no failure, the baseline with the phsyical failure, and the $\mathcal{L}_1$ controller with the physical failure. To ensure a fair comparison, the vehicle tracking responses are compared when a $\pm 1m/s$ sinusoid is given as the horizontal X-velocity command.

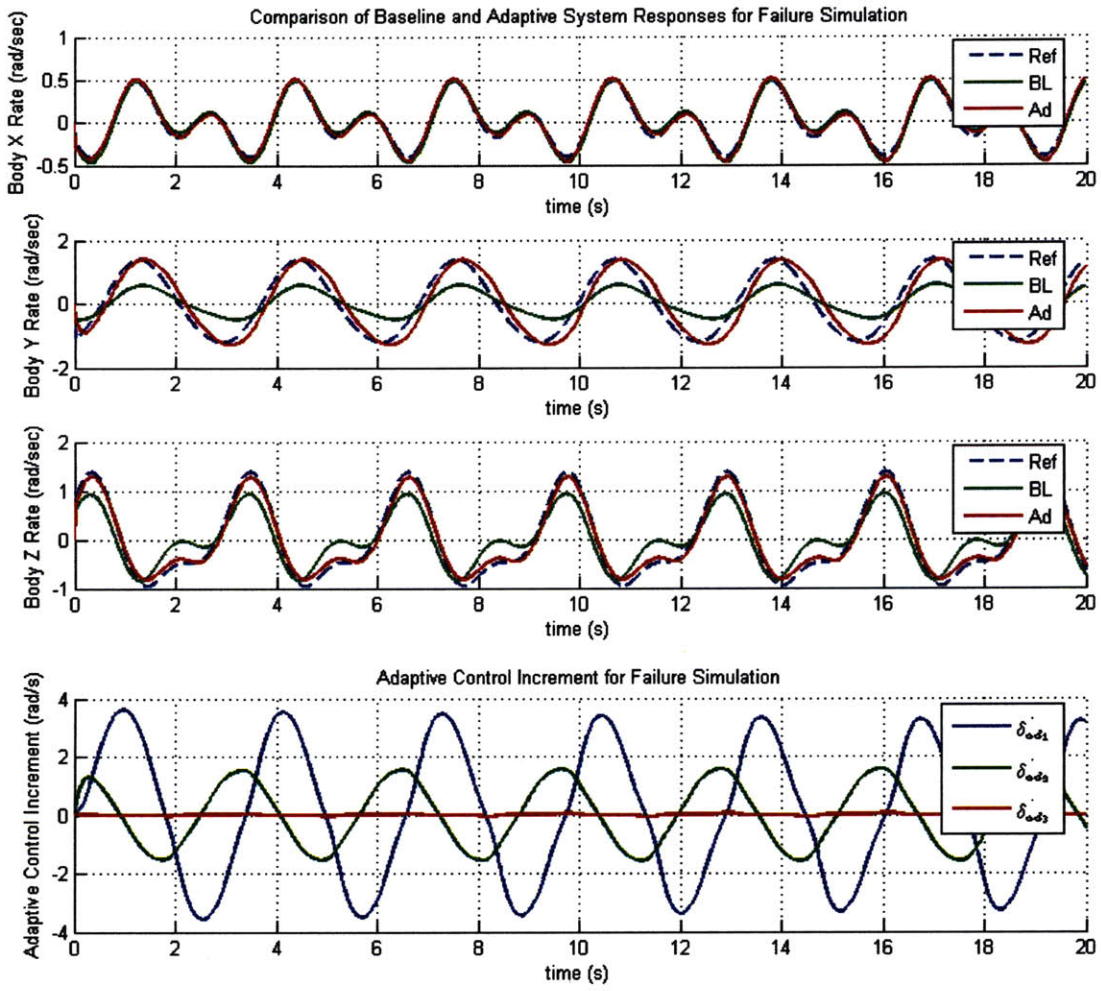

Figure 5-3: Results from failure simulation. Top three plots show the body rates $\{p, q, r\}$ for the reference system, baseline system, and system with adaptive augmentation. Bottom plot shows the adaptive control increments, which correspond to the failures on the first and second actuators.

5.6.2 Modification to Adaptive Laws

Upon implementation and flight testing, the $\mathcal{L}_1$ controller was found to slowly saturate the control surface deflections, leading to vehicle instability and failure. Note in Figure 5-1 that the $\mathcal{L}_1$ adaptive control signal $\dot{\delta}_{ad}$ is integrated to obtain δ_{ad} before being added to the baseline control signal. Thus if the output of the adaptive controller is non-zero the signal δ_{ad} will slowly grow with time, eventually saturating the control surface. This effect, known as "windup", must be addressed for stable

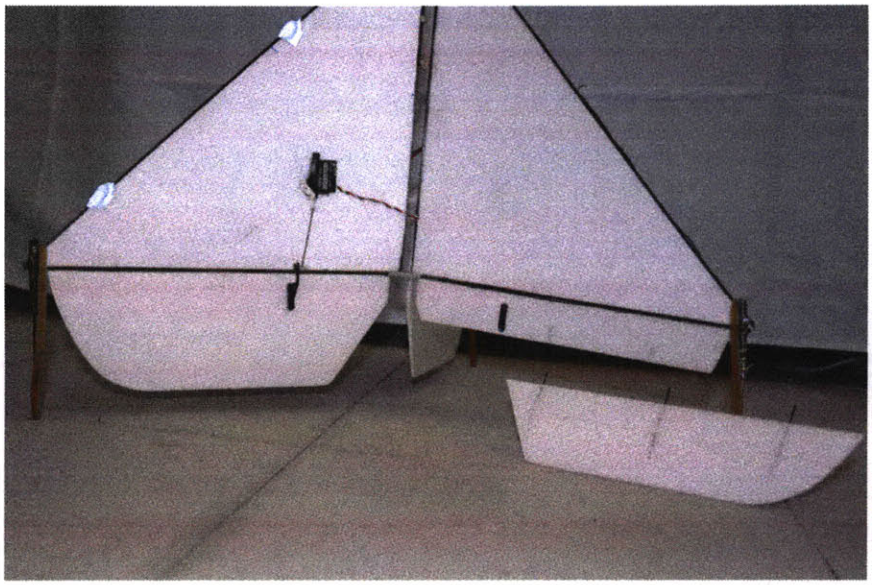

Figure 5-4: To test the ability of the MIMO $\mathcal{L}_1$ controller to recover baseline performance after an actuator failure, roughly 60% of a three-wing control surface is removed.

adaptive augmentation in flight.

To counteract the windup, a few small modifications to the adaptive laws are needed. It should be noted that windup is a common problem in flight control, and many anti-windup schemes have been developed to address it [27]. The method presented here was devised to be simple to implement in hardware and, more importantly, was found to work such that the $\mathcal{L}_1$ controller still had an effect on the system response without saturating the control surfaces. It should be noted that no theoretical proof is given asserting the stability and performance of the resulting adaptive system, only experimental results are shown.

The first modification is the addition of a dissipation term to the integrated adaptive signal δ_{ad}. Instead of simply integrating $\dot{\delta}_{ad}$, the signal is constructed as:

$$\delta_{ad}(s) = \frac{1}{s}(\dot{\delta}_{ad}(s) - \epsilon_1 \delta_{ad}(s)) \tag{5.9}$$

where ϵ is a positive dissipation constant which in practice typically ranged from 0.1 to 0.4. This update law slowly drives the adaptive control augmentation to zero after

some initial transient. To ensure that the adaptive parameters do not windup as a result, similar modifications are made to the adaptive laws:

$$\dot{\hat{\Lambda}}(t) = -\Gamma\dot{\delta}_{ad}(t)\tilde{x}^{\mathrm{T}}(t)PB_m - \epsilon_2\hat{\Lambda} \tag{5.10a}$$

$$\dot{\hat{\sigma}}(t) = -\Gamma\tilde{x}^{\mathrm{T}}(t)PB_m - \epsilon_3\hat{\sigma} \tag{5.10b}$$

$$\dot{\hat{k_x}}(t) = -\Gamma x(t)\tilde{x}^{\mathrm{T}}(t)PB_m - \epsilon_4\hat{k_x} \tag{5.10c}$$

$$\dot{\hat{k_u}}(t) = -\Gamma\dot{\delta}_{bl}(t)\tilde{x}^{\mathrm{T}}(t)PB_m - \epsilon_5\hat{k_u} \tag{5.10d}$$

where the added dissipation terms are in red, and the ϵ_i's are positive dissipation constants. The ϵ_i's are tuned during flight testing so as to eliminate the control saturation while still ensuring that the $\mathcal{L}_1$ augmentation contributes in a transient manner.

5.6.3 Flight Test Results

Figure 5-5 shows the response of the three-wing tailsitter to the sinusoid velocity commands described above. The top plot shows the baseline controller, with no failure, track the velocity command relatively well. The middle plot shows the obvious performance degradation of the baseline controller when 60% of one control surface is physically removed. As seen on the bottom plot, the MIMO $\mathcal{L}_1$ adaptive system with the same 60% failure recovers the nominal baseline performance.

These results experimentally confirm the input failure robustness benefits of $\mathcal{L}_1$ augmentation, and validate the inner-loop adaptation scheme used for the three-wing tailsitter. They also confirm that the dissipation modification, though not theoretically justified, is effective in preventing windup of the control and adaptive signals without apparently negating the effects of the adaptive controller itself.

5.7 Summary

Multi-input multi-output $\mathcal{L}_1$ adaptive control is applied successfully to a three-wing tailsitter both in simulation and flight testing. An inner-loop adaptation scheme is

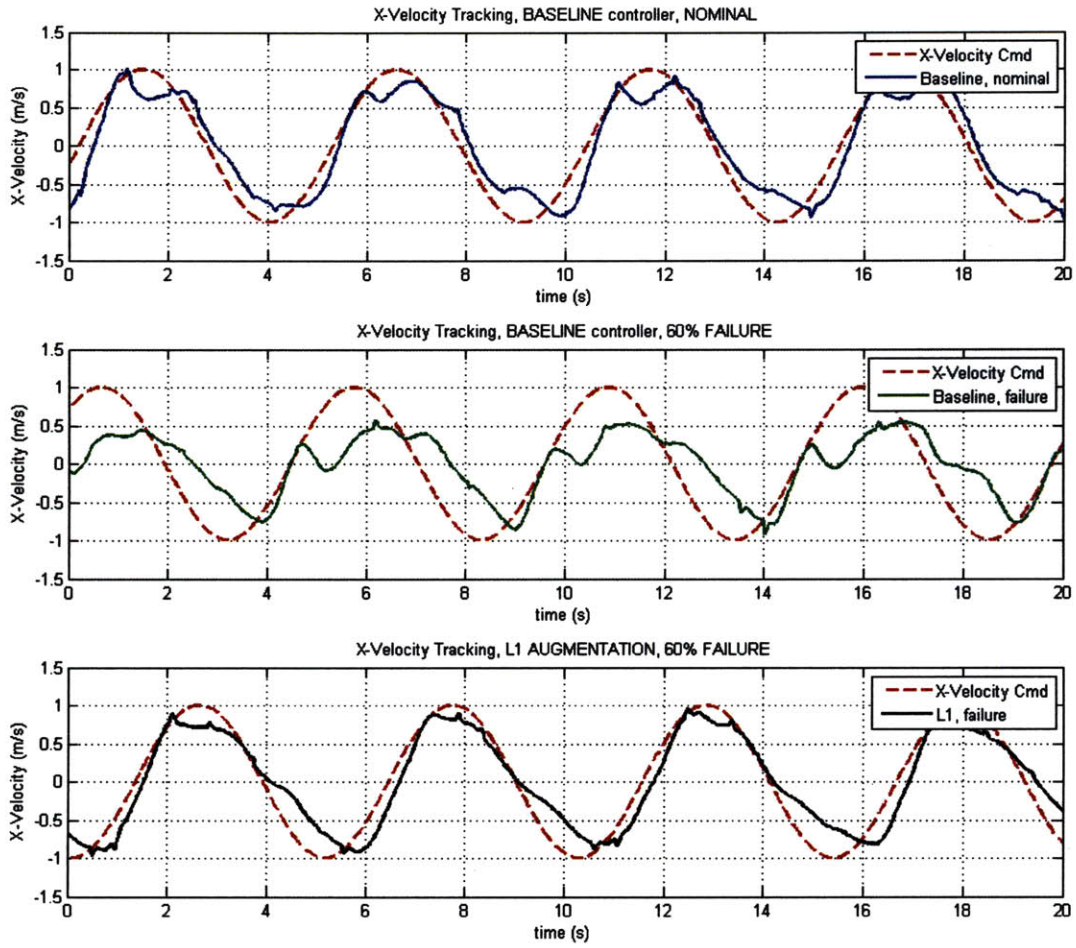

Figure 5-5: X-velocity tracking for the baseline controller with no failures (top,blue), the baseline controller with a 60% failure on one control surface (middle,green), and the augmented $\mathcal{L}_1$ system with the same failure (bottom,black).

used such that the adaptive augmentation is able to operate independently of the non-linear baseline controller. System identification is used to identify a suitable reference model of these inner-loop dynamics, which themselves are complex due to a complex propeller flow field and asymmetric control forces.

The adaptive controller shows greatly improved performance over the baseline system in spite of severe actuator failures both in simulation and flight testing. This is an encouraging result promoting further use of MIMO adaptive control on unmanned vehicles, especially on indoor autonomous vehicles that face many difficult control problems.

One area of future work is the application of the same inner-loop body-rate adaptation scheme to other indoor flight vehicles. Since the scheme is general and invariant to the type of closed-loop controller used, in theory it should be applicable to any vehicle. More important, though, is the theoretical justification of the dissipation modification to the adaptive laws. In cases like these where the adaptive control signal must be integrated in time, windup and control saturation is a realistic consequence. Further research should be directed to the study of anti-windup in these types of adaptive systems.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 6

Low-Fidelity Modeling

6.1 Introduction

Chapters 4 and 5 relied on system identification for choosing an appropriate plant model to be used in designing the adaptive controllers. In the case of the quadrotor in Chapter 4, identification was used since only a model of the closed-loop dynamics was needed. As for the three-wing tailsitter in Chapter 5, complexities of the propeller flow and control forces made first principles modeling difficult and inaccurate. In this chapter an attempt is made to model the Clik fixed-wing aerobatic aircraft (from Section 2.4) using basic propeller modeling techniques and flat-plate theory. While assumptions are made which decrease the fidelity of the model, the objective is to capture the dominant dynamic behavior without resorting to more complex computational fluid dynamics methods. Low complexity is an important requirement since the model will be used for trajectory design and optimization in Chapter refchap:traj.

The chapter proceeds as follows. Section 6.2 presents a blade-element model for predicting thrust and induced flow velocities from the propeller, Section 6.3 outlines a method for modeling the development of the slipstream axially towards the control surfaces, and Section 6.4 describes the method by which control surface forces are predicted. Model predictions are compared to measured data from the actual aircraft throughout to verify the accuracy of the method.

6.2 Modeling the Flow Field Behind the Propeller

Following from [16], the first step in modeling the dynamics of the aircraft as a whole is to model the expected flow generated by the propeller. At low forward speeds this will be the dominant flow affected by the control surfaces. The modeling strategy is as follows:

1. Identify a simple blade-element modeling technique
2. Implement the model in Matlab
3. Input propeller geometry and compare with known thrust, power, and RPM measurements to validate the model

6.2.1 Blade-Element Model

A simple blade-element model is presented by Phillips [28] and later used specifically for flow field calculations by Hunsaker [17]. Details of the method can be found in the references, but the calculation at each blade element is derived by equating the lift from Goldstein's vortex theory [29] to the circulation about the element. The resulting equation that must be solved numerically for ϵ_i at each radial station is:

$$\frac{kc_b}{16r}C_L - \cos^{-1}\left\{\exp\left[-\frac{k(1-2r/d_p)}{2\sin(\beta_t)}\right]\right\}\tan(\epsilon_i)\sin(\epsilon_\infty + \epsilon_i) = 0 \tag{6.1}$$

where

k = number of blades

c_b = local chord length (m)

r = radius at this station (m)

C_L = 2D lift coefficient, function of angle of attack

C_D = 2D drag coefficient, function of angle of attack

d_p = propeller diameter (m)

β_t = geometric angle of attack at propeller tip (rad)

ϵ_i = blade section induced angle of attack (rad)

ϵ_∞ = blade section advance angle of attack (rad)

ω = Propeller angular velocity (rad/s)

From here thrust can be calculated as:

$$\begin{aligned} T &= \int_R^{r=r_h} \frac{dT}{dr} dr \\ &= \frac{k\rho\omega^2}{2} \int_R^{r=r_h} r^2 c_b \frac{\cos^2(\epsilon_i)}{\cos^2(\epsilon_\infty)} \left[C_L \cos(\epsilon_\infty + \epsilon_i) - C_D \sin(\epsilon_\infty + \epsilon_i)\right] dr \end{aligned} \tag{6.2}$$

and similarly torque can be calculated as:

$$\begin{aligned} l &= \int_R^{r=r_h} \frac{dl}{dr} dr \\ &= \frac{k\rho\omega^2}{2} \int_R^{r=r_h} r^3 c_b \frac{\cos^2(\epsilon_i)}{\cos^2(\epsilon_\infty)} \left[C_D \cos(\epsilon_\infty + \epsilon_i) + C_L \sin(\epsilon_\infty + \epsilon_i)\right] dr \end{aligned} \tag{6.3}$$

The velocity vector directly behind the propeller at each radial station has an axial component V_{xi} and a tangential component $V_{\theta i}$, which are given by:

$$V_{xi} = \frac{\omega r}{\cos(\epsilon_\infty)} \sin(\epsilon_i) \cos(\epsilon_i + \epsilon_\infty) \tag{6.4}$$

$$V_{\theta i} = \frac{\omega r}{\cos(\epsilon_\infty)} \sin(\epsilon_i) \sin(\epsilon_i + \epsilon_\infty) \tag{6.5}$$

Using this model the thrust, power (the product of torque and angular velocity), RPM, and flow velocities directly behind the propeller can be calculated. However, using the model requires knowledge of the lift and drag coefficients C_L and C_D, which depend on the angle of attack and Reynolds number.

6.2.2 C_L and C_D

Once an airfoil is chosen, C_L and C_D must be estimated as a function of angle of attack and Reynolds number. XFOIL [30] can be used to estimate pre-stall lift and drag coefficients at various Reynolds numbers and angles of attack. However, typical propeller airfoils stall at angles of attack between 15 and 20 degrees while angles of

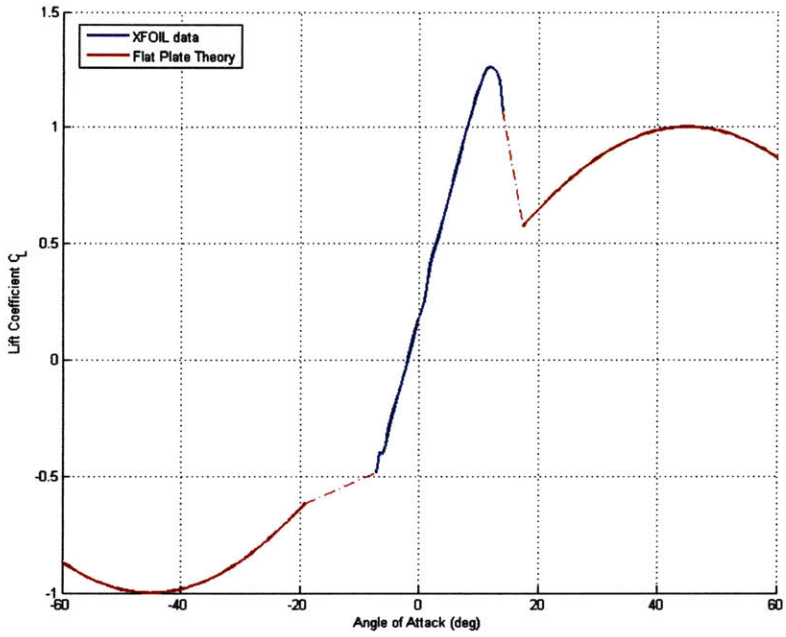

Figure 6-1: Combining XFOIL and flat-plate theory for SC1094 airfoil, Re=1e5.

attack can range all the way up to 70 degrees in the blade-element calculations. Thus, XFOIL data must be appended with flat-plate theory predictions for post-stall angles of attack [31]. Flat-plate theory predicts the following lift and drag coefficients:

$$C_L = 2\sin(\alpha)\cos(\alpha) \tag{6.6a}$$

$$C_D = 2\sin^2(\alpha) \tag{6.6b}$$

where α is the aerodynamic angle of attack. An example of combined CL data for the SC1094 airfoil (common on helicopter rotors) is shown in Figure 6-1. After the data has been appended for various Reynolds numbers, 2-D tables of CL and C_D vs. angle of attack and Reynolds number are constructed and used in the blade-element calculations.

6.2.3 Matlab Implementation

The blade-element model presented above is implemented as a Matlab script. The user sets the following design parameters:

- Propeller diameter, number of blades, and RPM
- Blade chord, twist and airfoil as a function of radius
- Number of radial stations for blade element calculations
- Freestream velocity (aligned with propeller axis)

The code calculates the following outputs:

- Thrust, torque and power
- Axial and tangential velocities directly behind the propeller
- 3D propeller visualization with calculated flow fields

With 50 radial stations, the code executes in less than two seconds. The code thus allows the user to quickly and easily modify propeller geometries and obtain the resulting approximate flow-field.

6.2.4 Comparison to Measured Data

In an effort to validate the thrust prediction from the blade element model, the Clik propeller geometry parameters were input to the code. The propeller is a Graupner SlowFlyer with 8" diameter and 4.5" pitch, and the SC1094 airfoil (with XFOIL/flate-plate C_L/C_D) is used to approximate the airfoil of the propeller.

Figure 6-2 shows predicted propeller thrust and measured propeller thrust versus propeller RPM. Measured data is taken from [20] in which a high-precision 6-axis load cell was used to measured forces and moments. The plot confirms that, while not exact, the blade element model accurately predicts the trend of thrust with increasing RPM. The maximum prediction error is $0.2N$, or 9%. Given that the airfoil on the Graupner propeller is not the SC1094, assumptions in the model, and possible measurement inaccuracies, these errors were deemed acceptable.

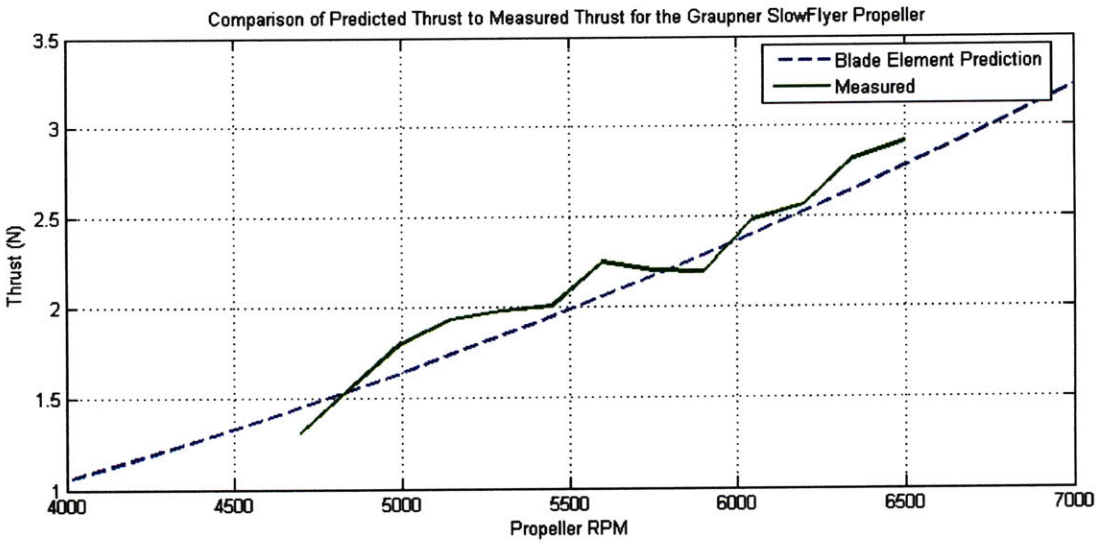

Figure 6-2: Comparison of predicted thrust (dashed) to measured thrust (solid) for the Graupner SlowFlyer propeller used on the fixed-wing aerobatic aircraft.

6.3 Flow Field Propagation

The flow velocities calculated just behind the propeller must be propagated axially towards the control surfaces. A slipstream contraction and development model used is from Stone [16] and is summarized below. Matlab implementation of the development model coupled with the propeller model above yields a three-dimensional flow field which can be used for control analysis.

6.3.1 Slipstream Development and Contraction Model

As presented in [16], a slipstream development factor k_d is defined:

$$k_d = 1 + \frac{s}{\sqrt{s^2 + R^2}} \tag{6.7}$$

where:

$$\begin{aligned} s &= \text{distance aft of propeller disk} \\ R &= \text{propeller radius} \\ k_d &= \text{slipstream development factor} \end{aligned} \tag{6.8}$$

For a given distance aft of the propeller, s, the slipstream radius and induced velocities are given by:

$$r_{s_m} = \begin{cases} r_{nacelle} & m = 1 \\ \sqrt{r_{s_{m-1}}^2 + (r_m^2 - r_{m-1}^2)\frac{2V_a + v_{ia_m} + v_{ia_{m-1}}}{2V_a + k_d(v_{ia_m} + v_{ia_{m-1}})}} & m = 2, \ldots, n \end{cases} \quad (6.9)$$

$$v'_{ia_m} = k_d v_{ia_m} \quad (6.10)$$

$$v'_{it_m} = 2v_{it_m}\left(\frac{r_m}{r_{s_m}}\right) \quad (6.11)$$

where:

$$\begin{aligned} s &= \text{axial position downstream of the propeller disk} \\ r_{s_m} &= \text{m'th radial position in the slipstream at distance } s \\ r_m &= \text{m'th radial blade station coordinate } (s = 0) \\ r_{nacelle} &= \text{nacelle radius at distance } s \\ V_a &= \text{axial free-stream velocity at rotor disc} \\ v_{ia_m} &= \text{axial induced velocity at m'th radial station } (s = 0) \\ v_{it_m} &= \text{tangential induced velocity at m'th radial station } (s = 0) \\ v'_{ia_m} &= \text{axial induced velocity at m'th radial station at distance } s \\ v'_{it_m} &= \text{tangential induced velocity at m'th radial station at distance } s \end{aligned} \quad (6.12)$$

6.3.2 Realistically Modeling Flow Field Dissipation

According to the above propagation model (6.7)-(6.11), it is assumed that the flow field does not dissipate with distance from the propeller plane. This is an unrealistic assumption especially given that the actuators are a substantial distance downstream of the propeller (see Figure 2-1).

To get an idea of the first-order trends in dissipation, two RC-scale propellers were tested using a hand-held anemometer. The propellers are a Graupner SlowFlyer (same as used above) and a single rotor from the Draganflyer Quadrotor. The anemometer used is a Circuit Specialists MS6250, listed as accurate between $0.4 - 30m/s$ to within

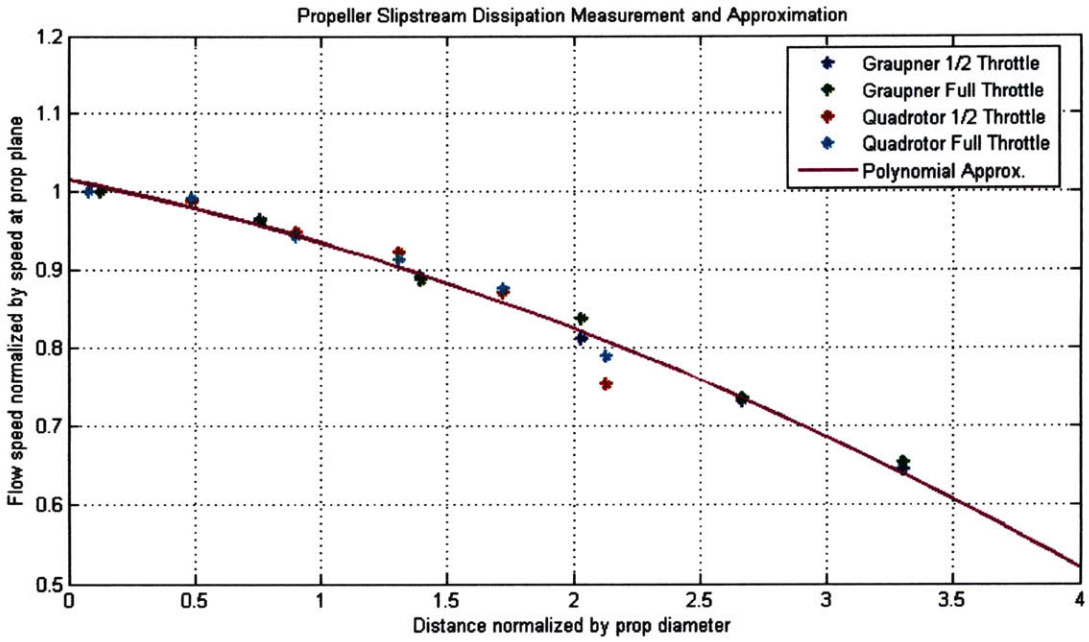

Figure 6-3: Propeller slipstream dissipation measurement results and polynomial approximation.

$\pm 2.0\%$. Velocity data is taken at several distances behind the propeller and tested at several different propeller throttle settings. The distances are then normalized by the propeller diameter, and the velocities normalized by the velocity just behind the propeller. This effectively yields a normalized dissipation coefficient that can be applied to any propeller slipstream. As shown in Figure 6-3, the normalized data is fairly consistent between the two propellers and throttle settings. A second order polynomial function is fit to the data and is also shown.

6.3.3 Matlab Implementation

Given the flow field directly behind the propeller as calculated numerically per Section 6.2.3, the contraction and development model above is applied to yield a 3D map of both the slipstream radius and the induced flow field. The code takes an (x, y, z) position as input and calculates the flow field velocity,(V_x, V_y, V_z), at that location (which includes both the freestream flow and the induced propeller flow).

Using this function, any arbitrary grid of 3D flow velocities can be created and

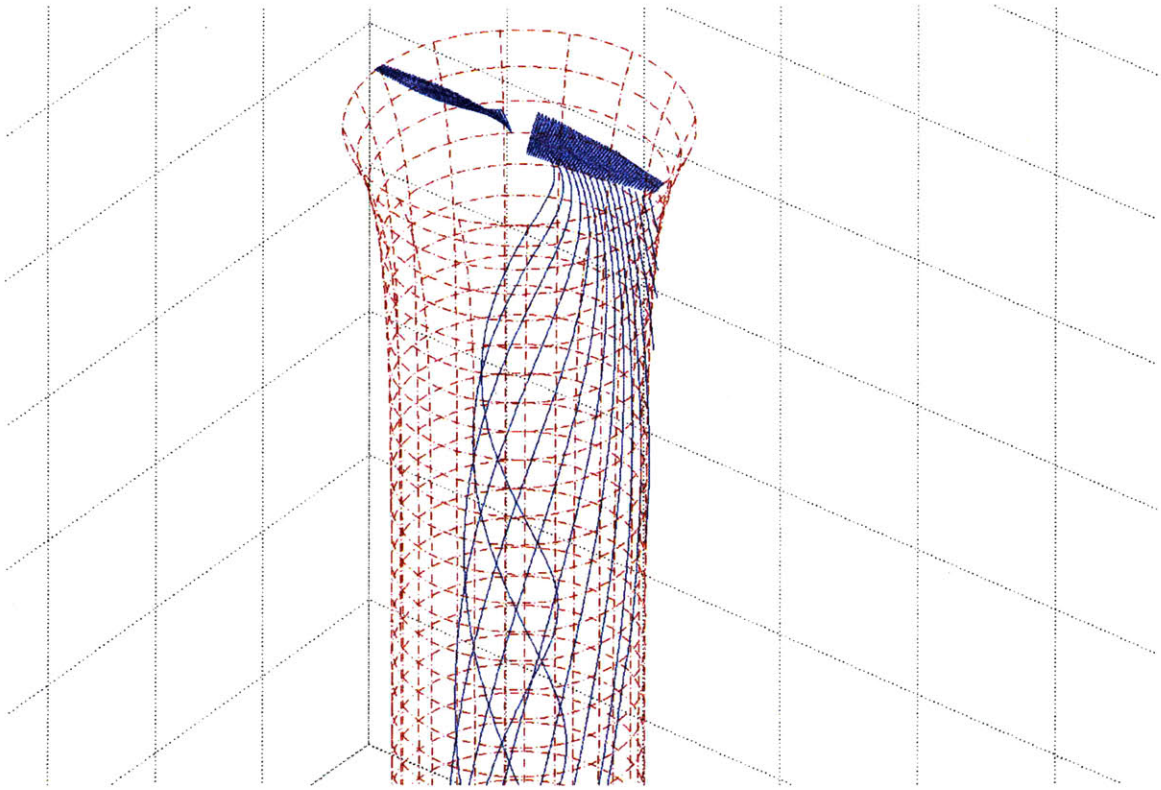

Figure 6-4: Slipstream development (red) and streamlines starting at the propeller (blue) for the Graupner propeller.

passed to other helpful functions such as Matlab's `streamlines`. A 3D rendering of the slipstream (red) and some streamlines starting at the propeller (blue) are shown in Figure 6-4 for the Graupner propeller.

6.4 Calculating Control Surface Forces

Once the three-dimensional flow field is known, control surface forces can be calculated. The general strategy is as follows:

1. Discretize the surface spanwise into n 2D stations, specify the leading edge position and chord length of each.
2. For each node, calculate the 3D flow velocity at the node's leading edge and calculate the 2D aerodynamic angle of attack. Use this to calculate the node's 2D lift and drag forces as a function of actuator deflection angle (which is just added or subtracted from the aerodynamic angle of attack).

3. Resolve the lift and drag forces into a 3D point force in the global coordinate system.

Assumptions:

1. While actuators can have any chord distribution along their span, they are assumed to be on a hinge parallel to the plane of the propeller (i.e. perpendicular to the axial flow).
2. For this first-order analysis, actuators are assumed to be 2D flat plates (of possibly varying chord as stated above). Aerodynamic forces are assumed to act at the quarter-chord of each spanwise node and pitching moments are assumed to be negligible (i.e. $C_M = 0$).
3. Spanwise flow along the control surfaces is assumed to be negligible.

6.4.1 Aircraft Control Surfaces

Measurements of the Clik fixed-wing aircraft geometry are input into the control force prediction code. Figure 6-5 shows the control surfaces as well as the propeller geometry and slipstream development. The plot is helpful in predicting what portion of the control surfaces will be inside of the propeller slipstream. Note that static surfaces such as the wings and body are not shown. The forces and moments from these surfaces are calculated in exactly the same manner as the control surfaces with the exception that the deflection angle is set to zero.

6.4.2 Comparison to Measured Data

Again using data measured in [20], predicted control moments for the Clik are compared to measured values. Figure 6-6 shows the moment comparison for the elevator (top), rudder (middle), and aileron (bottom). Note that all moments are taken about the center of mass. From the elevator and rudder plots it is clear that the model consistently over-estimates the moment, which is likely a consequence of the flat-plate C_L used. However, both the trend and the maximum moments appear to be correctly

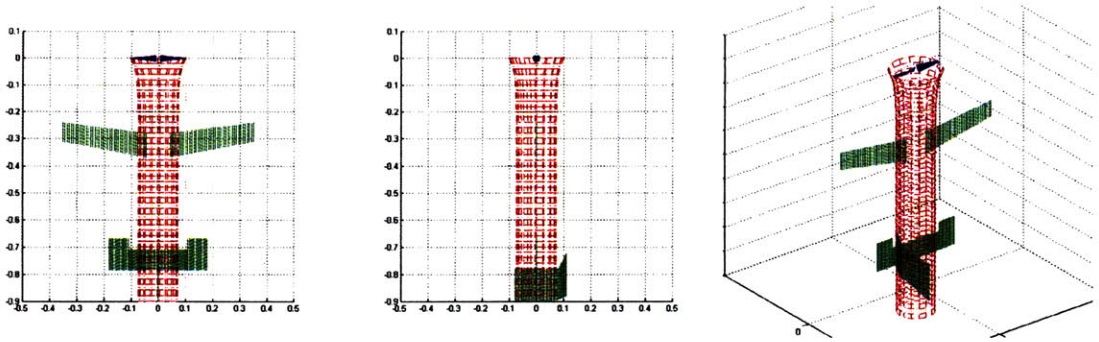

Figure 6-5: Fixed-wing aerobatic aircraft control surfaces (green), propeller geometry (blue), and slipstream development (red). Top view (left), side view (middle), and isometric view (right).

predicted. The aileron plots shows close agreement for negative deflection angles but over-prediction for positive deflections. This could be due to the precision of the load cell given the extremely small moments being measured ($\sim 0.01 Nm$). Again, though, the model predicts the maximum moments fairly accurately. The model also appears to correctly predict the asymmetry of the aileron moment due to the swirl of the slipstream induced by the propeller.

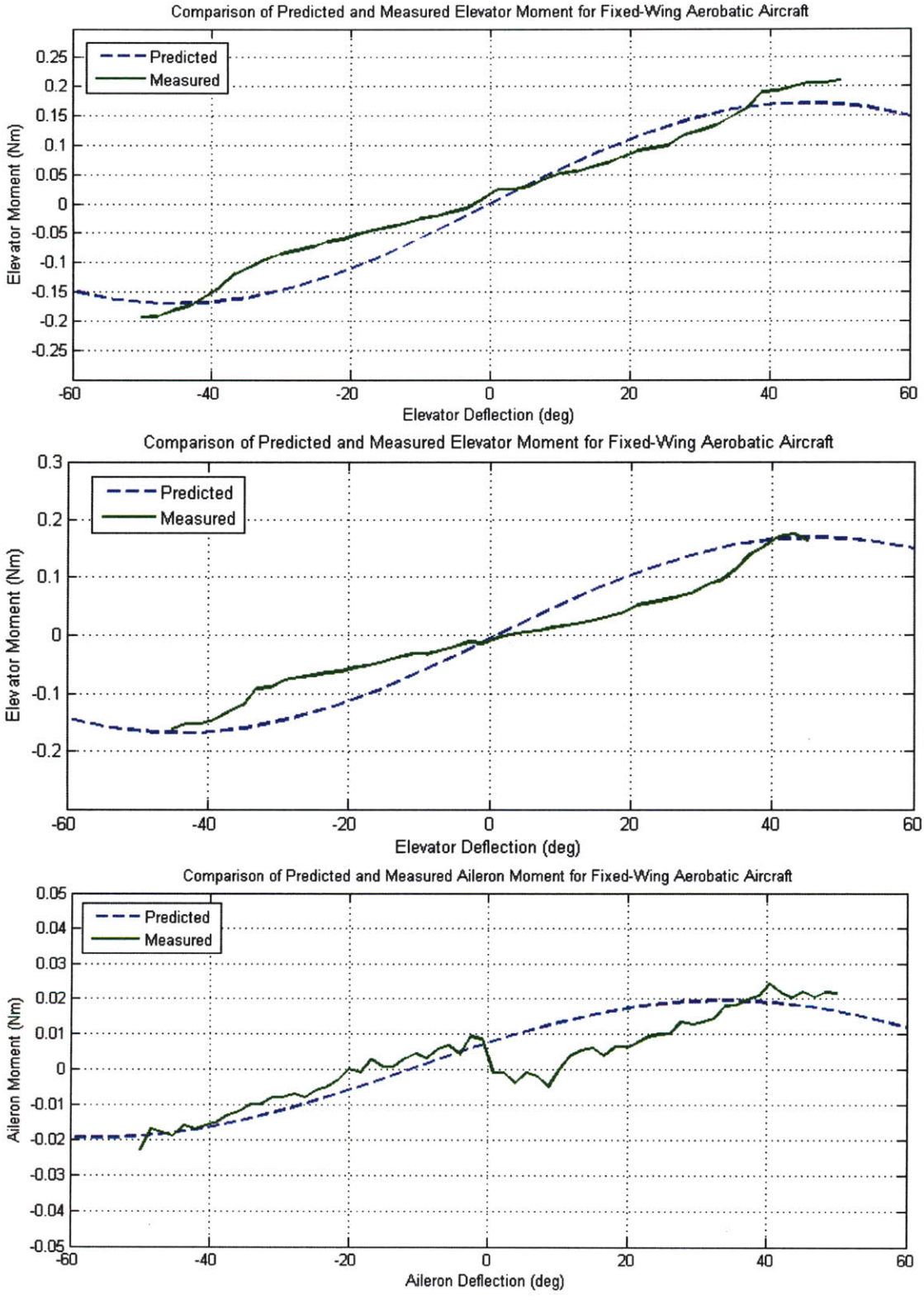

Figure 6-6: Comparison of predicted moments (dashed) to measured moments (solid) for the elevator (top), rudder (middle), and ailerons (bottom) of the Clik fixed-wing aerobatic aircraft.

Chapter 7

Trajectory Design and Optimization

7.1 Introduction

Many unmanned aircraft are capable of operation through several distinct flight regimes. For instance, the Clik from Section 2.4 can start in a "prop-hang" hover, transition to high-speed conventional flight, then resume hover. The ability to perform such a maneuver gives UAVs the potential to perform many types of missions that manned aircraft simply cannot match. This problem has been explored and tested experimentally on the Clik in [20], and this section serves to extend the previous work.

One difficulty, however, is the design of vehicle trajectories for these maneuvers. Given a vehicle's dynamics and actuator capabilities, even if relatively straightforward like those of the Clik aircraft, it is not obvious how to calculate control commands to achieve some arbitrary desired trajectory. Many techniques, such as backstepping [20], hybrid control [18] and rapidly-exploring random trees [32], have been developed and tested on unmanned aircraft. This chapter presents a method that utilizes an inner-loop attitude controller, a low-fidelity dynamic model, and a Gauss pseudospectral optimization technique to find, optimize, and implement dynamically feasible vehicle trajectories. There are several reasons for using this approach over

the others. Foremost, the method allows for intuitive specification of desired vehicle trajectories through specification of a cost function and a desired final state. As will be seen, the flexibility of the optimization software allows for the inclusion of obstacle avoidance and multi-phase trajectories. Finally, closed-loop implementation of the approach adds a substantial amount of robustness to model uncertainty, decreasing the amount of time and effort needed to find an accurate dynamic model.

The low-fidelity dynamic model developed for the Clik in Chapter 6 is used in conjunction with a Gauss pseudo-spectral optimization technique to identify dynamically feasible trajectories and optimize them using a user-specified cost function. The output of the optimization is a time history of vehicle attitudes, which is then sent to the inner-loop attitude controller from Chapter 3. The attitudes are tracked by the closed-loop controller during flight and the vehicle thus performs the desired maneuver. The examples are currently limited to two-dimensional maneuvers, but can be generalized to 3D with future work.

The chapter is structured as follows. Section 7.2 give a brief overview of the GPOPS optimization software, Section 7.3 describes the specification of the Clik vehicle dynamics in two dimensions, Section 7.4 presents the method by which desired trajectories are specified, Section 7.5 presents experimental flight test results, and Section 7.6 summarizes the work.

7.2 GPOPS

Gauss Pseudospectral Optimal Control Software (GPOPS) [33] is a new, free, and open-source Matlab tool for solving multiple-phase optimal control problems. GPOPS implements the Gauss Pseudospectral Method and has an easy-to-use structure-based interface. The software requires specifications of the system dynamics, initial conditions, state and time constraints, and cost function. It then attempts to minimize the cost function given the constraints, and if successful returns a time history of the states and minimizing control inputs.

For the trajectory design problem addressed here, GPOPS is used to find and op-

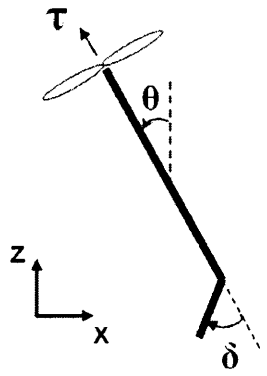

Figure 7-1: Diagram representing the 2D model of the Clik aerobatic aircraft used for trajectory optimization. In the diagram θ represents pitch angle, δ elevator deflection, and τ thrust.

timize a feasible vehicle trajectory. The following sections describe the specifications given to the software to achieve this task.

7.3 Specifying Vehicle Dynamics

GPOPS requires dynamic laws by which the system is governed. The software passes a state vector to a user-defined function that must return the derivatives of each state. While the model derived in Chapter 6 is designed to predict 3D forces and moments, for the purposes of demonstrating this trajectory design method the model is reduced to 2D forces and moments where the pitch axis of rotation is considered. Figure 7-1 shows the definitions of coordinates x and z, pitch angle θ, elevator deflection δ, and propeller thrust τ. The model takes in the specified state vector $X = [x, \dot{x}, z, \dot{z}, \theta, \dot{\theta}]^{\mathrm{T}}$ and GPOPS control inputs $u = [\delta, \tau]$ to calculate the linear forces F_x and F_z and the pitch moment M_θ. This routine is called frequently by GPOPS whenever it requires state derivatives, but it should be noted that the model can just as easily be used to pre-compute tables of these outputs to speed up execution. Physical parameters such as mass and moment of inertia of the aircraft are taken from [20], where they were measured empirically.

7.3.1 Imposing Actuator Limitations

GPOPS can be configured to set bounds on the control inputs δ and τ. In the case of the Clik aircraft, these limits are $\delta \in [-60°, 60°]$ in elevator deflection and $\tau \in [0N, 2.9N]$ in thrust. While these represent the measured physical limits of the aircraft itself, they do not account for the dynamic limits of the actuators. For instance, GPOPS could legally command an elevator deflection of $-60°$ at one time step and $+60°$ at the next time step though the elevator servo can not possibly match this. To enforce rate-limiting on the control inputs, the two control states δ and τ are added to the state vector. The GPOPS control inputs are now the time derivatives of the controls, $\dot{\delta}$ and $\dot{\tau}$. The input bounds are set to coincide with the physical actuator rate limits (based on published specifications), and bounds are set on the control states δ and τ coinciding to the deflection and thrust limits listed above. This guarantees that the control inputs to the system are achievable by the flight hardware thus ensuring a dynamically feasible trajectory.

7.3.2 State Constraints

GPOPS also accepts hard state constraints set by the user. Since experimental tests are done in the RAVEN indoor testbed, constraints on x and z are set to match the physical volume of the test area. Also, constraints are set on $\dot{\theta}$ to prevent extremely fast pitching motions.

7.4 Specifying a Desired Trajectory

Since GPOPS is a generalized optimization routine, there are many ways to specify a desired vehicle trajectory. One method is simply to fix the desired final state of the vehicle. In practice, however, this hard constraint leads to erratic behavior of the solution and does not give the user control over the actual path taken. Instead, trajectory shaping is effected through the cost function.

7.4.1 Cost Function for Desired Trajectories

A more robust method to specify vehicle trajectories is through penalization in the cost function. The general form of the cost functional is:

$$J = M(X(t_f)) + \int_0^{t_f} L(X, u, t)dt \tag{7.1}$$

where t_f is the final time. To specify arbitrary desired position trajectories $x_d(t)$ and $z_d(t)$, the following Lagrange cost $L(X, u, t)$ is used:

$$L(X, u, t) = c_x[x(t) - x_d(t)]^2 + c_z[z(t) - z_d(t)]^2 \tag{7.2}$$

where c_x and c_z are positive weighting terms. Thus the minimizing trajectory is $x(t) = x_d(t), z(t) = z_d(t), \forall t$, which is the desired result. To minimize unnecessary control effort, a penalty is added:

$$L(X, u, t) = c_x[x(t) - x_d(t)]^2 + c_z[z(t) - z_d(t)]^2 + c_\delta \delta(t)^2 + c_\tau \tau(t)^2 \tag{7.3}$$

Additional terms can be added to the terminal cost $M(X(t_f))$ to ensure that the vehicle reaches the final position:

$$M(X(t_f)) = c_{x_f}[x(t_f) - x_d(t_f)]^2 + c_{z_f}[z(t_f) - z_d(t_f)]^2 \tag{7.4}$$

To specify that the vehicle not be moving at the end of the trajectory, penalties can be added to the final velocities:

$$M(X(t_f)) = c_{x_f}[x(t_f) - x_d(t_f)]^2 + c_{z_f}[z(t_f) - z_d(t_f)]^2 + c_{\dot{x}_f}\dot{x}(t_f)^2 + c_{\dot{z}_f}\dot{z}(t_f)^2 \tag{7.5}$$

Using this cost function, the importance of each of the aspects above can be weighted by adjusting the c's. GPOPS then produces a dynamically feasible solution, composed of a time history of states and control inputs, the minimizes the cost function.

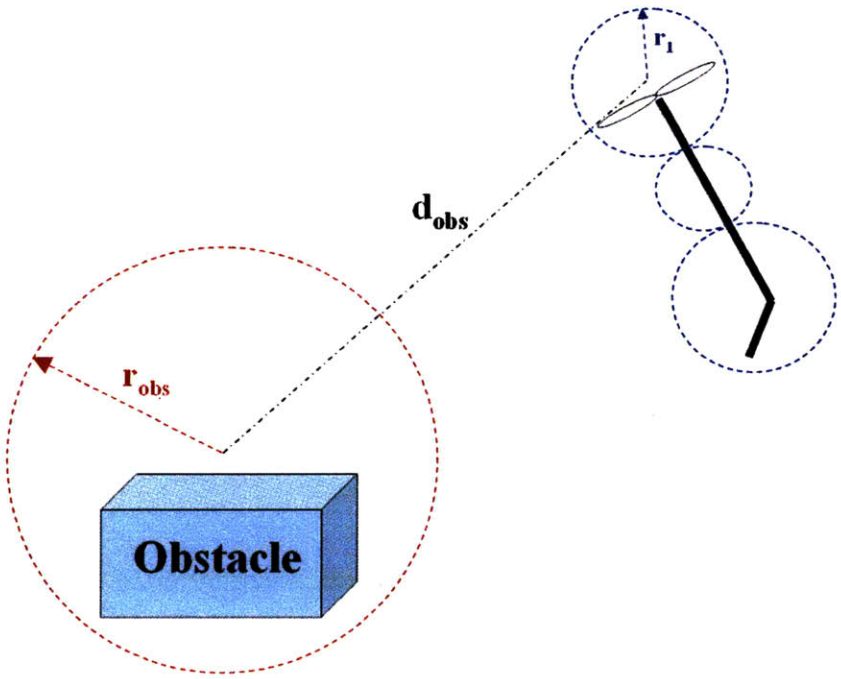

Figure 7-2: Simple obstacle avoidance strategy for trajectory optimization. GPOPS ensures that $d_{obs} > r_1 + r_{obs}$.

7.4.2 Obstacle Avoidance

To demonstrate the inclusion of a obstacle avoidance to the trajectory optimization, a simple method is presented. As shown in Figure 7-2, the flight vehicle is conservatively enclosed by a set of attached circles, and the same is done for the obstacle. A hard constraint is then set in GPOPS such that $d_{obs} > r_1 + r_{obs}$ for each circle surrounding the vehicle, ensuring that the obstacle is physically avoided throughout the trajectory. The radii of the enclosing circles can be adjusted to make the method more or less conservative.

7.5 Experimental Results

7.5.1 Closed-Loop Implementation on Flight Hardware

GPOPS is used to find and optimize dynamically feasible trajectories offline, and the state and control histories are obtained. One method to implement the trajectories on the Clik flight hardware would be to send the elevator and thrust control histories

directly to the actuators in an open-loop fashion. The obvious problem with this and any open-loop control method is that any uncertainty in the dynamic model, hardware variations, and external disturbances will induce errors in the observed trajectory. Given that the model from Chapter 6 used in the optimization is designed more for low-complexity than extreme accuracy, the open-loop errors would likely grow very large.ed trajectory. Given that the model from Chapter 6 used in the optimization is designed more for low-complexity than extreme accuracy, the open-loop errors would likely grow very large.

Instead, closed-loop control is used to track the pitch angle and forward speed obtained from the trajectory state history. The attitude controller from Chapter 3 is used to track the desired pitch angle, and a proportional+integral controller on the throttle is used to track the plane's forward velocity (i.e. the component of velocity in the propeller axis). As is expected when feedback control is applied, this closed-loop strategy adds a great deal of robustness to model uncertainty and is a more realistic implementation technique.

7.5.2 Flight Test Results

Three cases were chosen to test the GPOPS trajectory generation process and implementation on the Clik aerobatic aircraft. In the first case, $x(t)$ and $z(t)$ are specified in the cost function to be a linear path that covers a horizontal distance of 6 meters at roughly $3m/s$ while holding a constant altitude. Figure 7-3 compares the desired trajectory, the GPOPS-generated trajectory, and the measured flight test trajectory. Note that the GPOPS and desired trajectories are not the same, since GPOPS produces a *dynamically feasible* solution whereas the desired trajectory is arbitrarily set by the designer. Close matching can be seen between GPOPS and flight test x-position. However, the flight test $z-$position (altitude) is consistently higher than the GPOPS trajectory. The inner-loop controllers track the commanded pitch and forward velocity closely, implying that the flight vehicle is achieving the attitudes and velocities commanded by GPOPS.

In the second case, $x(t)$ and $z(t)$ are specified as a climb-and-descend trajectory

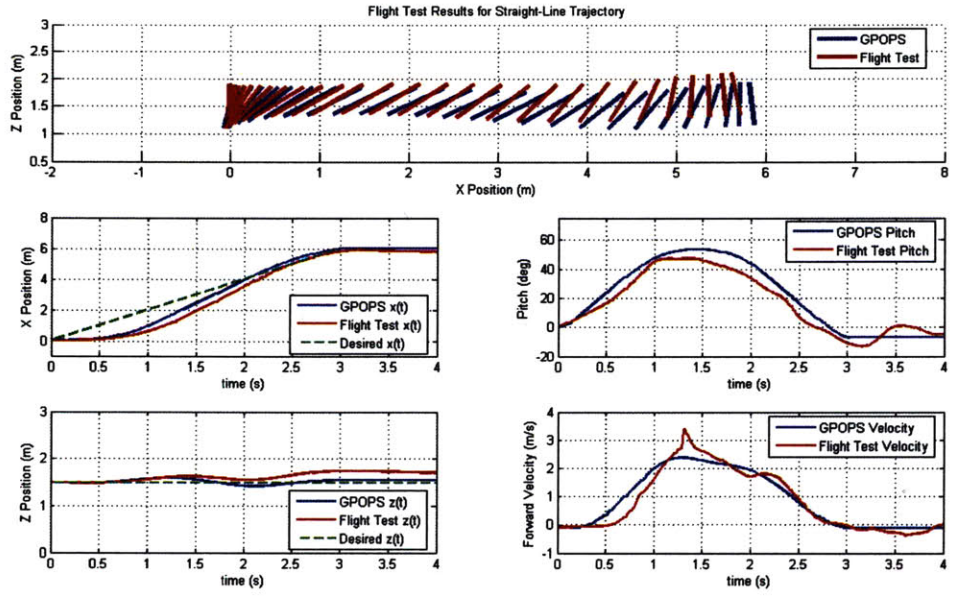

Figure 7-3: Flight test results for straight-line trajectory. Desired trajectory (green), GPOPS solution (blue) and measured flight test data (red) are shown.

again covering 6 meters at roughly $3m/s$. Figure 7-4 compares the desired trajectory, the GPOPS-generated trajectory, and the measured flight test trajectory. Again, close matching is observed in $x-$position, pitch, and velocity, yet the flight test $z-$position is consistently higher than the GPOPS trajectory.

In the third case, the obstacle avoidance method from Section 7.4.2 is used where only the obstacle location and a desired final position are specified (no path is given). Figure 7-5 compares the obstacle location, the GPOPS-generated trajectory and the measured flight test trajectory (no desired trajectory since none is specified). The aircraft is seen to clear the obstacle in both the GPOPS and flight trajectories. The same errors in $z-$position are observed.

While pitch and velocity are tracked in all three cases (i.e., the GPOPS commands to the system are being matched properly), flight test $z-$position is consistently higher than in the GPOPS trajectory. This implies that the model under-predicts the lift generated by the Clik. This could spawn from inaccuracies in the flat-plate lift coefficient or measurement of lifting surface area on the aircraft itself. To amend these discrepancies, the model can be iteratively tuned until better $z-$position matching is

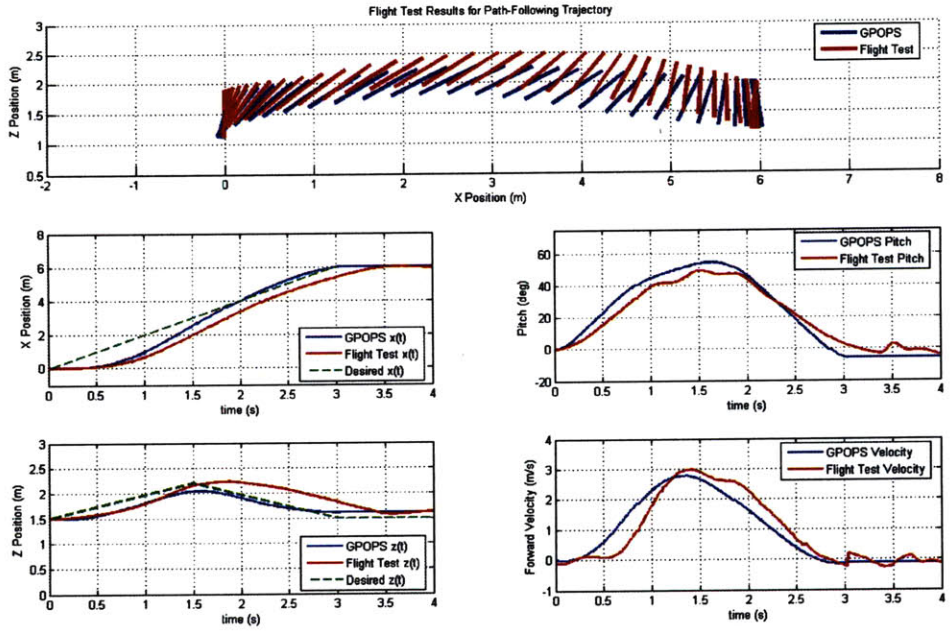

Figure 7-4: Flight test results for path following trajectory. Desired trajectory (green), GPOPS solution (blue) and measured flight test data (red) are shown.

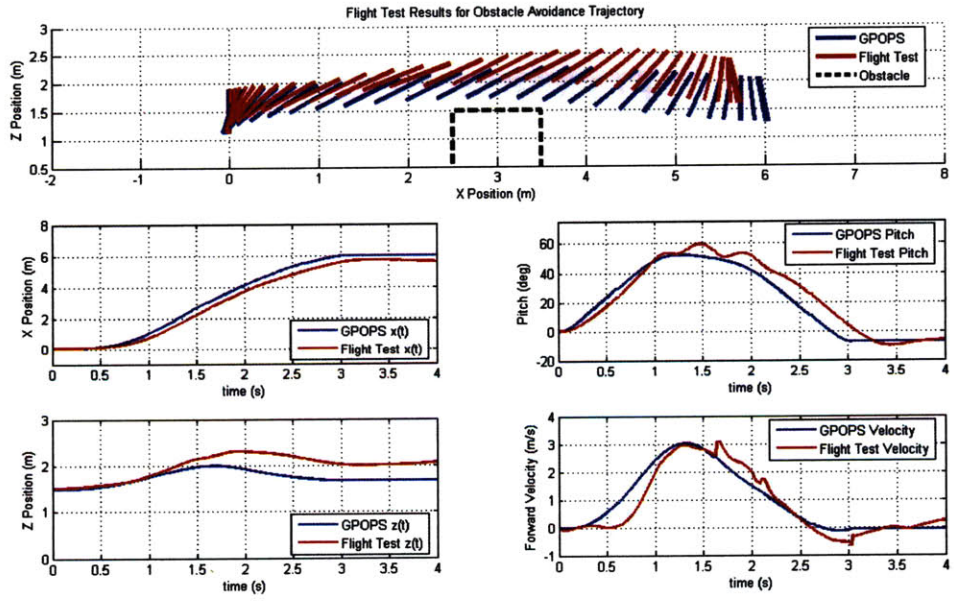

Figure 7-5: Flight test results for path following trajectory. Obstacle (black), GPOPS solution (blue) and measured flight test data (red) are shown.

obtained.

7.6 Summary

The trajectory optimization method presented makes use of a low-complexity, general model of an indoor flight vehicle and generates a dynamically feasible trajectory via an intuitive user-specified cost function. Flight tests confirm the feasibility of applying the method to an indoor UAV. The closed-loop implementation adds margin for model uncertainty and disturbances, as seen in the flight trajectory results. In addition, the closed-loop tracking can be examined to determine whether trajectory errors are the fault of the model or the control implementation. In the case of the results presented, it is seen that the model under-predicts the aircraft's lift and can be iteratively tuned as necessary.

The main limitation of this method is that currently it only supports generation of 2D trajectories. There is no fundamental limitation restricting the expansion to three dimensions as the method can be generalized to any set of dynamics. One potential issue with 3D trajectories is that more states must be added, making the optimization substantially more complex. Another limitation of the method is that the optimization cannot be done in real-time, and as of this writing takes 3-5 minutes to run on a machine with dual 2.4GHz AMD Opteron processors.

Chapter 8

Conclusion

8.1 Summary

This thesis investigates modeling, adaptive control and trajectory optimization methods as applied to indoor autonomous flight vehicles. Importance is placed on experimentally verifying theoretical results through flight testing. The successful implementation of adaptive control and trajectory generation techniques on indoor autonomous vehicles is considered a key aspect of the overall contribution.

Three indoor UAVs are presented in Chapter 2, each with unique control challenges. The RAVEN testbed is introduced as a facility enabling indoor autonomous flight of the vehicles and is the cornerstone of experimental validation throughout the thesis. Chapter 3 describes the adaptation of a quaternion-based attitude controller for the flight vehicles. An outer-loop horizontal velocity controller is designed to pass reference attitudes to the inner-loop controller, and the combined scheme forms the baseline controller for all three UAVs.

Chapter 4 presents a systematic design process for the use of $\mathcal{L}_1$ adaptive output feedback control in realistic flight control applications. The proposed process provides the control designer with an intuitive method linking relevant performance and robustness metrics to the selection of the $\mathcal{L}_1$ parameters $C(s)$ and $M(s)$. Flight test results verify the process for an indoor autonomous quadrotor, demonstrating that variations in the specified cost function produce the expected and desired physical re-

sponses. In flight tests comparing it with the baseline linear controller, the augmented $\mathcal{L}_1$ adaptive system shows definite performance and robustness improvements.

In Chapter 5, multi-input multi-output $\mathcal{L}_1$ adaptive control is applied to a three-wing tailsitter both in simulation and experiment. An inner-loop adaptation scheme is used such that the adaptive augmentation is able to operate independently of the non-linear baseline controller. The MIMO controller shows improved performance over the baseline system in spite of severe physical actuator failures both in simulation and flight testing.

Chapter 6 presents a dynamic modeling procedure for propeller-driven UAVs. A propeller model is adapted from previous work and is used to ascertain the slipstream velocities and profile. A simple flat-plate method is used to calculate control surface forces and moments and the predictions are compared to measured data with satisfactory results. The model predicts 3D forces and moments, yet is not too complex to use in trajectory generation and optimization.

In Chapter 7, a trajectory optimization method is presented that makes use of the low-complexity, general UAV model and generates dynamically feasible trajectories via an intuitive user-specified cost function. Flight tests confirm the feasibility of applying the method to indoor autonomous flight. A closed-loop trajectory implementation increases robustness to model uncertainty and disturbances. The method is limited in that it has only been tested for 2D trajectories, but can be extended to three dimensions by using the full 3D model of the aircraft.

8.2 Future Work

8.2.1 $\mathcal{L}_1$ Adaptive Output Feedback Control Design

Several limitations of the design process have been identified, most stemming from the non-convexity of the cost function. This acts to limit the complexity of the assumed forms of $C(s)$ and $M(s)$, preventing the potential benefits of higher-order filters from being explored. Future work is focused on converting the performance

and robustness metrics to a set of linear matrix inequality (LMI) constraints. Such a system is much more efficiently solved, thus having the potential to handle more complex solution forms. Some problems currently being faced are conservatism in conversion of the metrics to LMIs, and the inability of available numerical solvers to find initial feasible solutions.

8.2.2 MIMO $\mathcal{L}_1$ Adaptive Control

The primary area of future work is to apply the inner-loop MIMO $\mathcal{L}_1$ scheme to other types of flight vehicles to confirm that it is, in fact, independent of the type of baseline controller used. Application to the Clik and quadrotor should be straightforward since the adaptive controller is wrapped around the inner-most body rate loop. Also, work should be done to attempt to theoretically justify the dissipation modification used in the adaptive law.

8.2.3 Trajectory Optimization

Future work is planned to test the optimization method with the three-dimensional dynamic model to generate 3D trajectories. The performance of GPOPS with the larger state vector should be investigated, as it could slow down execution and make identification of a feasible trajectory more difficult without an initial guess. Also, tabulation of the dynamic model, use of some pre-computed trajectory as an initial guess, and reduction of the optimality tolerance could all serve to speed up the execution of the optimization allowing for real-time trajectory generation.

Finally, another area of future work is to combine individually generated trajectory segments in a piecewise fashion to create a larger, multi-phase flight maneuver. These trajectory segments can be pre-computed and stored in a library that can be accessed in real-time. The method would allow for a wide variety of possible maneuvers from a limited set of segments.

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] G. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. J. Tomlin, "The stanford testbed of autonomous rotorcraft for multi-agent control," in *the Digital Avionics System Conference 2004*, Salt Lake City, UT, November 2004.

[2] N. Knoebel, S. Osborne, D. Snyder, T. McLain, R. Beard, and A. Eldredge, "Preliminary modeling, control, and trajectory design for miniature autonomous tailsitters," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2006.

[3] V. Dobrokhodov, E. Xargay, I. I. Kaminer, M. Lizarraga, C. Cao, N. Hovakimyan, I. Gregory, and I. Kitsios, "Flight validation of metrics driven l1 adaptive control," in *Proc. AIAA Guidance, Navigation, and Control*, August 2007.

[4] N. Knoebel, "Adaptive control of a miniature tailsitter UAV," Master's thesis, Brigham Young University, December 2007.

[5] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-Time Indoor Autonomous Vehicle Test Environment," *IEEE Control Systems Magazine*, vol. 28, no. 2, pp. 51–64, April 2008.

[6] M. Valenti, B. Bethke, G. Fiore, J. How, and E. Feron, "Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Keystone, CO, August 2006, AIAA-2006-6200.

[7] J. How, B. Michini, J. McGrew, and D. Levine, "Aerobatic flight control experiments using raven," in *Proceedings of 23rd Bristol International Unmanned Air Vehicle Systems Conference*, 2008.

[8] B. D. Anderson and A. Dehghani, "Challenges of adaptive control: past, permanent and future," vol. 32, no. 2, pp. 123–135, December 2008.

[9] C. Cao and N. Hovakimyan, "Design and Analysis of a Novel L1 Adaptive Controller, Part I: Control Signal and Asymptotic Stability," in *Proc. American Control Conference*, 14–16 June 2006, pp. 3397–3402.

[10] ——, "Design and Analysis of a Novel L1 Adaptive Controller, Part II: Guaranteed Transient Performance," in *Proc. American Control Conference*, 14–16 June 2006, pp. 3403–3408.

[11] ——, "L1 adaptive output feedback controller for systems with time-varying unknown parameters and bounded disturbances," in *Proc. American Control Conference ACC '07*, 9–13 July 2007, pp. 486–491.

[12] D. Li, V. Patel, C. Cao, and N. Hovakimyan, "Optimization of the Time-Delay Margin of L1 Adaptive Controller via the Design of the Underlying Filter," in *Proc. AIAA Guidance, Navigation and Control*, August 2007.

[13] R. Hindman, C. Cao, and N. Hovakimyan, "Designing a high performance, stable l1 adaptive output feedback controller," in *Proc. AIAA Guidance, Navigation and Control*, August 2007.

[14] V. Patel, C. Cao, N. Hovakimyan, K. Wise, and E. Lavretsky, "L1 adaptive controller for tailless unstable aircraft in the presence of unknown actuator failures," in *Internation Journal of Control*, vol. 82, no. 4, April 2009.

[15] J. Wang, V. Patel, C. Cao, N. Hovakimyan, and E. Lavretsky, "Novel l1 adaptive control methodology for aerial refueling with guaranteed transient performance," in *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 1, February 2008.

[16] R. H. Stone, "Configuration design of a canard configured tail sitter unmanned air vehicle using multidisciplinary optimization," Ph.D. dissertation, University of Sydney, March 1999.

[17] D. Hunsaker, "A numerical blade element approach to estimating propeller flowfields," in *AIAA Aerospace Sciences Meeting and Exhibit*, 2007.

[18] E. Frazzoli, "Robust hybrid control for autonomous vehicle motion planning," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[19] A. Coates, P. Abbeel, and A. Ng, "Learning for control from multiple demonstrations," in *International Conference on Machine Learning*, 2008.

[20] F. Sobolic, "Agile flight control techniques for a fixed-wing aircraft," Master's thesis, Massachusetts Institute of Technology, 2009.

[21] A. Frank, J. S. McGrew, M. Valenti, D. Levine, and J. P. How, "Hover, transition, and level flight control design for a single-propeller indoor airplane," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, no. AIAA 2007-6318, Hilton Head, South Carolina, 20 - 23 August 2007.

[22] W. F. Phillips, C. E. Hailey, and G. A. Gebert, "Review of attitude representations used for aircraft kinematics," in *Journal of Aircraft*, vol. 38, no. 4, 2001.

[23] J. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality.* Princeton University Press, 2002.

[24] C. Cao and N. Hovakimyan, "L1 adaptive output feedback controller to systems of unknown dimension," in *Proc. American Control Conference ACC '07*, 9–13 July 2007, pp. 1191–1196.

[25] D. Li, N. Hovakimyan, C. Cao, and K. Wise, "Filter Design for Feedback-loop Trade-off of L1 Adaptive Controller: A Linear Matrix Inequality Approach," in *Proc. AIAA Guidance, Navigation and Control*, August 2008.

[26] J. Pomet and L. Praly, "Adaptive nonlinear regulation: Estimation from the lyapunov equation," in *Transactions on Automatic Control*, vol. 27, 1992, p. 729?740.

[27] D. Bernstein and A. Michel, "A chronological bibliography on saturating actuators," in *International Journal of Robust and Nonlinear Control*, vol. 5, 1995, pp. 375–380.

[28] W. F. Phillips, *Mechanics of Flight.* Wiley, 2004.

[29] S. Goldstein, "On the vortex theory of screw propellers," in *Royal Society of London Proceedings Series*, vol. 123, 1929, pp. 440–465.

[30] M. Drela. (http://web.mit.edu/drela/Public/web/xfoil/) Xfoil.

[31] D. Uhlig and M. Selig, "Post stall propeller behavior at low reynolds numbers," in *AIAA Aerospace Sciences Meeting and Exhibit*, 2008.

[32] S. M. LaValle, *Planning Algorithms.* Cambridge University Press, 2006.

[33] A. V. Rao, D. A. Benson, C. L. Darby, M. A. Patterson, C. Francolin, I. Sanders, and G. T. Huntington, "GPOPS: A matlab software for solving multiple-phase optimal control problems using the gauss pseudospectral method," in *ACM Transactions on Mathematical Software*, 2009.