

ABSTRACT

Title of thesis: **DESIGN AND IMPLEMENTATION
OF A CONTROL SYSTEM FOR A
QUADROTOR MAV**

Dean Bawek, Master of Science, 2012

Thesis directed by: **Dr. Inderjit Chopra**
Department of Aerospace Engineering

The quadrotor is a 200 g MAV with rapid-prototyped rotors that are driven by four brushless electric motors, capable of a collective thrust of around 400 g using an 11 V battery. The vehicle is compact with its largest dimension at 188 mm. Without any feedback control, the quadrotor is unstable. For flight stability, the vehicle incorporates a linear quadratic regulator to augment its dynamics for hover. The quadrotor's nonlinear dynamics are linearized about hover in order to be used in controller formulation. Feedback comes both directly from sensors and a Luenberger observer that computes the rotor velocities. A Simulink simulation uses hardware and software properties to serve as an environment for controller gain tuning prior to flight testing. The results from the simulation generate stabilizing control gains for the on-board attitude controller and for an off-board PC autopilot that uses the Vicon computer vision system for position feedback. Through the combined effort of the on-board and off-board controllers, the quadrotor successfully demonstrates stable hover in both nominal and disturbed conditions.

DESIGN AND IMPLEMENTATION OF A CONTROL SYSTEM
FOR A QUADROTOR MAV

by

Dean Bawek

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2012

Advisory Committee:

Dr. Inderjit Chopra, Chair/Advisor

Dr. J. Sean Humbert

Dr. Robert M. Sanner

© Copyright by
Dean Bawek
2012

Dedication

For Dr. William E. Lotz, who would never turn down a trip to the National Air & Space Museum.

Acknowledgments

I would like to give thanks to everyone who made this thesis possible. Many thanks go to Dr. Inderjit Chopra, my advisor, for giving me the opportunity to conduct research under his guidance, patience, and understanding. Thanks go as well to Dr. J. Sean Humbert and Dr. Robert M. Sanner for not only being on my committee, but also for setting a strong foundation for my knowledge of control theory and its application. Gratitude is also in order for Dr. Paul Samuel for his guidance and understanding of the challenges inherent with integrating hardware.

In addition, I would like to thank Dr. Vikram Hrishikeshavan for his support on this thesis and his willingness to listen. Thank you to Dr. Jason Pereira for taking me on the project and for his guidance, and to Shawn Westfall for introducing me to the quadrotor project in the first place. Also, thanks to Greg Germillion and UMD's AVL team for all of their support with getting the quadrotor interfacing properly with Vicon. Thanks to Cyrus Abdollahi and Bryan Patrick as well for the brainstorming late into the night trying to work out our controls homework. Thanks also to Nicole Johnston, for her love and patience.

Last but not least, thanks are in order for my family, whose care, support, and patience helped me through my educational career. Thanks to my parents Brad and Alison Bawek for their encouragement in all of my endeavors, my brother Shane Bawek for always being there for me, my grandmother Ruth Lotz for her kindness, and to my grandfather, the late Dr. William E. Lotz, for fostering my passion for science and knowledge.

Table of Contents

List of Tables	v
List of Figures	v
List of Abbreviations	vii
1 Introduction	1
1.1 Motivation	3
1.2 Control Implementation	4
1.3 Contemporary Quadrotors	5
1.4 Objectives and Overview	7
2 Configuration	10
2.1 Structure	11
2.2 Microcontroller	15
2.3 Inertial Measurement Unit	16
2.3.1 Kalman Filter	17
2.4 Rotors	23
2.5 Speed Controllers	28
2.6 Rotor Dynamics	31
3 Control	36
3.1 Quadrotor Rotational Dynamics	36
3.2 System Linearization	44
3.3 LQR	56
3.4 Luenberger Observer	64
4 Simulation and Testing	67
4.1 Simulation	68
4.1.1 Attitude Control	69
4.1.2 Position Control	76
4.2 Flight Testing	86
4.2.1 Implementation and Testing in Vicon	86
4.2.2 Quadrotor Flight Testing in Vicon	87
4.2.3 Quadrotor Flight Test Results	91
4.3 Concluding Remarks	99
5 Conclusions	101
5.1 Future Work	103
Bibliography	105

List of Tables

2.1	Rotor Properties	25
3.1	System Properties for LQR	64
4.1	Position Control Gains	78

List of Figures

1.1	Basic controls of a quadrotor.	2
1.2	Photographs of the Breguet-Richet and Oehmichen quadrotors.	3
1.3	A small selection of quadrotors.	8
2.1	Exploded view of the quadrotor and its constituent parts.	11
2.2	CAD representation of quadrotor frame.	13
2.3	Custom PCB for connecting avionics components.	14
2.4	Arduino Pro Mini microcontroller.	14
2.5	Depiction of data flow to and from microcontroller and other components.	16
2.6	Basic wiring diagram of avionics systems.	17
2.7	Power spectral density of sensor noise.	21
2.8	Recorded sensor data corrected using a Kalman filter simulated in MATLAB.	24
2.9	Test apparatus for rotor performance testing.	26
2.10	Rotor test data.	27
2.11	Example of a PWM signal assuming a 5-volt input.	29
2.12	Rotor thrust and torque relative to PWM input.	31
2.13	Example of rotor dynamics test case. In this case, the hover input is increased by a step of $30 \mu s$	33
2.14	When compared to test data, the first-order model accurately approximates the rotor dynamics.	34
2.15	Rotor models generated from different test runs were averaged to approximate the rotor dynamics.	34
3.1	Coordinate frame for common symmetric quadrotor configuration.	37
4.1	Depiction of data flow for the simulation.	69
4.2	Results of hover simulation with zero initial conditions.	71
4.3	Error between predicted and simulated rotor states used to verify the Luenberger observer.	72
4.4	Results of hover simulation with nonzero initial conditions.	72
4.5	Results of hover simulation with impulse.	73
4.6	Attitude results of hover simulation with different \mathbf{Q} matrices.	73
4.7	Results of hover simulation with LQR controller using \mathbf{Q}_2	74

4.8	An example of the unstable oscillation induced by increasing the proportional gain for the z-axis.	78
4.9	Results of hover simulation with position control and zero initial conditions.	80
4.10	Results of hover simulation with position control, \mathbf{K}_2 controller, and zero initial conditions.	81
4.11	Results of hover simulation with position control and nonzero initial conditions.	82
4.12	Results of hover simulation with position control, \mathbf{K}_2 controller, and nonzero initial conditions.	83
4.13	Results of hover simulation with position control and impulse.	84
4.14	Results of hover simulation with position control, \mathbf{K}_2 controller, and impulse.	85
4.15	Laboratory setup for quadrotor flight testing in Vicon (one of eight cameras shown).	87
4.16	Diagrams depicting tests conducted in flight.	89
4.17	Results of three flight tests with position control in undisturbed hover.	90
4.18	Video frames from the quadrotor impulse flight test.	91
4.19	Results of flight test with position control in hover	93
4.20	Results of flight test with position control and \mathbf{K}_2 controller in hover.	94
4.21	Results of flight test with position control and impulse.	95
4.22	Results of flight test with position control, \mathbf{K}_2 controller, and impulse.	96
4.23	Results of flight test with position control and wind.	97
4.24	Results of flight test with position control, \mathbf{K}_2 controller, and wind.	98

Notation

0	Zero matrix with appropriate dimensions
<i>A</i>	Rotor-disk area (m ²)
A	State matrix
<u><i>a</i></u>	Acceleration vector
<i>a_i</i>	Acceleration along the i-axis
B	Control matrix
C	Observation matrix
<i>C_P</i>	Rotor power coefficient
<i>C_Q</i>	Rotor torque coefficient
<i>C_T</i>	Rotor thrust coefficient
<i>g</i>	Gravity constant (m/s ²)
<u><i>g</i></u>	gravity vector (m/s ²)
H	Process noise covariance matrix
I	Identity matrix with appropriate dimensions
<i>I_i</i>	Moment of Inertia along the i-axis (kg·m ²)
<i>J</i>	Rotor inertia (kg·m ²)
K	LQR control gain matrix
<i>K_i</i>	Integral motor gain (μs/m)
<i>K_m</i>	Motor gain
<i>K_p</i>	Proportional motor gain (μs·s/m)
<i>K_u</i>	Unstable motor gain (μs·s/m)
<i>k_Q</i>	Torque constant
<i>k_T</i>	Thrust constant
<i>L</i>	Angular momentum (rad·kg·m ² /s ²)
L	Luenberger observer matrix
<i>l</i>	Moment arm of motors (m)
M_{ij}	Matrix comprising row i and column j of a partitioned matrix
<i>m</i>	Vehicle mass (kg)
N	Sensor measurement noise covariance matrix
<u><i>n</i></u>	Accelerometer measurement vector (m/s ²)
<i>n_i</i>	Accelerometer measurement along the i-axis (m/s ²)
P	Error covariance matrix
<i>Q</i>	Rotor torque (N·m)
Q	State accuracy cost matrix
<i>Q_i</i>	Torque of i-th rotor (N·m)
R	Control cost matrix
<i>R</i>	Rotor radius (m)

T	Rotor thrust (N)
$\sum T$	Net thrust (N)
T_i	Thrust of i-th rotor (N)
T_u	Period of oscillations induced by K_u (s)
t	Time (s)
t_r	Rise time (s)
U	Motor input transfer function
\underline{u}	Vector of linearized system inputs (μs)
u_i	Input to i-th motor (μs)
\underline{u}_p	Vector of autopilot inputs (μs)
\underline{W}	Ω^2 (rads ² /s ²)
W_i	Ω^2 of the i-th rotor (rads ² /s ²)
W_{ss}	Steady-state rotor velocity (rads ² /s ²)
\mathbf{W}	Matrix form of motor transfer function
x	Position along the x-axis (m)
\underline{x}	Vector of linearized system states
$\hat{\underline{x}}$	Vector of system state estimates
y	Position along the y-axis (m)
\underline{y}	Vector of linearized system measurements
z	Position along the z-axis (m)
\underline{z}	Vector of actual system states
\underline{z}^*	Vector of system states at reference condition
α	Motor pole
η	Sensor measurement noise
$\underline{\mu}$	Vector of actual system inputs
$\underline{\mu}^*$	Vector of system inputs at reference condition
ν	Sensor process noise
Ω	Rotor rotational velocity (rad/s)
$\Sigma\Omega$	Total rotor rotational velocity (rad/s)
Ω_i	Rotational velocity of the i-th rotor (rad/s)
ω	Angular velocity (rad/s)
ω_p	Rate of precession (rad/s)
ϕ	Roll attitude (rad)
ψ	Heading (rad)
θ	Pitch attitude (rad)
ρ	Air density (kg/m ³)
τ	Torque (N·m)

AGRC	Alfred Gessow Rotorcraft Center
ARE	Algebraic Ricatti Equation
AVL	Autonomous Vehicles Laboratory
CAD	Computer-Aided Design
DAQ	Data Acquisition
DARE	Discrete Algebraic Ricatti Equation
EEPROM	Electrically Erasable Programmable Read-Only Memory
EPFL	cole Polytechnique Fdrale de Lausanne
ESC	Electronic Speed Controller
FM	Figure of Merit
GRASP	General Robotics Automation Sensing and Perception
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
LQR	Linear Quadratic Regulator
LiPo	Lithium-Polymer
MAV	Micro Air Vehicle
MBC	Model Based Control
MEMS	Micro Electro Mechanical System
MIMO	Multi-Input-Multi-Output
PC	Personal Computer
PCB	Printed Circuit Board
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
PWM	Pulse Width Modulation
RPM	Rotations Per Minute
SRAM	Static Random-Access Memory
VI	Virtual Instrument
VTOL	Vertical Take Off and Landing
UAV	Unmanned Air Vehicle

Chapter 1

Introduction

The popularity of quadrotors has increased in recent years, due in part to the mechanical simplicity of their design. Quadrotors utilize four rotors to lift and control the vehicle without the need for complex rotor mechanisms. Instead of controlling the vehicle via rotor blade articulation using a swashplate assembly, a quadrotor simply changes the rotational velocity of each of its rotors (Fig. 1.1). Until the mid-twentieth century, without the innovation of articulated blades to overcome the challenges of vertical flight the quadrotor configuration seemed a logical choice in the pursuit of a VTOL aircraft for Charles Richet and the Breguet brothers in 1906 (Fig. 1.2) [19]. Counter-rotating rotor pairs would take care of any torque imbalances, and the thrust generated by four sets of propellers would lift the vehicle given enough power. Thus the concept of a quadrotor is far from new, but the Breguet-Richet quadrotor encountered challenges that exemplify the difficulty of implementing such a configuration at a large scale. For the time period, and also for large scale aircraft, the original quadrotor relied on a gasoline powered engine. The Breguet-Richet design would require a complex transmission system to try and control each rotor independently. Even if a contemporary combustion engine were to be used for each of the four rotors, a considerable amount of weight would incur. Additionally, the engines would have been very sluggish in making the necessary

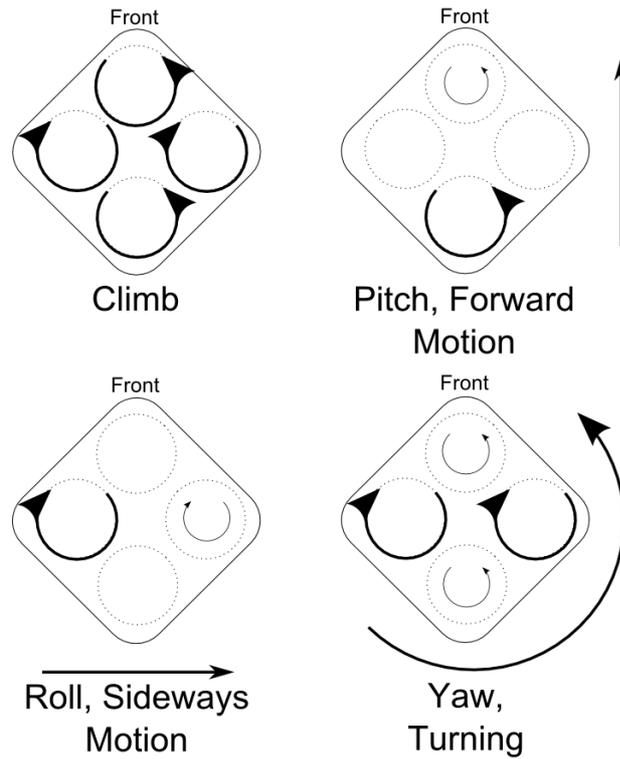
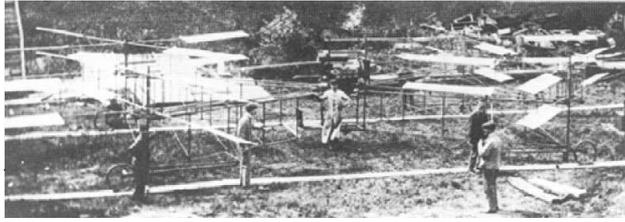
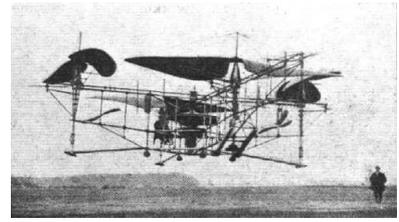


Figure 1.1: Basic controls of a quadrotor.

torque changes for stabilizing the vehicle [2]. In the 1920's, Etienne Oemichen tried to stabilize his quadrotor using a series of additional propellers, but the vehicle was too heavy to fly without the aid of a hydrogen balloon (Fig. 1.2) [29]. The configuration's promised simplicity seems to decay from limitations in methods of actuation at large scales. However, quadrotors became viable at smaller scales, which has been made possible in part due to electronics miniaturization and an interest in unmanned aircraft.



(a)



(b)

Figure 1.2: Photographs of the (a) Breguet-Richet and (b) Oehmichen quadrotors.

1.1 Motivation

Unmanned systems offer users the advantage of sending eyes into a location that would otherwise be hostile to a human. These tasks are often designated as dull, dirty, and dangerous, where a human would run the risk of fatigue from monotonous, long, or difficult missions, or exposure to hazardous materials and waste, or attracting hostile attention. Ground-based robots have been performing many of these tasks since before the 1980's, but it has not been until the last two decades that the world has seen unmanned aerial drones in wide use. These UAVs have demonstrated their efficacy at retrieving vital tactical information without endangering the lives of skilled pilots. There are times, however, when a closer look into a cluttered area, such as a city, is necessary. The ever-increasing rate of urban sprawl makes such a scenario inevitable, and so the demand for MAVs increases. Conversely, stringent size requirements and problematic aerodynamics hamper MAV development. Fundamental aerodynamic principles lose efficacy at low Reynolds numbers, so fixed- and rotary-wing aircraft drastically become less efficient at the MAV-scale. Given a robust controller, quadrotors are more stable platforms than conventional rotors and can tolerate a far more overall center of gravity shift.

Every hinge and linkage that makes a helicopter operable becomes a liability at such a small scale, in which the structural integrity of the mechanisms becomes suspect. Durability concerns undermine one of the primary appeals of MAVs, which is their potential to navigate small, cluttered environments where the risk of mishap increases. Despite a lack of success at larger scales, the advantages of the quadrotor configuration come to light within the MAV and small UAV range. Using electric motors and fixed, unarticulated rotors, quadrotors can perform as functional VTOL MAV's without the mechanical complexity of conventional helicopter or ornithopter configurations. Quadrotors use four rotors for thrust, and thus have high control authority and payload capacity. However, the increased simplicity comes at a cost to stability. The four rotor configuration is unstable, but the benefits of the quadrotor make it a challenge worth approaching.

1.2 Control Implementation

Quadrotors are unstable in their base configuration and require a control system to augment stability. Control system design, as with any design process, can be approached by multiple avenues to achieve a desired goal while trying to balance advantages and disadvantages. In the case for the quadrotor, the objective is to develop a system that manipulates the rotors to render a stable vehicle. The control methodology to perform this task can take several forms, however the most notable difference is between on-line tuning and model-based system design. Using on-line techniques such as PID allow the designer to formulate a controller without prior

knowledge of the dynamical system model, though this can entail considerable time dedicated to adjusting the control parameters. Tuning in such a manner generally assumes that each state of a MIMO system can be addressed as an individual SISO control problem, which neglects cross-coupling in the system dynamics [22]. As will be detailed in Chapter 3, the quadrotor's dynamics are coupled, so such an approach should be used with caution. Additionally, since the quadrotor is intended for flight, gain adjustments must be performed on a gimble or other test hardware to ensure the safety of the vehicle as well as the researcher. Conversely, the model-based approach allows for design to be performed almost entirely via computation and simulation. Provided that the model is known and accurate, any number of techniques can be used to formulate and evaluate the control system. Bouabdallah has formulated and analyzed several controllers through both simulation and flight testing, using a dynamic model of a quadrotor [6]. The quadrotor model is nonlinear and several of the techniques tested use the Lyapunov theorem as the basis for a nonlinear controller. For the sake of safety, simplicity, and repeatability, the quadrotor in this thesis uses an LQR controller formulated with a linearized model of the system.

1.3 Contemporary Quadrotors

As there are different methods of control design, there were a multitude of quadrotors built by both researchers and commercial companies in pursuit of developing a stable hovering unmanned platform (Fig. 1.3). As mentioned previ-

ously, Bouabdallah at the EPFL autonomous vehicle laboratory in Switzerland built the OS4 quadrotor, which studied implementation of PID, adaptive LQR, and Lyapunov-based stabilization algorithms with substantial success [6]. The OS4 also established a groundwork simultaneous design of quadrotor configuration and control, as well as simulation tools. Castillo et al designed a nonlinear nested saturation controller implemented on a modified commercial quadrotor, the Draganfly [8]. The Draganfly itself was offered for sale for consumer use as well as cinematic and surveillance purposes [11]. The AVL at the University of Maryland used a modified X-UFO quadrotor for optic-flow station-keeping tests, as well as a platform for embedded systems testing [25]. Stanford and Berkeley's STARMAC quadrotors have been used for networked control systems research [14]. In a similar vein the GRASP laboratory at Penn State University demonstrated cooperative quadrotor flight [2]. Additionally, the GRASP laboratory tested the aerobatic capabilities of a commercial quadrotor when coupled with Vicon vision tracking. Though their quadrotors operated using an in-house global reference system, the GRASP laboratory was also working to eliminate reliance on an external system with implementation of SLAM using a laser range finder [12]. The GRASP and other aforementioned quadrotors were technically small UAVs as opposed to MAVs, since their dimensions were substantially outside DARPA's specifications of 6 inches maximum length, width, or height [21]. There had been several quadrotors in development that fall under MAV specifications, however. The AVL's Micro Quad was the subject of several theses exploring hardware and software design ([13], [22]). The Micro Quad implemented a PID control algorithm for attitude and translational control, making use of the

Vicon global reference system as well. The Mote microcontroller and sensor suite was used for both on- and off-board processing of flight data and also kept the vehicle's size small. Another quadrotor MAV, the CrazyFlie, weighed 20 grams. The low weight was achieved by designing the PCB to be the vehicle structure as well [9]. The body of the vehicle contained all of the embedded avionics required for stable flight by a remote pilot. However, these quadrotor MAVs suffered from limited payload capacity.

1.4 Objectives and Overview

The quadrotor described in this thesis fills the niche between the larger, heavy-lift quadrotors and compact, low-payload MAVs, thus providing substantial payload capabilities at a compact size. In addition to fabricating the airframe and selecting the hardware necessary to achieve flight, a control algorithm is designed and tuned to work from an on-board embedded microcontroller for stability in hover. The controller is formulated via a model based LQR algorithm that includes motor dynamics in the system model to increase controller efficacy. Due to sensor limitations, the rotor velocities are estimated by a Luenberger observer using the rotational system dynamics. The controller is implemented and tested on the quadrotor with the aid of an external autopilot using the Vicon visual positioning system to ensure repeatability.

Specifically, Chapter 2 will discuss the selection of hardware, fabrication of the airframe, and the methods used to determine the physical properties of the



(a)



(b)



(c)



(d)



(e)

Figure 1.3: A small selection of quadrotors including the (a) Ascending Technologies Hummingbird, (b) Draganfly X4, (c) STARMAC, (d) AVL's Micro Quad, and (e) CrazyFlie.

quadrotor. Formulation of the vehicle dynamics control and estimation algorithms used to achieve stability in hover are covered in Chapter 3. The algorithms are tested, analyzed, and adjusted using a combination of simulation in Matlab as well as flight testing with Vicon autopilot in Chapter 4. Finally, Chapter 5 discusses the results of the testing and evaluates the relative efficacy of the control algorithm in stabilizing the quadrotor. The chapter also presents conclusions that can be gathered from the experimentation as well as suggestions for further research.

Chapter 2

Configuration

The fabricated quadrotor was designed around its propulsion and on-board electronics (Fig. 2.1). Vehicle flight was achieved using rapid-prototyped rotors driven by four brushless DC motors, capable of a collective thrust of nearly 400 g using a 3-cell Lithium-Polymer (LiPo) battery. The motors collectively drew currents up to 8 A, which was more than enough to damage on-board electronics if connected to the microcontroller directly. To protect the electronics, Dualsky 6 A brushless electronic speed controllers (ESC's) were implemented. Also, each ESC drove a motor with a three-phase signal converted from the output of an Arduino Pro Mini with an ATmega328 microcontroller. The control signals were based on a combination of sensor feedback from a Sparkfun 6 Degree-of-Freedom Inertial Measurement Unit (IMU) and control inputs from an external PC delivered via bluetooth. The electronics were held together by a custom-designed PCB and received power via a voltage regulator board, which converted the 11.1 V power supply to 5 V, as well as 3.3 V for auxiliary purposes. Placing a voltage regulator between the power supply and the avionics reduced electronic noise from motor operation. The electronics and propulsion system were mounted on a frame of Dragonplate carbon-fiber composite material with a foam core for its durability, low weight, and bending stiffness for the quadrotor arms. A frame of carbon rods served

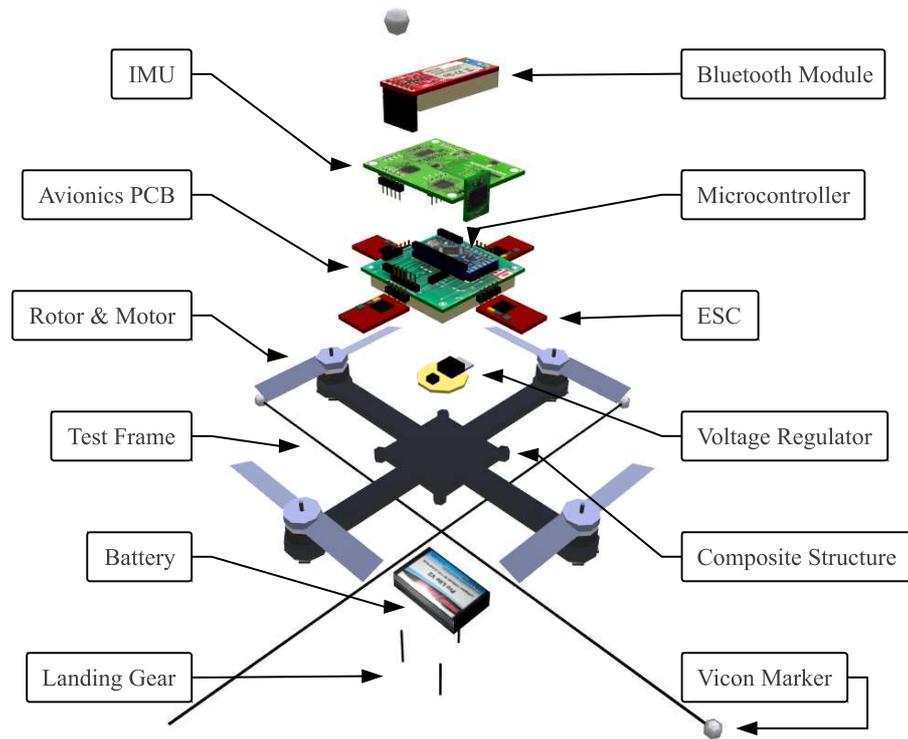


Figure 2.1: Exploded view of the quadrotor and its constituent parts.

as both landing support and protection for the quadrotor. Taking a closer look at these parts and their relations will reveal the trade-off between structural simplicity and digital complexity.

2.1 Structure

The motors and other components were mounted symmetrically on an X-frame (Fig. 2.2). A carbon fiber foam sandwich material from Dragonplate constituted the vehicle body in order to keep the airframe as light as possible. The machinable 6 mm Depron single-layer carbon fiber composite was cut using computer-controlled milling. The dimensions of the frame were designed to keep all components as

close as possible without interference. Additionally, the frame was such that four units could be manufactured from one 12 in x 12 in sheet of the composite material. Though foam appeared to separate the two carbon-fiber layers, there were also small pegs spaced amongst the foam to add structural rigidity. Despite the added support, the foam layer tended to tear laterally near the motor mounts from vibration. To overcome this problem, a thin layer of epoxy was applied on the exposed foam inside of motor mount holes, as well as on the outside and along part of the arms, thereby eliminating this problem. The epoxy also bound the frame to the metal mounts that came with the motors. A small hole was drilled at the end of each arm to allow access for the motor mounting set screws. To protect the electronics from adverse motor vibrations, a foam pad rested between the carbon frame and the avionics PCB. The PCB itself was held in place using nylon screws that were inserted via through-holes on the PCB and screwed into nylon nuts glued to the quadrotor frame. The screws kept the PCB in place, while still allowing the foam to flex and act as a simple vibration isolator to protect sensitive electronics. The electronics were plugged into the custom PCB using various connectors, allowing modularity for the sake of easy component replacement (Fig. 2.3). The only exceptions to this were the ESCs, which were soldered into the tabs at each side of the PCB to avoid the weight that would be added by connectors. With the speed controllers extended away from the main PCB, the PCB's pin locations were designed around microcontroller and IMU placement.

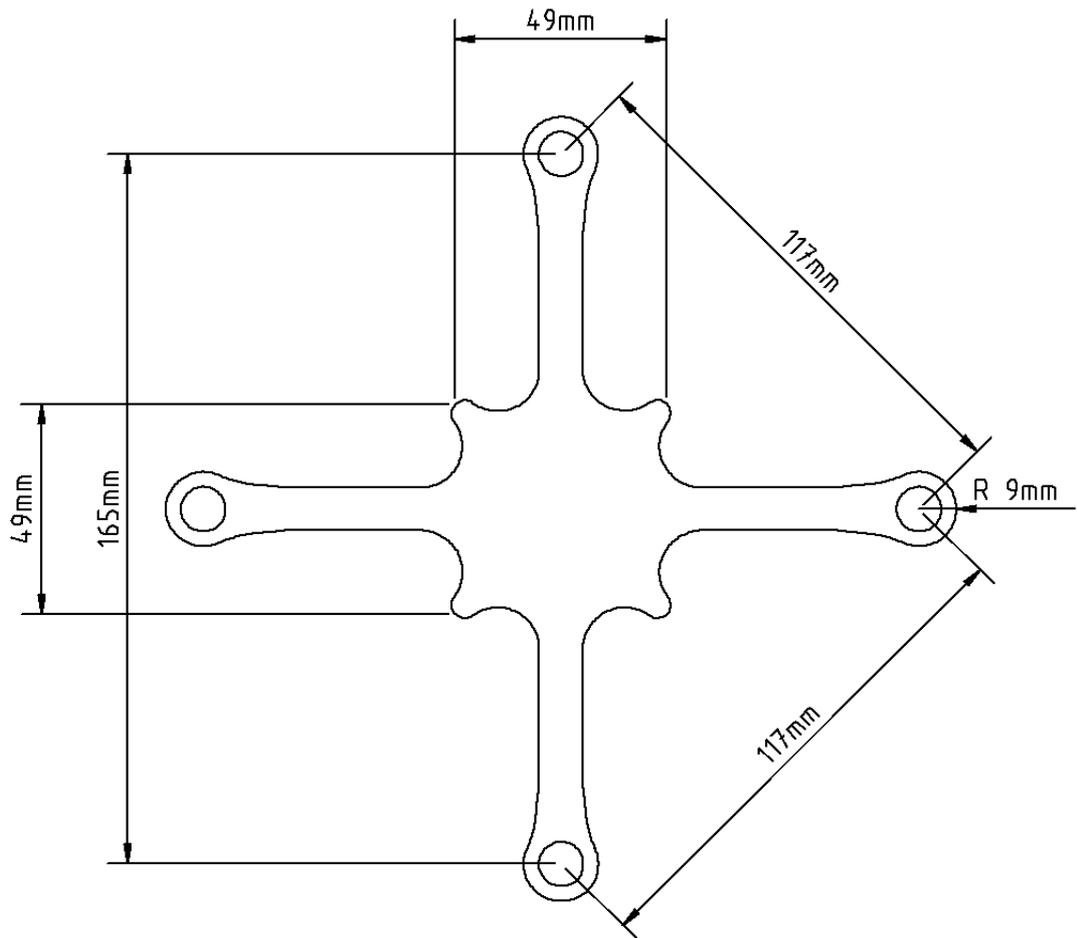


Figure 2.2: CAD representation of quadrotor frame.

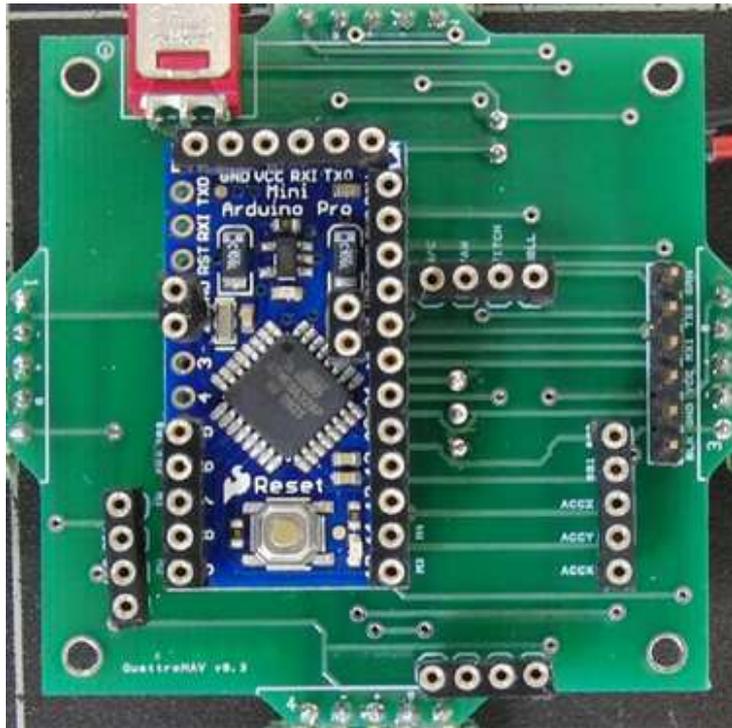


Figure 2.3: Custom PCB for connecting avionics components.

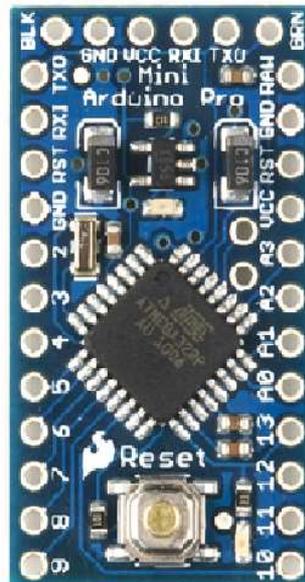


Figure 2.4: Arduino Pro Mini microcontroller.

2.2 Microcontroller

At the center of the on-board avionics package was an Arduino Pro Mini microcontroller (Fig. 2.4) [3]. The microcontroller handled all digital filter and controls implementation for the quadrotor. An ATmega328 chip provided 10-bit analog-to-digital conversion for reading the sensor data, filtered the data digitally, processed pilot inputs from the receiver, and modified the commands to stabilize the quadrotor (Fig. 2.5). Using an embedded system introduced restrictions, too. Though a processor clock speed of 16 MHz was acceptable, circumventing the restrictive 2 kB of memory offered additional challenges. These restrictions could be avoided by controlling the quadrotor via wireless communications with an off-board system like a laptop or desktop PC performing the appropriate filter, observer, and feedback control calculations. However, if the wireless connection failed, the quadrotor would lose control. This fact, coupled with the risk of instability induced by transmission latency, range limitations, and signal instability, made for a strong case supporting on-board attitude control. Providing on-board stability augmentation allowed for the quadrotor to maintain a level attitude even if the external PC dropped the connection. Storing large matrices into the ATmega's program memory compensated for the low flash memory and allowed for expanded system capability. The programs stored in the Arduino were programmed using the Arduino's free IDE. The programming environment was open-source, making use of AVR-dude and GNU compilers, which used the C++ programming language. These qualities made the Arduino platform popular amongst hobbyists, but the current quadrotor attempted

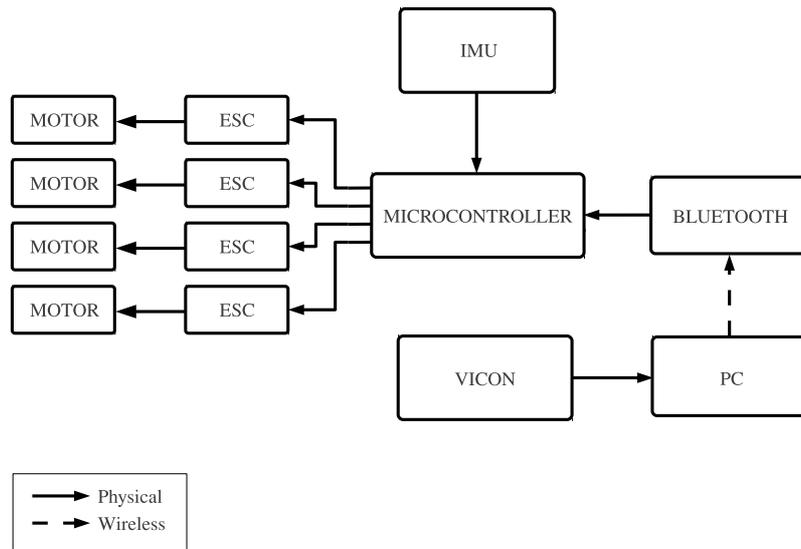


Figure 2.5: Depiction of data flow to and from microcontroller and other components.

to demonstrate the research potential for the microcontroller.

2.3 Inertial Measurement Unit

For an on-board system to augment the quadrotors stability, the microcontroller needed information on the vehicles attitude. MEMS electronics were compact, light-weight, and inexpensive integrated sensing packages that provided the required state measurements for a feedback system. The Sparkfun 6 DOF v4 IMU consisted of two PCB's: a sensor board with a tri-axis accelerometer, two dual-axis gyros, and a magnetometer, and a control board with microcontrollers and a bluetooth module for data transmission. The component boards were modular, and the sensor board was removed to operate independent of the control board. Since the sensors were analog, they were sensitive to changes in voltage and any elec-

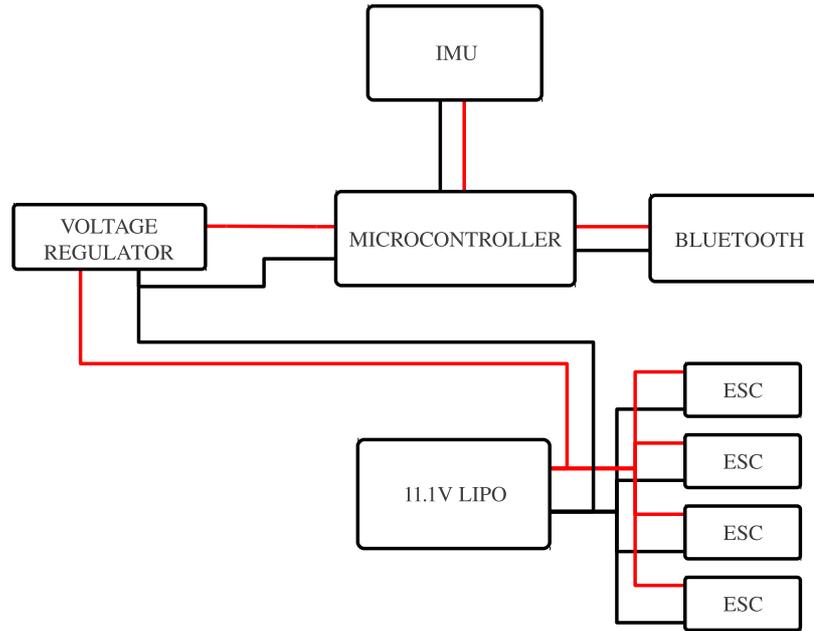


Figure 2.6: Basic wiring diagram of avionics systems.

tronic noise. The IMU shared the power system with the microcontroller, utilizing a 5 V voltage regulator connected to the main power supply (Fig. 2.6). The ESCs themselves had on-board regulators with 5V output, but the voltage fluctuated with motor startup and negatively affected sensor calibration. Once the electronics were fully functional, the sensor readings were taken, averaged, and then zeroed while the quadrotor was on the ground readying for takeoff. However, due to rotor vibrations, it was important to further process sensor data to eliminate noise.

2.3.1 Kalman Filter

Data from the on-board gyroscopes and accelerometer could not be used independently. The accelerometer was extremely sensitive to vibration and became

noisy during flight even with a foam damper underneath the avionics PCB. Gyro data appeared relatively unaffected by motor vibrations, but could not be integrated for attitude determination because of a drift component that afflicted MEMS gyros and caused integrations to diverge quickly [5]. A discrete Kalman filter took care of both issues by effectively combining sensor data; the gyro measurements corrected accelerometer noise, and the accelerometer compensated for the gyroscopes drift component. The accelerometers contributed by essentially measuring the pitch and roll angles of the quadrotor. Since the sensors measured acceleration in the quadrotor's body-frame, then they were measuring the components of the gravity vector in the body-frame, thus

$$\underline{g}_B = g \begin{bmatrix} -\sin(\theta) \\ \cos(\theta)\sin(\phi) \\ \cos(\theta)\cos(\phi) \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \underline{a}_B \quad (2.1)$$

where \underline{g}_B was the gravity vector in the body frame and \underline{a}_B was the body acceleration vector. The accelerometer measured accelerations in g's, so the measurements were then related by

$$\underline{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} a_x/g \\ a_y/g \\ a_z/g \end{bmatrix} = \begin{bmatrix} -\sin(\theta) \\ \cos(\theta)\sin(\phi) \\ \cos(\theta)\cos(\phi) \end{bmatrix} \quad (2.2)$$

where \underline{n} was the vector of accelerometer measurements. The above equations as-

sumed that the thrust vectors from the motors were all perpendicular to the x-y (horizontal) plane of the quadrotor and thereby had no components along the x and y body axes. Assuming small angles as well as setting aside the z-acceleration term provided further simplification, resulting in

$$n_x = -\theta \tag{2.3}$$

$$n_y = \phi \tag{2.4}$$

As long as the quadrotor operated around level hover and translational accelerations were negligible, the assumptions held well. Using the small angle approximation, assume the accelerometers directly measured the attitudes. Thus, the sensor noise dynamics could be used in formulation of a Kalman filter without modification.

Traditionally, a Kalman filter required knowledge of the system dynamics. Since the system was embedded, and memory was a premium, a discrete Kalman filter that accounted only for the sensor dynamics sufficed. The generic discrete-time sensor dynamics for attitude angles and rates were as follows:

$$\begin{aligned} \theta_{k+1} &= \dot{\theta}_k \Delta t + \theta_k \\ \dot{\theta}_{k+1} &= \dot{\theta}_k \end{aligned} \tag{2.5}$$

or

$$\underline{\theta}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \underline{\theta}_k \quad (2.6)$$

where Δt was the sampling period, $\underline{\theta}_k^T = [\theta_k \quad \dot{\theta}_k]$, $\dot{\theta}_k$ was the gyroscope measurement at sample k , and $\theta_k = n_{x,k}$ from the relevant accelerometer axis. The sensor dynamics applied to roll as well, substituting θ for ϕ , and n_x for n_y . As mentioned previously, the IMU measurements contained noise, so introducing process noise ν and measurement noise η into the sensor dynamics gave

$$\begin{aligned} \underline{\theta}_{k+1} &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \underline{\theta}_k + \underline{\eta}_k \\ \underline{y}_{\theta,k} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \underline{\theta}_k + \underline{\nu}_k \end{aligned} \quad (2.7)$$

For effective implementation of a Kalman filter, the noise statistics were measured. Sensor readings were taken while the quadrotor was at rest to determine the measurement noise ν . Again, it was expected that the quadrotors motors operated near hover input to find the properties of the process noise η . Once collected, the data was analyzed to determine the noise covariance of each sensor axis (Fig. 2.7). Kalman filters required the assumption that the noise could be characterized as white, though this did not appear so for the process noise statistics. For the sake of

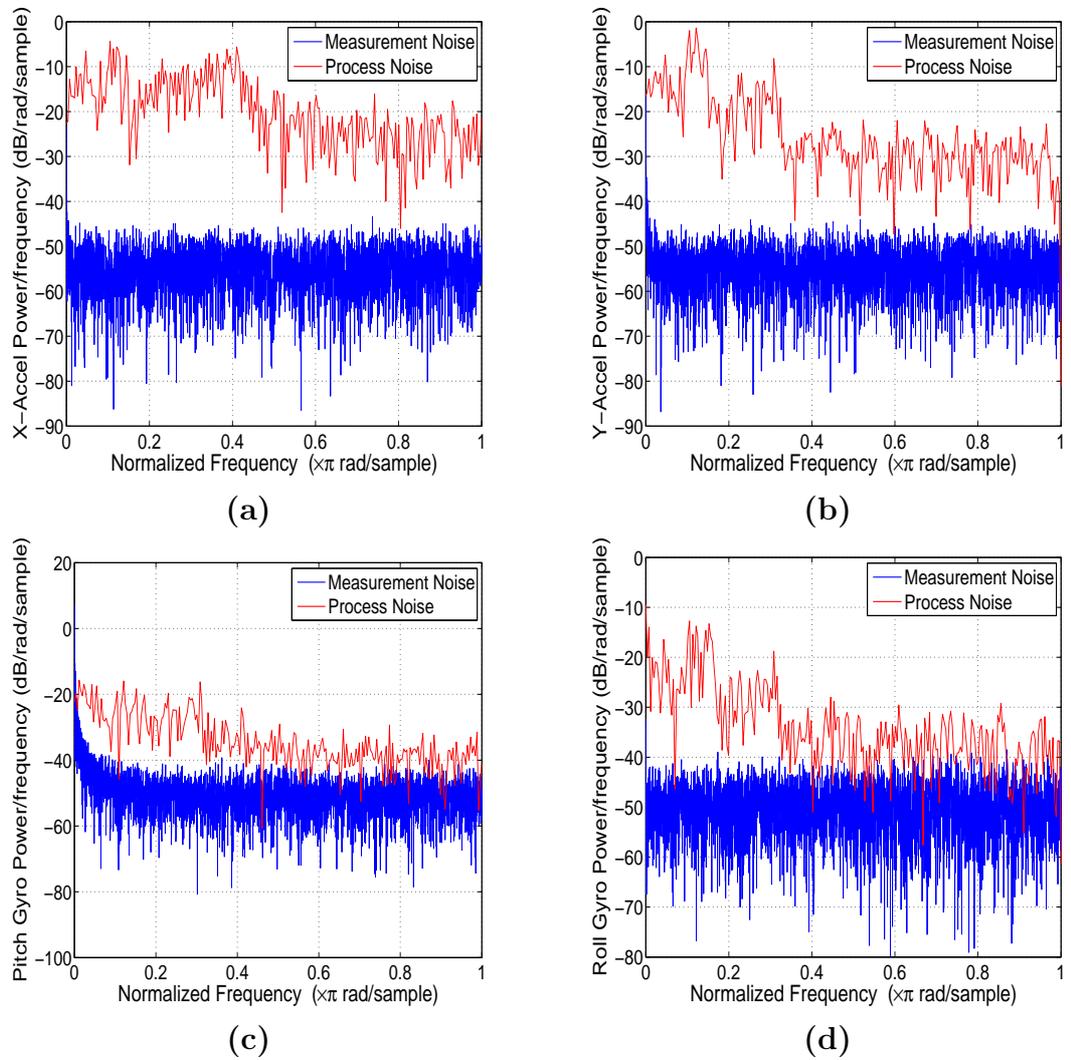


Figure 2.7: Power spectral density of sensor noise, which includes the (a) x-axis accelerometer, (b) y-axis accelerometer, (c) pitch gyro, and (d) roll gyro. Each PSD plot shows the sensor noise at rest (Measurement Noise) and while the motors are running in the hover condition (Process Noise).

simplicity and because of limited processing power, the process noise was the overall covariance of the roll and pitch gyroscopes. The values were then used to build the noise covariance matrices, with the x-accelerometer and pitch-gyro process noise covariances with vibration inserted into H_θ , and the y-accelerometer and roll-gyro process noise covariances with vibration populating H_ϕ . The sensors were assumed to have no correlation with one another, so the covariance matrices were diagonal:

$$\begin{aligned}
 H_\theta &= \begin{bmatrix} \text{var}[n_x] & 0 \\ 0 & \text{var}[q] \end{bmatrix} = \begin{bmatrix} 0.0973 & 0 \\ 0 & 0.0040 \end{bmatrix} \\
 H_\phi &= \begin{bmatrix} \text{var}[n_y] & 0 \\ 0 & \text{var}[p] \end{bmatrix} = \begin{bmatrix} 0.0708 & 0 \\ 0 & 0.0080 \end{bmatrix}
 \end{aligned} \tag{2.8}$$

The measurement noise was white for all sensors (Fig. 2.7), and was used to compute the measurement covariances for pitch and roll:

$$\begin{aligned}
 N_\theta &= \begin{bmatrix} 1.6115 \times 10^{-5} & 0 \\ 0 & 3.779 \times 10^{-5} \end{bmatrix} \\
 N_\phi &= \begin{bmatrix} 1.6038 \times 10^{-5} & 0 \\ 0 & 5.1827 \times 10^{-5} \end{bmatrix}
 \end{aligned} \tag{2.9}$$

These computed values were used on-board the quadrotor by computing the discrete steady-state Kalman gain matrix for an LTI system. Using the LTI Kalman

gains significantly reduced the number of computations the embedded system had to perform in flight. To compute the gains, $Q = H$ and $R = N$ were used for each sensor axis. The Kalman filter improved the pitch and roll attitude measurements, as evident in Figure 2.8, even though the noise was assumed to be white. The pitch and roll rate estimates did not improve much, though this was because the gyros were more accurate and resistant to the vibrational modes that adversely affected the accelerometers. After the Kalman filter processed the sensor measurements and estimated the quadrotor attitude, the angles and rates were used to convert to the fixed body axis for control feedback.

2.4 Rotors

How the system behaved in response to feedback was determined in part by the rotors' properties. Rotor selection was based on prior rotor testing, which revealed rotor design performance relative to varied camber and solidity. The rotors operated with greater efficiency in a two-blade configuration than a four-blade one, and high thrust performance was achieved using blades with a fixed collective of 20° , 5% camber, 102 mm diameter, 13 mm chord, and thin plate airfoils [26]. Though designing for increased endurance was important for long-duration flights, the initial focus was on high thrust performance for guaranteeing successful hover as well as increased control authority. The rotors were fabricated using stereolithography from a CAD model. The blades consisted of VeroWhite rapid-prototyping material from Objet. Some blades of VeroBlue were tested, but they deformed at high

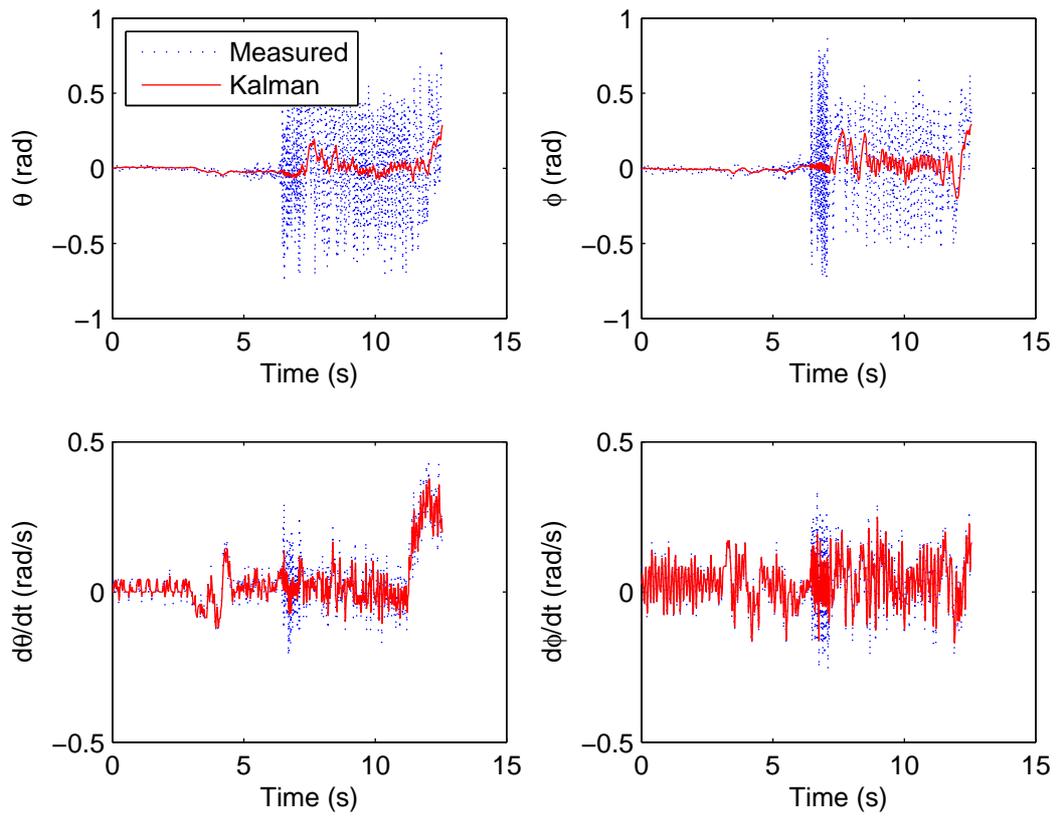


Figure 2.8: Recorded sensor data corrected using a Kalman filter simulated in MATLAB.

RPMs and degraded flight performance, drastically compromising the predictability required for computing control gains. VeroGray was more brittle, but having to produce large batches of brittle blades proved favorable compared to producing a few under-performing blades. The resulting rotors were tested using an automated experimental setup (Fig. 2.9). The rotor was mounted upside-down on one of the 18-11 2000 kv brushless outrunner motors used on the quadrotor. The inverted rotor thrust upwards, reducing wake interference from the test apparatus as well as eliminating erroneous thrust measurements from the wake impinging on the load cell. The motor itself was mounted on a cylindrical structure attached to a load and torque cell. The sensors were connected to a laptop through a National Instruments DAQ board, and the resulting data recorded using a LabView interface. The rotor RPM was measured with a laser tachometer that was modified to interface with the DAQ unit. The RPM itself was controlled by an Arduino microcontroller that took the motor through a range of set inputs. After testing, the data was processed in MATLAB using custom scripts to help identify the rotor parameters to judge performance as well as aid in controller design. The rotors performed as expected (Fig. 2.10), with aerodynamic properties displayed in Table 2.1.

Table 2.1: Rotor Properties

C_T	0.0264
C_Q	0.0061
$C_{P,ideal}$	0.0030
FM	0.5001

The results demonstrated the expected quadratic relationship between rotor RPM and thrust, as well as RPM and torque. Observing the location of the operating

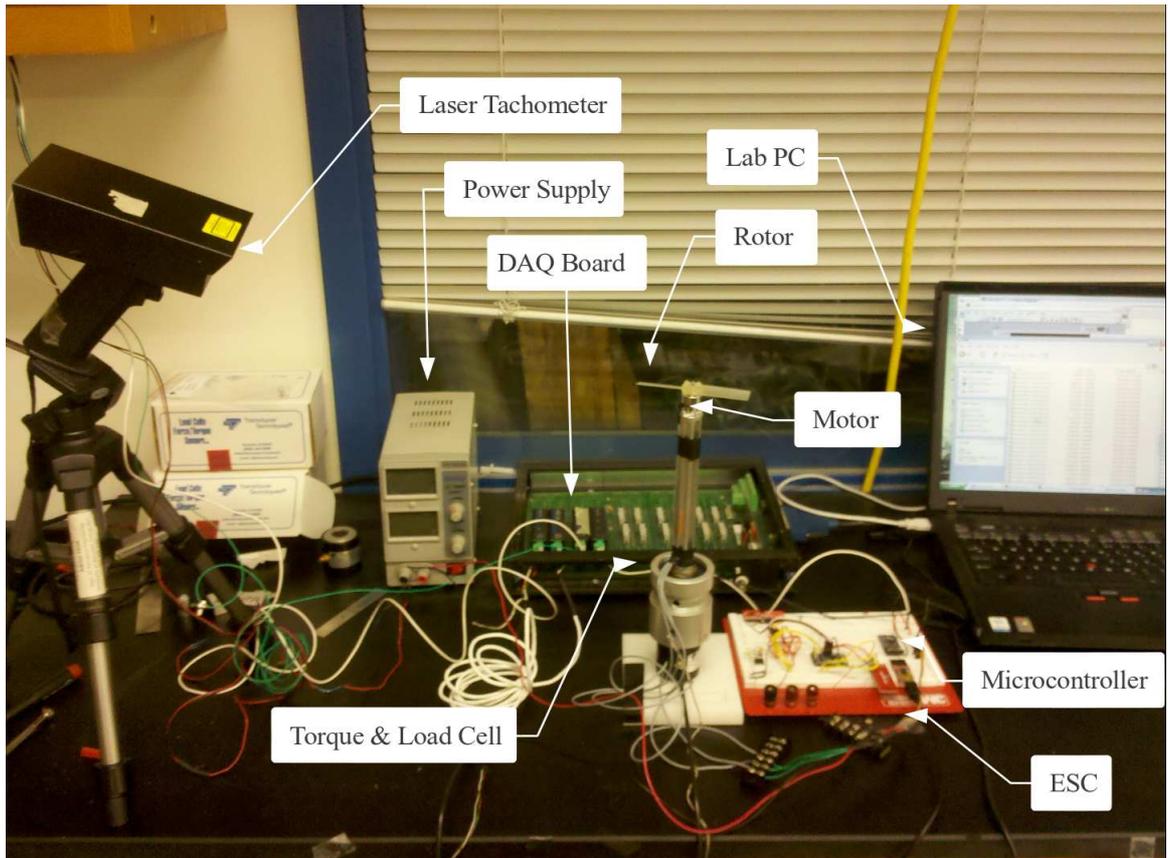
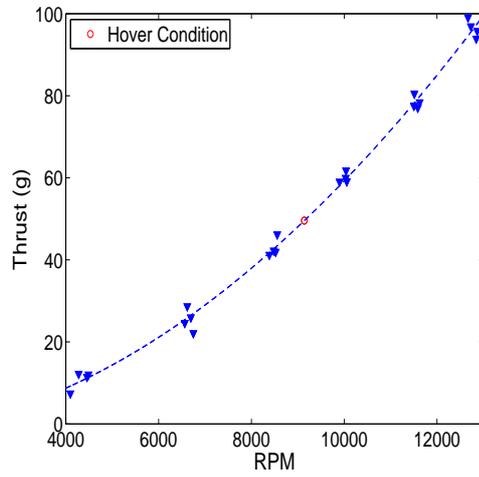
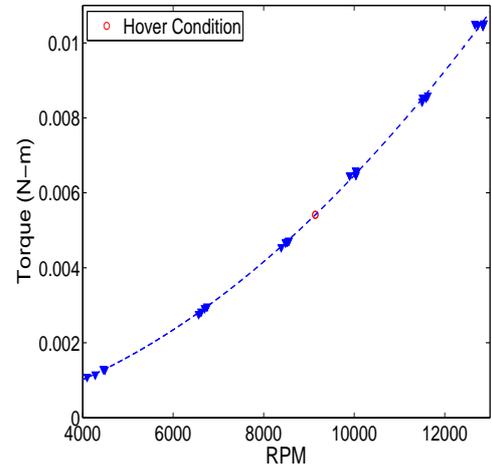


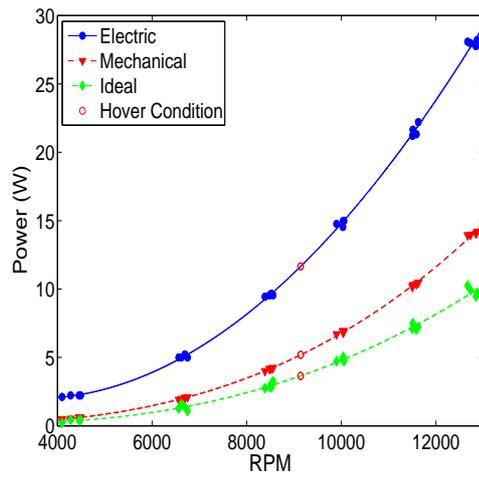
Figure 2.9: Test apparatus for rotor performance testing.



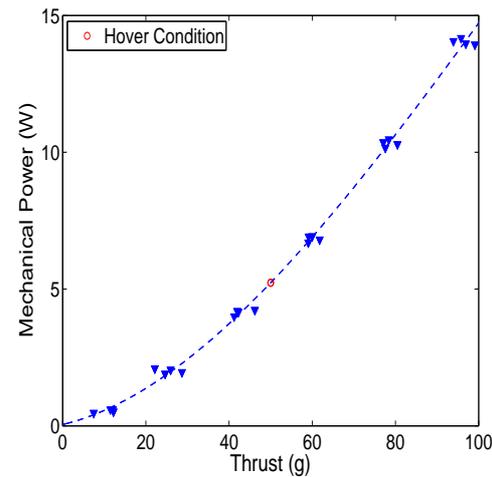
(a)



(b)



(c)



(d)

Figure 2.10: Rotor test data showing (a) thrust, (b) torque, and (c) power relative to the rotor's RPM, as well as (d) power versus thrust.

condition in Figure 2.10a, the excess thrust was readily identifiable. Looking at rotor power in Figure 2.10c, the discrepancy between mechanical and electric power became apparent. Approximately 12 W of electrical power at hover translated to only 5 W of mechanical power, signifying the losses due to electrical to mechanical conversion, as well as aerodynamic inefficiency at the scale. With a Figure of Merit of 0.5, rotor efficiency was relatively low, but expected [16]. However, the largest thrust recorded in the tests was more than enough for the quadrotor to sustain hover and carry a payload. The thrusts recorded were not the absolute maximum, however; the rotors were capable of reaching over 100 g (0.980 N) of lift, but the current draw of all four motors together was enough to damage the electronics board that connected them. Looking again at Figure 2.10c, the electric power used approaching maximum thrust was nearly 30 W for each motor. Even at the hover condition, the electronics needed some protection.

2.5 Speed Controllers

Speed controllers protected the on-board electronics and converted PWM control inputs into multiple out-of-phase voltage outputs that spun the motor bell and drove the rotor. Note that the motors could draw a current large enough to damage sensitive equipment, so a direct connection without any means of current protection could be catastrophic. The motor power supply, in this case a 11.1 V 680 mAh 3-cell Lithium-Polymer battery, connected directly to the ESCs. Since there were four motors, four ESCs were connected in parallel with one another to the battery. Though

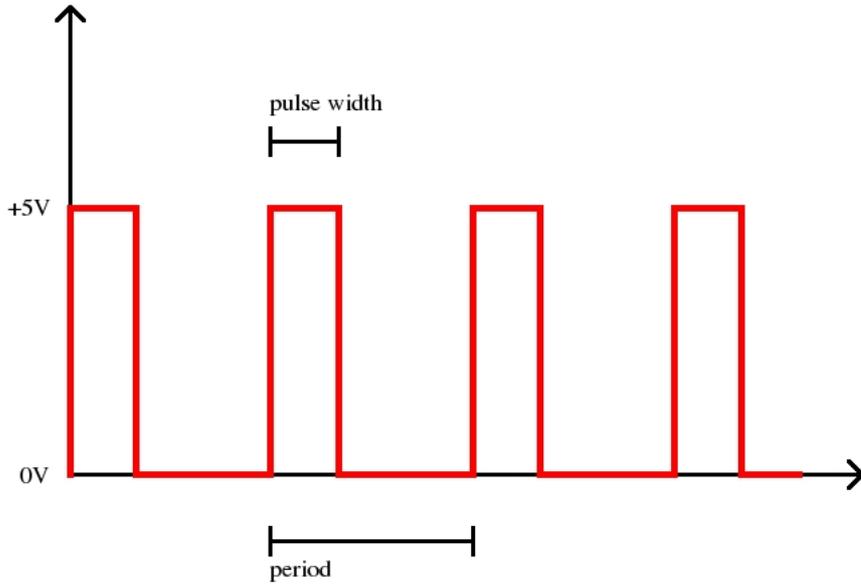


Figure 2.11: Example of a PWM signal assuming a 5-volt input.

an ESC did not directly affect the motor performance, it did affect how the motor dynamics were represented. The relationship between the pulse-width input and the motors rotational output depended entirely on the ESC. Consequently, dynamics of the motor had to be reconsidered whenever a speed controller was changed. A pulse-width was the duration of time that a digital signal was high, which was typically at 5V (Fig. 2.11). Most ESCs for RC aircraft operated on a period of around 2 milliseconds, and the range of operational pulse-widths typically varied depending on the ESC and the motor. For the Dualsky 6 A ESCs used on the quadrotor, the input could be between 1200 and 1700 μs . However, the relationship with thrust was only linear between 1200 and 1500 μs and began to saturate afterwards. These ranges could be altered by following instructions included with the speed controllers or by using a microcontroller. The process worked for Turnigy Plush 6A ESC's as well, so a set of Dualsky and Turnigy 6A ESC's could perform similarly given the

same redefined pulse-width ranges. Data from the rotor tests mentioned previously revealed the relationship between ESC input and rotor output, as seen in Figure 2.12, which was

$$T_i = k_T u_i + T_0 \quad (2.10)$$

$$Q_i = k_Q u_i + Q_0 \quad (2.11)$$

where $k_T = 3.38 \times 10^{-3} \text{ N}/\mu\text{s}$ and $k_Q = 3.72 \times 10^{-5} \text{ Nm}/\mu\text{s}$ were the slopes of the linear best-fit approximation of the thrust and torque data, respectively. These coefficients helped to characterize the relationship between the ESC and rotor performance. Larger values of k_T and k_Q denoted a larger sensitivity in thrust and torque for given control inputs compared to smaller coefficients. The values were affected not only by the rotor aerodynamic properties, but also by the ESC's input range. As mentioned previously, the speed controllers could be reprogrammed to accept different control input ranges. The speed controllers used on the quadrotor were at default ranges, but they could have their inputs remapped such that k_T and k_Q were increased. Changing the ESC performance in such a manner increased control authority, but at the cost of control resolution. A similar effect could be achieved by remapping control values digitally on the microcontroller. Either way, the control resolution had a linearly proportional effect on the rotor output. This relationship between speed controller input and rotor output was ideal for control application, but it assumed the rotors instantaneously reach these values, which was

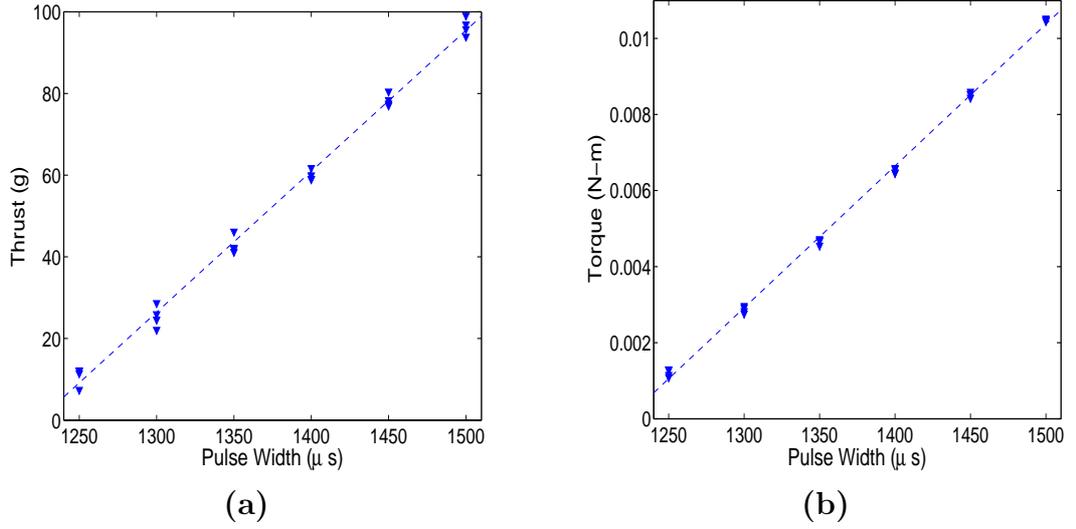


Figure 2.12: Rotor (a) thrust and (b) torque relative to the input pulse width.

not the case.

2.6 Rotor Dynamics

Neglecting the rotor dynamics relative to ESC input could have detrimental effects on control implementation. Any nonzero amount of time from control input to steady-state rotor output was essentially introducing a delay to the dynamical system and thus potentially causing instability when not accounted for. Though thrust and torque depended on the square of the rotor RPM, the rotor still reached a steady RPM without transience or overshoot, so it could be safely assumed that the system was first order for the sake of simplicity. Under this assumption, the system was expressed in the Laplace domain as:

$$W(s) = \frac{K_m U(s)}{(s + \alpha)} \quad (2.12)$$

where $W = \Omega^2$. The parameters were determined by analyzing test data taken around the operating input for the quadrotor in hover, $1368 \mu s$ pulse-width, which was found using Equation 2.10 and assuming the total thrust $\sum T$ was equal to the quadrotor's weight. Rotor testing was conducted using the same setup as previously shown in Figure 2.9, but only the RPM data was recorded. The motor received the hover input for 2 s, then got an additional step input that increased the pulse-width by a different increment for each test, and then returned the motor to the hover input. The first set of tests increased the input by $20 \mu s$, and the second test set increased the input by $30 \mu s$ (Fig. 2.13). The variation of motor input helped confirm that the rotor dynamics computed from the data remained constant regardless of input. To find the dynamical parameters, the motor pole was determined by identifying the rise-time of the system and using the relation $t_r = 4/\alpha$ [24]. Because the motor was given a step input, the Final Value Theorem was used to compute K_m :

$$\begin{aligned}
\lim_{s \rightarrow 0} sW(s) &= W_{ss} \\
&= \lim_{s \rightarrow 0} s \frac{K_m U(s)}{(s + \alpha)} \\
&= \lim_{s \rightarrow 0} s \frac{K_m \Delta U}{s(s + \alpha)} \\
&= \lim_{s \rightarrow 0} \frac{K_m \Delta U}{(s + \alpha)} \\
&= \frac{K_m \Delta U}{\alpha}
\end{aligned} \tag{2.13}$$

where ΔU was the step input given relative the initial input, and W_{ss} was the

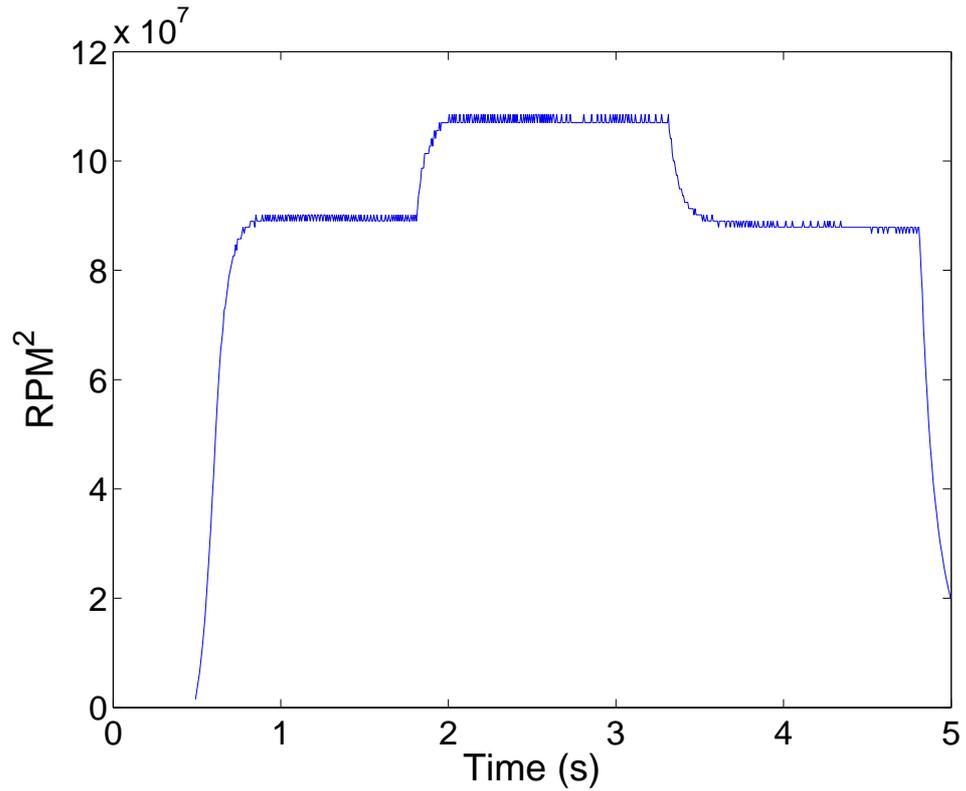


Figure 2.13: Example of rotor dynamics test case. In this case, the hover input is increased by a step of $30 \mu s$.

squared-RPM achieved by the motor.

The rotor dynamics parameters were determined for each test and compared as shown in Figure 2.14. The tests were then averaged to compute the mean rotor dynamics. A simulation of all tests along with the mean dynamics was recorded in Figure 2.15. The mean dynamics fit neatly between the tested values and well within the scatter of resultant outputs. Thus, the values $K_m = 1.195 \times 10^5$ and $\alpha = 18.5$ could be used in dynamical simulation and for control algorithm formulation. These values depended on the physical properties of the rotor and the electrical properties of the motor; the rotor inertia and motor torque were the dominant effects that determined α , however K_m was also affected by ESC implementation.

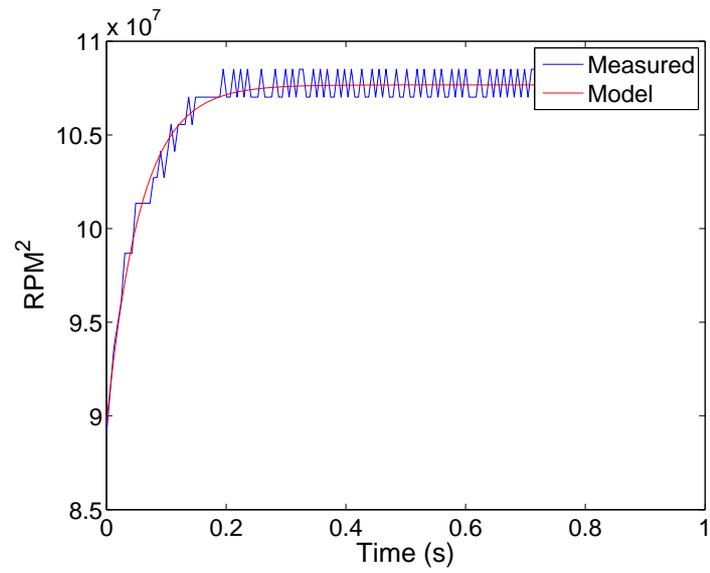


Figure 2.14: When compared to test data, the first-order model accurately approximates the rotor dynamics.

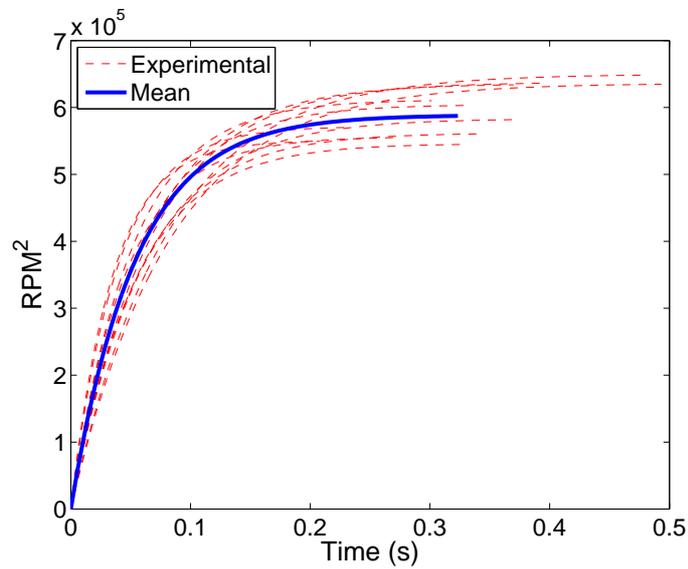


Figure 2.15: Rotor models generated from different test runs were averaged to approximate the rotor dynamics.

For completion and future reference, the time-domain dynamics for the rotor were obtained by performing an inverse Laplace-transform on the transfer function, giving

$$\dot{W}(t) = -\alpha W(t) + K_m u(t) \quad (2.14)$$

as the motor dynamics relative to a pulse-width input $u(t)$. With the properties of the rotors and the quadrotor found, the control laws governing the vehicle's stability were formulated.

Chapter 3

Control

The quadrotor’s mechanical simplicity comes at a cost to stability. Looking at a formulation of the quadrotor’s attitude dynamics makes the inherent instability clear, while also providing the necessary groundwork for designing a model-based controller to augment the vehicle’s stability. Though a PID controller proves viable for quadrotor operation [7], a MIMO MBC accounts for control couplings and also reduces the complexity of gain tuning. Specifically, the quadrotor employs the LQR algorithm for feedback control. Before either controller can be implemented, a dynamical model of the quadrotor must be formulated.

3.1 Quadrotor Rotational Dynamics

The dynamics are formulated by considering the body and rotor gyroscopic effects, as well as by balancing the torques and moments generated by the motors in the inertial frame (Fig. 3.1). Since the goal of the project is to control a quadrotor in hover, the vehicle’s lateral translations are considered small and slow enough so as to neglect the body’s aerodynamic effects. Thus, the rotational kinematics of the quadrotor are derived from the Newtonian relationship:

$$I\dot{\omega} = -\sum \omega_p \times L + \sum \tau \tag{3.1}$$

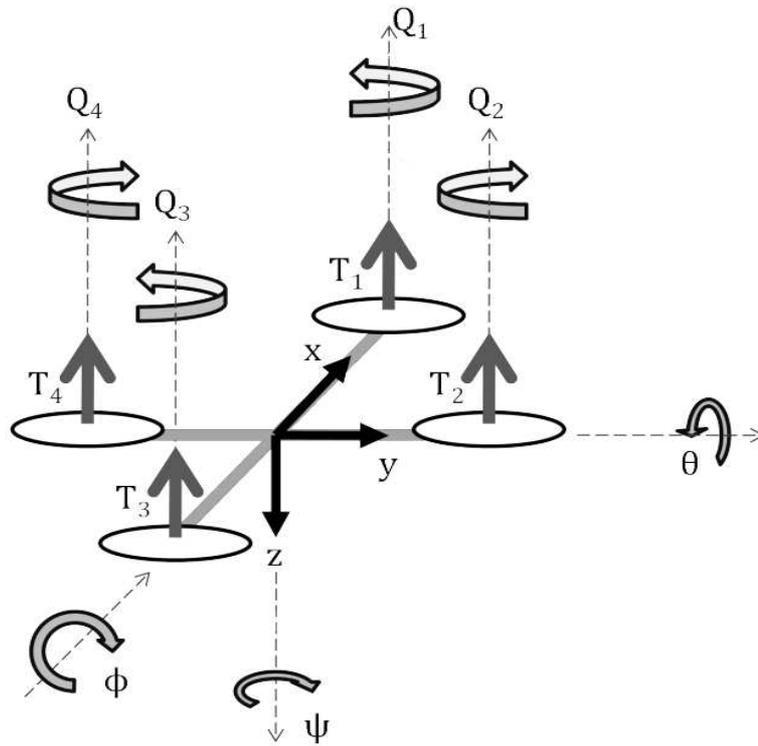


Figure 3.1: Coordinate frame for common symmetric quadrotor configuration.

where $\dot{\omega}$ is the angular acceleration about an axis, ω_p is the rate of precession about an orthogonal axis, L is the angular momentum of an axis orthogonal to both other rates of rotation, and τ is any miscellaneous moment or torque imparted on the system. In other words, the rotational momentum of the system about an axis is equal to the summation of net gyroscopic effects and any additional torques or moments affecting rotation on this axis. For example, if the quadrotor rotates about the z-axis at a rate $\dot{\psi}$, then

$$L_z = I_z \dot{\psi} \quad (3.2)$$

and if the vehicle body precesses about the y-axis, then there is a torque about the x-axis due to the gyroscopic coupling $\tau_{b,x}$ from body rotation,

$$\tau_{b,x} = \dot{\theta} \times I_z \dot{\psi} \quad (3.3)$$

The torque $\tau_{b,x}$ can also be generated by precession in the z-axis if the quad-rotor is rotating about the y-axis:

$$\tau_{b,x} = -\dot{\psi} \times I_y \dot{\theta} \quad (3.4)$$

The two equations can then be summed to give the torque about the x-axis generated by gyroscopic coupling:

$$\tau_{b,x} = \dot{\theta} \dot{\psi} (I_z - I_y)$$

This method can be applied to each axis of rotation as well to find the resultant torques due to body rotation for the other axes as well:

$$\begin{aligned}
\tau_{b,x} &= \dot{\theta}\dot{\psi}(I_z - I_y) \\
\tau_{b,y} &= \dot{\phi}\dot{\psi}(I_x - I_z) \\
\tau_{b,z} &= \dot{\phi}\dot{\theta}(I_y - I_x)
\end{aligned} \tag{3.5}$$

As the quad-rotor rotates about an axis, it is changing the direction of a rotating rotor. This only applies to rotation about the x and y axes, since the rotors are always rotating about the body z-axis. The angular momentum of the rotor is

$$L_r = J\Omega \tag{3.6}$$

so the rotational torques for each axis are thus:

$$\begin{aligned}
\tau_{r,x} &= \dot{\theta}\Sigma\Omega J \\
\tau_{r,y} &= -\dot{\phi}\Sigma\Omega J
\end{aligned} \tag{3.7}$$

where Ω is the net rotational velocity of the rotors, i.e.,

$$\Sigma\Omega = -\omega_1 + \omega_2 - \omega_3 + \omega_4 \tag{3.8}$$

The summation of equations 3.5 and 3.7 gives the total torque due to gyroscopic motion about each axis:

$$\begin{aligned}
\tau_{g,x} &= \dot{\theta}\dot{\psi}(I_z - I_y) + \dot{\theta}\Sigma\Omega J \\
\tau_{g,y} &= \dot{\phi}\dot{\psi}(I_x - I_z) - \dot{\phi}\Sigma\Omega J \\
\tau_{g,z} &= \dot{\phi}\dot{\theta}(I_y - I_x)
\end{aligned} \tag{3.9}$$

Since the motors are mounted at the end of each arm, there are additional torques acting on the quad-rotor. Assuming that the motors are perfectly aligned, each i -th motor generates a thrust vector T_i in the $-z$ direction. Also assuming symmetry, all motors are a distance l from the center of gravity of the quad-rotor. As such, there are four moments in action at all times, but only specific pairs of motors will generate motion about each axis. For example, the quad-rotor will roll about the x -axis if motors 2 and 4 (fore and aft) have different thrusts:

$$\tau_x = l(T_4 - T_2)$$

The quad-rotor will pitch about the y -axis in a similar fashion if motors 1 and 3 are unbalanced. To yaw about the z -axis, however, involves unbalancing the net torque of motor-pairs. Since motors 1 and 3 are rotating counter-clockwise, and motors 2 and 4 clockwise, then having the former pair operate at a net torque higher than the latter will result in positive yaw rotation by conservation of angular momentum.

Therefore,

$$\tau_z = (Q_1 + Q_3) - (Q_2 + Q_4)$$

where Q_i is the torque of the i -th rotor. The net torques due directly to motor control are then

$$\begin{aligned}\tau_{m,x} &= l(T_4 - T_2) \\ \tau_{m,y} &= l(T_1 - T_3) \\ \tau_{m,z} &= (Q_1 - Q_2 + Q_3 - Q_4)\end{aligned}\tag{3.10}$$

The rotational kinematics for the system, then, is the substitution of equations 3.9 and 3.10 into 3.1, giving:

$$\begin{aligned}I_x\ddot{\phi} &= \dot{\theta}\dot{\psi}(I_y - I_z) + \dot{\theta}\Sigma\Omega J + l(T_4 - T_2) \\ I_y\ddot{\theta} &= \dot{\phi}\dot{\psi}(I_z - I_x) - \dot{\phi}\Sigma\Omega J + l(T_1 - T_3) \\ I_z\ddot{\psi} &= \dot{\phi}\dot{\theta}(I_x - I_y) + (Q_1 - Q_2 + Q_3 - Q_4)\end{aligned}\tag{3.11}$$

Examining the above equations reveals some characteristics of the system. For example, the rotational kinematics are nonlinear and highly coupled. If a quadrotor begins to pitch about the y-axis, it will induce both roll and yaw rotation as well. Unwanted rotations also result from imbalances in the rotor torques. Since the rotors are arranged as counter-rotating pairs, there is an expectation of negligible net torque when the operational RPMs are equal. Motor control inputs must be

kept equal and opposite among the pairs to preserve this equilibrium. For example, if motor 2 needs to increase its thrust, then motor 4 must decrease its rotational velocity by the same amount. As a result, controlling the quadrotors attitude involves changing the RPM of rotor pairs rather than individually (Fig. 1.1). Roll can be achieved by changing the thrust of motor 4 and the thrust of motor 2 in an equal and opposite increment; pitch control operates similarly using motors 1 and 3. The quadrotor changes its heading by offsetting the torque balance between motor pairs, taking advantage of principles of angular momentum; since motors 1 and 3 are rotating counter-clockwise, and motors 2 and 4 clockwise, then the quadrotor itself rotates clockwise if the 1-3 motor pair has a higher net torque than the 2-4 pair. Some other configurations make use of tilting rotors or motors mounted at angles in order to provide heading control and balance, but at small scales, implementation is complex, thereby losing the design simplicity gained via the quadrotor configuration. Since the quadrotor relies on adjusting thrust and torque for control, the dynamics depend on rotor and motor performance. Knowing that

$$\begin{aligned}
 T &= C_T \rho A R^2 \Omega^2 \\
 Q &= C_Q \rho A R^3 \Omega^2
 \end{aligned}
 \tag{3.12}$$

the quadrotor dynamics described by equations 3.11 can be modified thusly:

$$\begin{aligned}
I_x \ddot{\phi} &= \dot{\theta} \dot{\psi} (I_y - I_z) + \dot{\theta} \Sigma \Omega J + l(C_T \rho A R^2)(\Omega_4^2 - \Omega_2^2) \\
I_y \ddot{\theta} &= \dot{\phi} \dot{\psi} (I_z - I_x) - \dot{\phi} \Sigma \Omega J + l(C_T \rho A R^2)(\Omega_1^2 - \Omega_3^2) \\
I_z \ddot{\psi} &= \dot{\phi} \dot{\theta} (I_x - I_y) + (C_Q \rho A R^3)(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \quad (3.13)
\end{aligned}$$

The above equations reveal that the quadrotor's control authority is proportional to the rotors' aerodynamic properties. Higher rates of motion are achievable by increasing either the rotor disk area or the rotor coefficients, which would help with quadrotors that have high inertial properties. Since the changes in RPM are not instantaneous, the more significant factors in determining the quadrotor's attitude kinematics are the motor dynamics. Recall that $\Omega^2 = W$, then substitute equation 2.14 into 3.13, and assume that all motors and rotors have identical properties:

$$\begin{aligned}
I_x \ddot{\phi}(t) &= \dot{\theta}(t) \dot{\psi}(t) (I_y - I_z) + \dot{\theta}(t) \Sigma \Omega(t) J + l(C_T \rho A R^2)(W_4(t) - W_2(t)) \\
I_y \ddot{\theta}(t) &= \dot{\phi}(t) \dot{\psi}(t) (I_z - I_x) - \dot{\phi}(t) \Sigma \Omega(t) J + l(C_T \rho A R^2)(W_1(t) - W_3(t)) \\
I_z \ddot{\psi}(t) &= \dot{\phi}(t) \dot{\theta}(t) (I_x - I_y) + (C_Q \rho A R^3)(W_1(t) - W_2(t) + W_3(t) - W_4(t)) \\
\dot{W}_1(t) &= -\alpha W_1(t) + K_m u_1(t) \\
\dot{W}_2(t) &= -\alpha W_2(t) + K_m u_2(t) \\
\dot{W}_3(t) &= -\alpha W_3(t) + K_m u_3(t) \\
\dot{W}_4(t) &= -\alpha W_4(t) + K_m u_4(t) \quad (3.14)
\end{aligned}$$

With the rotor states included, most of the significant factors in determining the quadrotor's dynamics in hover are accounted for. Aside from the rotors, most of the equations derived are nonlinear, and so must be linearized.

3.2 System Linearization

As noted previously, the quadrotor's dynamics are nonlinear. In order to use LQR, the system must be linearized. Linearization can be performed by considering small perturbations about a reference condition. Assume that the states of the actual system dynamics can be denoted as $\underline{z}(t)$, where

$$\begin{aligned}\underline{z}(t) &= [\phi(t) \dot{\phi}(t) \theta(t) \dot{\theta}(t) \psi(t) \dot{\psi}(t) W_1(t) W_2(t) W_3(t) W_4(t)]^T \\ &= [z_1(t) z_2(t) z_3(t) z_4(t) z_5(t) z_6(t) z_7(t) z_8(t) z_9(t) z_{10}(t)]^T\end{aligned}\quad (3.15)$$

and the inputs of the system as:

$$\underline{\mu}(t) = [\mu_1(t) \mu_2(t) \mu_3(t) \mu_4(t)]^T\quad (3.16)$$

Thus the dynamical equations 3.14 become

$$\begin{aligned}
\dot{z}_2(t) &= z_4(t)z_6(t)\left(\frac{I_y - I_z}{I_x}\right) - z_4(t)\Sigma\Omega(t)\frac{J}{I_x} + \frac{lC_T\rho AR^2}{I_x}(z_{10}(t) - z_8(t)) \\
\dot{z}_4(t) &= z_2(t)z_6(t)\left(\frac{I_z - I_x}{I_y}\right) + z_2(t)\Sigma\Omega(t)\frac{J}{I_y} + \frac{lC_T\rho AR^2}{I_y}(z_7(t) - z_9(t)) \\
\dot{z}_6(t) &= z_2(t)z_4(t)\left(\frac{I_x - I_y}{I_z}\right) + \frac{C_Q\rho AR^3}{I_z}[(z_7(t) + z_9(t)) - (z_8(t) + z_{10}(t))] \\
\dot{z}_7(t) &= -\alpha z_7(t) + K_m\mu_1(t) \\
\dot{z}_8(t) &= -\alpha z_8(t) + K_m\mu_2(t) \\
\dot{z}_9(t) &= -\alpha z_9(t) + K_m\mu_3(t) \\
\dot{z}_{10}(t) &= -\alpha z_{10}(t) + K_m\mu_4(t)
\end{aligned} \tag{3.17}$$

with the additional stipulation

$$\begin{aligned}
z_2(t) &= \dot{z}_1(t) \\
z_4(t) &= \dot{z}_3(t) \\
z_6(t) &= \dot{z}_5(t)
\end{aligned} \tag{3.18}$$

The equations will be linearized about a reference condition with states $\underline{z}^*(t)$ and inputs $\underline{\mu}^*(t)$. Assuming that the perturbations of the nonlinear states away from the reference condition are minimal such that

$$\underline{0} = \underline{f}(\underline{\dot{z}}, \underline{z}, \underline{\mu}, t) - \underline{f}(\underline{\dot{z}}^*, \underline{z}^*, \underline{\mu}^*, t) \tag{3.19}$$

the system can be linearized with a multi-variable Taylor series expansion of the above expression [27]:

$$\begin{aligned} \underline{0} = & \underline{f}(\underline{\dot{z}}^*, \underline{z}^*, \underline{\mu}^*, t) + \left. \frac{\partial \underline{f}}{\partial \underline{\dot{z}}} \right|_{RC} (\underline{\dot{z}}(t) - \underline{\dot{z}}^*(t)) + \left. \frac{\partial \underline{f}}{\partial \underline{z}} \right|_{RC} (\underline{z}(t) - \underline{z}^*(t)) \\ & + \left. \frac{\partial \underline{f}}{\partial \underline{\mu}} \right|_{RC} (\underline{\mu}(t) - \underline{\mu}^*(t)) + \underline{r}(\underline{z}(t), \underline{\mu}(t)) \end{aligned} \quad (3.20)$$

where $\underline{f}(\underline{\dot{z}}, \underline{z}, \underline{\mu}, t)$ is the nonlinear system dynamics as a function of the states, inputs, and time t , or

$$\underline{f}(\underline{\dot{z}}, \underline{z}, \underline{\mu}, t) =$$

$$\left[\begin{array}{l} \dot{z}_1(t) - z_2(t) \\ z_4(t)z_6(t)\left(\frac{I_y - I_z}{I_x}\right) - z_4(t)\Sigma\Omega(t)\frac{J}{I_x} + \frac{lC_T\rho AR^2}{I_x}(z_{10}(t) - z_8(t)) - \dot{z}_2(t) \\ \dot{z}_3(t) - z_4(t) \\ z_2(t)z_6(t)\left(\frac{I_z - I_x}{I_y}\right) + z_2(t)\Sigma\Omega(t)\frac{J}{I_y} + \frac{lC_T\rho AR^2}{I_y}(z_7(t) - z_9(t)) - \dot{z}_4(t) \\ \dot{z}_5 - z_6 \\ z_2(t)z_4(t)\left(\frac{I_x - I_y}{I_z}\right) + \frac{C_Q\rho AR^3}{I_z}[(z_7(t) + z_9(t)) - (z_8(t) + z_{10}(t))] - \dot{z}_6(t) \\ - \alpha z_7(t) + K_m\mu_1(t) \\ - \alpha z_8(t) + K_m\mu_2(t) \\ - \alpha z_9(t) + K_m\mu_3(t) \\ - \alpha z_{10}(t) + K_m\mu_4(t) \end{array} \right] \quad (3.21)$$

Since linearization is for the hover condition, then $\underline{f}(\underline{\dot{z}}^*, \underline{z}^*, \underline{\mu}^*, t) = 0$ and $\underline{r}(\underline{z}, \underline{\mu}) = 0$.

Then, compute the partial derivatives:

$$\begin{aligned}
 \frac{\partial \underline{f}}{\partial \underline{\dot{z}}} &= \begin{bmatrix} \frac{\partial f_1}{\partial \dot{z}_1} & \frac{\partial f_1}{\partial \dot{z}_2} & \cdots & \frac{\partial f_1}{\partial \dot{z}_{10}} \\ \frac{\partial f_2}{\partial \dot{z}_1} & \frac{\partial f_2}{\partial \dot{z}_2} & \cdots & \frac{\partial f_2}{\partial \dot{z}_{10}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_{10}}{\partial \dot{z}_1} & \cdots & \frac{\partial f_{10}}{\partial \dot{z}_9} & \frac{\partial f_{10}}{\partial \dot{z}_{10}} \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{3.22}
 \end{aligned}$$

$$\begin{aligned}
\frac{\partial \underline{f}}{\partial \underline{z}} &= \begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \frac{\partial f_1}{\partial z_2} & \dots & \frac{\partial f_1}{\partial z_{10}} \\ \frac{\partial f_2}{\partial z_1} & \frac{\partial f_2}{\partial z_2} & \dots & \frac{\partial f_2}{\partial z_{10}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_{10}}{\partial z_1} & \dots & \frac{\partial f_{10}}{\partial z_9} & \frac{\partial f_{10}}{\partial z_{10}} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix}
\end{aligned} \tag{3.23}$$

where

$$\begin{aligned}
\mathbf{M}_{11} &= \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & z_6(t)\left(\frac{I_y - I_z}{I_x}\right) - \Sigma\Omega(t)\frac{J}{I_x} & 0 & z_4(t)\left(\frac{I_y - I_z}{I_x}\right) \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & z_6(t)\left(\frac{I_z - I_x}{I_y}\right) + \Sigma\Omega(t)\frac{J}{I_y} & 0 & 0 & 0 & z_2(t)\left(\frac{I_z - I_x}{I_y}\right) \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & z_4(t)\left(\frac{I_x - I_y}{I_z}\right) & 0 & z_4(t)\left(\frac{I_x - I_y}{I_z}\right) & 0 & 0 \end{bmatrix} \\
\mathbf{M}_{12} &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -\frac{lC_T\rho AR^2}{I_x} & 0 & \frac{lC_T\rho AR^2}{I_x} \\ 0 & 0 & 0 & 0 \\ \frac{lC_T\rho AR^2}{I_y} & 0 & -\frac{lC_T\rho AR^2}{I_x} & 0 \\ 0 & 0 & 0 & 0 \\ \frac{C_Q\rho AR^3}{I_z} & -\frac{C_Q\rho AR^3}{I_z} & \frac{C_Q\rho AR^3}{I_z} & -\frac{C_Q\rho AR^3}{I_z} \end{bmatrix} \\
\mathbf{M}_{21} &= \begin{bmatrix} \mathbf{0}_{4 \times 6} \end{bmatrix} \\
\mathbf{M}_{22} &= \begin{bmatrix} -\alpha & 0 & 0 & 0 \\ 0 & -\alpha & 0 & 0 \\ 0 & 0 & -\alpha & 0 \\ 0 & 0 & 0 & -\alpha \end{bmatrix}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial f}{\partial \underline{\mu}} &= \begin{bmatrix} \frac{\partial f_1}{\partial \mu_1} & \frac{\partial f_1}{\partial \mu_2} & \dots & \frac{\partial f_1}{\partial \mu_4} \\ \frac{\partial f_2}{\partial \mu_1} & \frac{\partial f_2}{\partial \mu_2} & \dots & \frac{\partial f_2}{\partial \mu_4} \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_{10}}{\partial \mu_1} & \dots & \frac{\partial f_{10}}{\partial \mu_3} & \frac{\partial f_{10}}{\partial \mu_4} \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ K_m & 0 & 0 & 0 \\ 0 & K_m & 0 & 0 \\ 0 & 0 & K_m & 0 \\ 0 & 0 & 0 & K_m \end{bmatrix}
\end{aligned} \tag{3.24}$$

To linearize about hover, evaluate the above matrices at the reference condition

$$\underline{z}(t) = \underline{z}^*(t) = \underline{0} \text{ and } \underline{\mu}(t) = \underline{\mu}^*(t) = \underline{0} \forall t:$$

$$\frac{\partial f}{\partial \underline{z}} \Big|_{RC} = \mathbf{E} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \quad (3.25)$$

$$\begin{aligned}
& \frac{\partial f}{\partial \underline{z}} \Big|_{RC} = \\
\mathbf{F} &= \begin{bmatrix}
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{lC_T\rho AR^2}{I_x} & 0 & \frac{lC_T\rho AR^2}{I_x} \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{lC_T\rho AR^2}{I_y} & 0 & -\frac{lC_T\rho AR^2}{I_x} & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{C_Q\rho AR^3}{I_z} & -\frac{C_Q\rho AR^3}{I_z} & \frac{C_Q\rho AR^3}{I_z} & -\frac{C_Q\rho AR^3}{I_z} \\
0 & 0 & 0 & 0 & 0 & 0 & -\alpha & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha
\end{bmatrix} \\
& \hspace{20em} (3.26)
\end{aligned}$$

$$\begin{aligned}
\left. \frac{\partial \underline{f}}{\partial \underline{z}} \right|_{RC} &= \\
\mathbf{G} &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ K_m & 0 & 0 & 0 \\ 0 & K_m & 0 & 0 \\ 0 & 0 & K_m & 0 \\ 0 & 0 & 0 & K_m \end{bmatrix} \quad (3.27)
\end{aligned}$$

As is already apparent, a large amount of information is lost in the linearization. Bouabdullah warns of performing the process with hover as a reference condition [7]. Though the resulting feedback system may not be suitable for a large flight envelope, for the purposes of stabilized hover, they appear adequate. To finalize and simplify the linearization process, revise the Taylor expansion from equation 3.20:

$$\underline{0} = \mathbf{E}(\underline{\dot{z}}(t) - \underline{\dot{z}}^*(t)) + \mathbf{F}(\underline{z}(t) - \underline{z}^*(t)) + \mathbf{G}(\underline{\mu}(t) - \underline{\mu}^*(t)) \quad (3.28)$$

Considering the perturbed states as

$$\underline{x}(t) = (\underline{z}(t) - \underline{z}^*(t)) \quad (3.29)$$

and the inputs as

$$\underline{u}(t) = (\underline{\mu}(t) - \underline{\mu}^*(t)), \quad (3.30)$$

the expansion can be rearranged to resemble the familiar form

$$\begin{aligned} \dot{\underline{x}}(t) &= -\mathbf{E}^{-1}\mathbf{F}\underline{x}(t) - \mathbf{E}^{-1}\mathbf{G}\underline{u}(t) \\ &= \mathbf{A}\underline{x}(t) + \mathbf{B}\underline{u}(t) \end{aligned} \quad (3.31)$$

Thus, the linearized state-space form of the quad-rotor dynamics about hover is as follows:

$$\begin{aligned}
 \dot{\underline{x}}(t) = & \begin{bmatrix}
 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{lC_T\rho AR^2}{I_x} & 0 & \frac{l(C_T\rho AR^2)}{I_x} \\
 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & \frac{lC_T\rho AR^2}{I_y} & 0 & \frac{l(C_T\rho AR^2)}{I_y} & 0 \\
 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & \frac{(C_Q\rho AR^3)}{I_z} & -\frac{(C_Q\rho AR^3)}{I_z} & \frac{(C_Q\rho AR^3)}{I_z} & -\frac{(C_Q\rho AR^3)}{I_z} \\
 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha
 \end{bmatrix} \underline{x}(t) \\
 + & \begin{bmatrix}
 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 \\
 K_m & 0 & 0 & 0 \\
 0 & K_m & 0 & 0 \\
 0 & 0 & K_m & 0 \\
 0 & 0 & 0 & K_m
 \end{bmatrix} \underline{u}(t)
 \end{aligned} \tag{3.32}$$

Some of the eigenvalues of the state matrix \mathbf{A} of the quadrotor dynamics are zero, specifically those dealing with attitude kinematics, which means that the attitude in hover is unstable [28]. Any disturbances can send the quadrotor away from level hover without any correction. The quadrotor in hover behaves similar to a ball sitting at the top of a hill, in which the ball only maintains equilibrium so long as it is undisturbed. For the quadrotor to achieve stability, a feedback control system must be implemented.

3.3 LQR

Before formulating an LQR controller for feedback control, several criteria must be met. Specifically, the system must be at least stabilizable and detectable. Since the state matrix \mathbf{A} has zero eigenvalues and the quad-rotor's heading is not measured because of interference from the motors, these qualifications must be verified. First, recall that a stipulation of controllability is

$$\text{rank}[\mathbf{B}|\mathbf{A}\mathbf{B}|\mathbf{A}^2\mathbf{B}|\dots|\mathbf{A}^{n-1}\mathbf{B}] = n \quad (3.33)$$

where $\mathbf{A} \in \mathbb{R}^{10 \times 10}$ and $\mathbf{B} \in \mathbb{R}^{10 \times 4}$, so $n = 10$ [1]. Using the matrices from equation 3.32, equation 3.33 is satisfied. As for observability, the observation matrix of the

system will be $\mathbf{C} \in \mathbb{R}^{9 \times 10}$, since heading angle is neglected,

$$\begin{aligned} \underline{y}(t) &= \mathbf{C}\underline{x}(t) \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \underline{x}(t) \end{aligned} \quad (3.34)$$

and is used in the equation

$$\text{rank} \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix} = m \quad (3.35)$$

in which $m = 9$. The above equation is also satisfied, so the system is observable as well. Considering that the system is observable despite lacking heading angle measurements, and also considering that the yaw dynamics of the system in equation

3.11 has no direct dependence on the yaw angle itself, one more modification of the dynamics can take place, simplifying the system further. Removing the yaw angle from the system in the state-space representation of the dynamics gives:

$$\begin{aligned}
 \dot{\underline{x}}(t) = & \begin{bmatrix}
 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & -\frac{lC_T\rho AR^2}{I_x} & 0 & \frac{l(C_T\rho AR^2)}{I_x} \\
 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & \frac{lC_T\rho AR^2}{I_y} & 0 & \frac{l(C_T\rho AR^2)}{I_y} & 0 \\
 0 & 0 & 0 & 0 & 0 & \frac{(C_Q\rho AR^3)}{I_z} & -\frac{(C_Q\rho AR^3)}{I_z} & \frac{(C_Q\rho AR^3)}{I_z} & -\frac{(C_Q\rho AR^3)}{I_z} \\
 0 & 0 & 0 & 0 & 0 & -\alpha & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha
 \end{bmatrix} \underline{x}(t) \\
 + & \begin{bmatrix}
 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 \\
 K_m & 0 & 0 & 0 \\
 0 & K_m & 0 & 0 \\
 0 & 0 & K_m & 0 \\
 0 & 0 & 0 & K_m
 \end{bmatrix} \underline{u}(t)
 \end{aligned} \tag{3.36}$$

which still satisfies equations 3.33 and 3.35. With the system linearized, the linear quadratic control problem can be solved by defining the control input for the above system as

$$\underline{u}(t) = -\mathbf{K}\underline{x}(t) \quad (3.37)$$

where \mathbf{K} is the solution of the Algebraic Ricatti Equation (ARE) for a linear time-invariant system. The solution can be tuned by manipulating the state and control cost matrices \mathbf{Q} and \mathbf{R} , respectively. Since their dimensions still reflect those of the system, a scaling procedure known as Brysons method quickly establishes the basic values for the cost matrices and allows them to be modified en masse with scalars. Using Brysons method, assume that \mathbf{Q} is a diagonal positive-definite matrix where each entry is an equilibrium point of the system:

$$\mathbf{Q} = q \begin{bmatrix} \left(\frac{1}{z_1^*}\right)^2 & 0 & \cdots & 0 \\ 0 & \left(\frac{1}{z_2^*}\right)^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \left(\frac{1}{z_i^*}\right)^2 \end{bmatrix} \quad (3.38)$$

If the reference conditions for the system are zero, then consider the reference conditions as the allowable accuracy of the states. The gain matrix \mathbf{K} resulting from the solution to the ARE can then be tuned by adjusting q , thus increasing or decreasing the penalty on the system relative to state accuracy [27]. Likewise, the control cost

matrix \mathbf{R} is populated with the control limits, so:

$$\mathbf{R} = r \begin{bmatrix} (\frac{1}{\mu_1^*})^2 & 0 & \cdots & 0 \\ 0 & (\frac{1}{\mu_2^*})^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & (\frac{1}{\mu_j^*})^2 \end{bmatrix} \quad (3.39)$$

where μ_j is the j -th control limit, and r is an arbitrary scalar that can be adjusted to affect control cost. Bryson's method effectively scales the LQR problem by normalizing the system states and controls, and thus computing \mathbf{K} using the ARE [28]. Keeping this in mind, assume the attitudes are nominally within 0.005 rad and their respective rates within 0.05 rad/s:

$$\begin{aligned} |\phi| &\leq 0.005 \text{ rad} \\ |\dot{\phi}| &\leq 0.05 \text{ rad/s} \\ |\theta| &\leq 0.005 \text{ rad} \\ |\dot{\theta}| &\leq 0.05 \text{ rad/s} \\ |\dot{\psi}| &\leq 0.05 \text{ rad/s} \end{aligned} \quad (3.40)$$

Likewise, assume that the squares of the rotor velocities are within 10000 radians²/second²:

$$\begin{aligned}
W_1 &\leq 10000 \text{ rad}^2/\text{s}^2 \\
W_2 &\leq 10000 \text{ rad}^2/\text{s}^2 \\
W_3 &\leq 10000 \text{ rad}^2/\text{s}^2 \\
W_4 &\leq 10000 \text{ rad}^2/\text{s}^2
\end{aligned}
\tag{3.41}$$

As seen in Section 2.5, the control inputs themselves are limited to a range of 200 μs :

$$\begin{aligned}
u_1 &\leq 200 \mu\text{s} \\
u_2 &\leq 200 \mu\text{s} \\
u_3 &\leq 200 \mu\text{s} \\
u_4 &\leq 200 \mu\text{s}
\end{aligned}
\tag{3.42}$$

The above assumptions generate the following \mathbf{Q} and \mathbf{R} matrices:

$$\mathbf{Q} = \begin{bmatrix} 4 \times 10^4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 \times 10^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 \times 10^4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 \times 10^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 \times 10^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \times 10^{-8} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \times 10^{-8} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \times 10^{-8} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \times 10^{-8} \end{bmatrix} \quad (3.43)$$

$$\mathbf{R} = \begin{bmatrix} 2.50 \times 10^{-5} & 0 & 0 & 0 \\ 0 & 2.50 \times 10^{-5} & 0 & 0 \\ 0 & 0 & 2.50 \times 10^{-5} & 0 \\ 0 & 0 & 0 & 2.50 \times 10^{-5} \end{bmatrix} \quad (3.44)$$

In order to compute \mathbf{K} using \mathbf{Q} and \mathbf{R} , some of the system's properties must be found to complete the model in the system of equations 3.36. Principally, the controller will be implemented on a discrete system, so the timing parameter Δt for

the program is found experimentally by implementing the flight control software on the quadrotor's microcontroller and having a pin on the chip send a pulse during each program iteration. Other parameters necessary for computation of \mathbf{K} , such as the motor dynamics and vehicle inertias, have been found in Chapter 2 and computed via CAD models, respectively. The system parameters can be found in Table 3.1. With the values for the variables in equation 3.36 identified, \mathbf{K} is computed using the Discrete Algebraic Ricatti Equation (DARE) with MATLAB's `lqrd()` command, which gives:

$$\mathbf{K} = \begin{bmatrix}
 3.30 \times 10^{-12} & 5.14 \times 10^{-14} & 1.49 \times 10^3 & 1.79 \times 10^2 & 1.38 \times 10^2 & \dots \\
 -1.51 \times 10^3 & -1.82 \times 10^2 & -2.94 \times 10^{-11} & 2.58 \times 10^{-12} & -1.38 \times 10^2 & \dots \\
 -3.47 \times 10^{-11} & -3.13 \times 10^{-12} & -1.49 \times 10^3 & -1.79 \times 10^2 & 1.38 \times 10^2 & \dots \\
 1.51 \times 10^3 & 1.82 \times 10^2 & -1.37 \times 10^{-11} & -3.78 \times 10^{-12} & -1.38 \times 10^2 & \dots \\
 \dots & 1.37 \times 10^{-3} & -5.85 \times 10^{-6} & -9.46 \times 10^{-5} & -5.85 \times 10^{-6} & \\
 \dots & -5.85 \times 10^{-6} & 1.37 \times 10^{-3} & -5.85 \times 10^{-6} & -9.10 \times 10^{-5} & \\
 \dots & -9.46 \times 10^{-5} & -5.85 \times 10^{-6} & 1.37 \times 10^{-3} & -5.85 \times 10^{-6} & \\
 \dots & -5.85 \times 10^{-6} & -9.10 \times 10^{-5} & -5.85 \times 10^{-6} & 1.37 \times 10^{-3} &
 \end{bmatrix} \tag{3.45}$$

Since the values pertaining to attitude in the \mathbf{Q} matrix are large compared to both the rotor states and the control cost matrix \mathbf{R} , the attitude gains in \mathbf{K} are fairly

large. The emphasis on state accuracy will result in a system that responds quickly to minute changes, theoretically minimizing drift when the quadrotor tries to hold position. Notably, the gains associated with the rotor RPMs are orders of magnitude smaller than those for the attitude, since the square of the rotor velocities will be considerably larger than radians. These rotor states are not measured, however, and so an observer must be formulated as well.

Table 3.1: System Properties for LQR

I_x	$1.883 \times 10^{-4} \text{ kg} \cdot \text{m}^2$
I_y	$1.782 \times 10^{-4} \text{ kg} \cdot \text{m}^2$
I_z	$3.120 \times 10^{-4} \text{ kg} \cdot \text{m}^2$
l	0.08255 m
A	0.007241 m^2
R	0.04801 m
ρ	1.204 kg/m^3
α	18.50
K_m	1.195×10^5
Δt	8 ms (125Hz)

3.4 Luenberger Observer

The system requires that the rotor velocity be measured, but there are no sensors mounted on the quadrotor capable of measuring the new states. The values are instead estimated by a Luenberger observer, which uses the systems dynamical model. The technique computes the gain matrix \mathbf{L} that allows the system to predict unobserved states from measured ones via an algorithm similar to the Kalman filter. The observer computes the expected rotor velocities from the quadrotor's filtered state measurements and makes corrections during each iteration. The observer gain

itself is computed using LQR [17], wherein the transpose of the state and observation matrices are used in lieu of the state and control matrices, so the ARE would be

$$\mathbf{0} = \mathbf{A}\mathbf{P} + \mathbf{P}\mathbf{A}^T + \mathbf{Q} - \mathbf{P}\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\mathbf{P} \quad (3.46)$$

and the Luenberger observer gain is

$$\mathbf{L}^T = \mathbf{R}^{-1}\mathbf{C}\mathbf{P} \quad (3.47)$$

In this case, the matrices \mathbf{Q} and \mathbf{R} are identity for the sake of simplicity, and $\mathbf{C} \in \mathbb{R}^{5 \times 9}$:

$$\mathbf{C} = [\mathbf{I}_{5 \times 5} \ \mathbf{0}_{5 \times 4}] \quad (3.48)$$

because only the states measured by the IMU are actually observed. The resulting observer gain matrix, \mathbf{L} , is then used to compute the states estimates $\hat{\underline{x}}$ via the following expression:

$$\dot{\hat{\underline{x}}}(t) = \mathbf{A}\hat{\underline{x}}(t) + \mathbf{B}\underline{u}(t) + \mathbf{L}\mathbf{C}[\underline{x}(t) - \hat{\underline{x}}(t)] \quad (3.49)$$

where

$$\mathbf{L} = \begin{bmatrix} 6.18 \times 10^{-1} & 5.79 \times 10^{-3} & -2.22 \times 10^{-18} & 7.31 \times 10^{-18} & -3.28 \times 10^{-17} \\ 8.45 \times 10^{-4} & 6.18 \times 10^{-1} & 6.44 \times 10^{-19} & -1.64 \times 10^{-18} & -2.09 \times 10^{-17} \\ -2.16 \times 10^{-18} & 6.05 \times 10^{-19} & 6.18 \times 10^{-1} & 5.79 \times 10^{-3} & 5.38 \times 10^{-19} \\ 7.32 \times 10^{-18} & -1.58 \times 10^{-18} & 8.45 \times 10^{-4} & 6.18 \times 10^{-1} & -2.26 \times 10^{-17} \\ -3.27 \times 10^{-17} & -2.09 \times 10^{-17} & 7.19 \times 10^{-19} & -2.26 \times 10^{-17} & 6.18 \times 10^{-1} \\ -5.20 \times 10^{-15} & 1.75 \times 10^{-14} & 1.43 \times 10^{-8} & 3.02 \times 10^{-6} & 2.32 \times 10^{-7} \\ -1.35 \times 10^{-8} & -2.86 \times 10^{-6} & -1.33 \times 10^{-14} & -7.03 \times 10^{-16} & -2.32 \times 10^{-7} \\ -2.98 \times 10^{-15} & -7.24 \times 10^{-15} & -1.43 \times 10^{-8} & -3.02 \times 10^{-6} & 2.32 \times 10^{-7} \\ 1.35 \times 10^{-8} & 2.86 \times 10^{-6} & -3.37 \times 10^{-16} & -1.54 \times 10^{-15} & -2.32 \times 10^{-7} \end{bmatrix} \quad (3.50)$$

and \mathbf{L} is computed using MATLAB's `lqrd()` command, since the system is discrete. Implementing the Luenberger observer requires some additional coding since it introduces additional floating point data that needs to be stored on the microcontroller. Fortunately, there is a method by which the values can be stored into what is normally reserved for program memory. The values cannot, however, be changed during runtime, so it is important to assume the observer time-invariant. With the observer and controller gain matrices computed, the system can be tested in simulation and then proven in flight.

Chapter 4

Simulation and Testing

With the quadrotor dynamics formulated and a control algorithm established, the next step is testing the vehicle's performance. Testing the control algorithm on a physical system has to be carried out systematically, because there is no guarantee that the calculated control gains will work in a physical system. Instead, a Simulink simulation predicts the performance of the vehicle in various flight conditions. Thus, the simulation helps in tuning the control system as well as verifying the assumptions made in the control algorithm's formulation. Once the simulation shows that the control configuration results in a stable hover in the presence of measurement noise and other disturbance inputs, the system is implemented on-board the vehicle. To ensure experimental repeatability, the quadrotor is operated by an external computer acting as a pilot.

The autopilot is a position-hold system based on a proportional-integral control scheme that uses position and rate feedback from the Vicon visual positioning system. The translational control problem is essentially wrapped around the attitude control simulation in Simulink. Simulation results are then used to tune the position-hold gains iteratively with the Ziegler-Nichols algorithm. A successful simulation exhibiting level attitude in hover as well as position hold using an external autopilot leads to implementation in LabView on a laboratory PC. With on-board

attitude control and off-board position control implemented in the physical system, the quadrotor is flown in hover next. In addition to having the quadrotor hold its position, a series of experiments are performed to check the robustness of the system, including disturbance rejection and impulse response. Success in flight, however, relies on the fidelity of the simulation.

4.1 Simulation

A simulation's utility depends on its accuracy. In addition to the vehicle dynamics from Chapter 3, the Simulink model accounts for how data is handled in the embedded system and the other physical properties that affect the vehicle as a whole. The structure of the simulation as described in Figure 4.1 shows the association between control algorithms and data flow. For example, the continuous plant model computes the six vehicle states based on the attitude dynamics, and then the states are converted from the inertial to body reference frames to account for how the inertial sensors measure data. White noise is added to the sensor data to account for vibration and electrical noise experienced during operation. Then, the states are modified to account for data lost in converting from an analog to digital signal with the microcontroller's 10-bit resolution. A discrete Kalman filter using the gains calculated from the data in Section 2.3.1 proceeds to correct the sensor noise, then the measurements are converted back into the inertial reference frame. With the states corrected, the simulation discards the yaw state and a discrete Luenberger observer estimates of rotor velocities, populating the now 9 degree-of-freedom

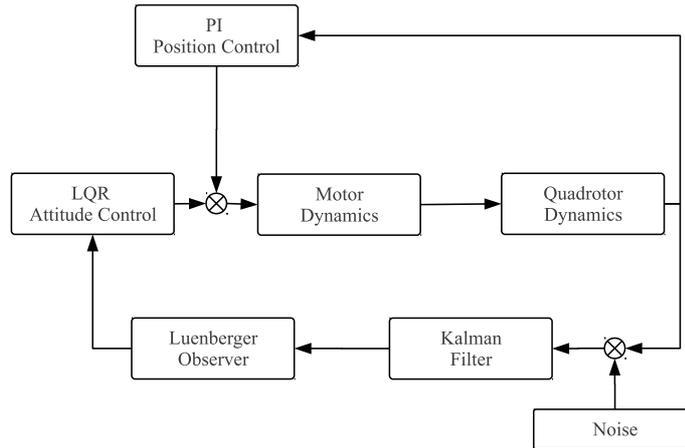


Figure 4.1: Depiction of data flow for the simulation.

state vector. The control inputs are then computed using the attitude and rotor states while taking into account motor control resolution and input saturation. A state-space representation of the continuous motor dynamics for all motors takes the control vector as input, computing the resulting rotor velocity vector that becomes the input for the plant model. The model transitions between discrete and continuous processes to take into account the implementation of algorithms in a physical environment. With the system properties determined and Simulink model built, simulations are performed to test the control system.

4.1.1 Attitude Control

Attitude control simulation testing consisted of observing the system response to noise, non-zero initial conditions, and impulse disturbances. Running a simulation with zero initial conditions and no external forces, the noise disrupted the system slightly (Fig. 4.2). Though the Kalman filter reduced the noise significantly, what remained was enough to induce small oscillations in the quadrotor's

attitude. In implementation, this could be avoided by using more accurate sensors, or by adding a damper to the sensor suite to physically alleviate noise level. The oscillations were still within the assumptions from Section 3.2, however, they were acceptable. Though the control algorithm neglected yaw, the heading stayed roughly around zero. Additionally, the assumptions made in the previous chapter held when looking at the components of the quadrotor dynamics negated by linearization. In Figure 4.2, the gyroscopic body components (Eq. 3.5) and gyroscopic rotor components (Eq. 3.7) were three orders of magnitude smaller than the motor moments (Eq. 3.10) in hover flight. While verifying previous assumptions, an examination of the error between predicted and simulated rotor states showed that the Luenberger observer worked properly (Fig. 4.3). The error between predicted and simulated RPMs reached a maximum of approximately 0.04 RPM, which was acceptable considering the changes in RPM for the vehicle were usually four orders of magnitude greater. These assumptions also held when introducing nonzero initial conditions; the system returned to equilibrium despite the sensor noise and neglected dynamics (Fig. 4.4). A transient occurred when the quadrotor corrected itself, but it returned to nominal conditions within 2 s. The simulation showed that the vehicle was capable of correcting in multiple axes successfully. Corrections for nonzero initial angles in multiple axes causes the heading to change to a nonzero value, though it stabilizes once the reference condition is reached. The nonlinear components of the attitude dynamics were generally two orders of magnitude smaller than the motor moments, though near the start of the simulation they were only an order of magnitude less than the motor moments. This was most likely due to

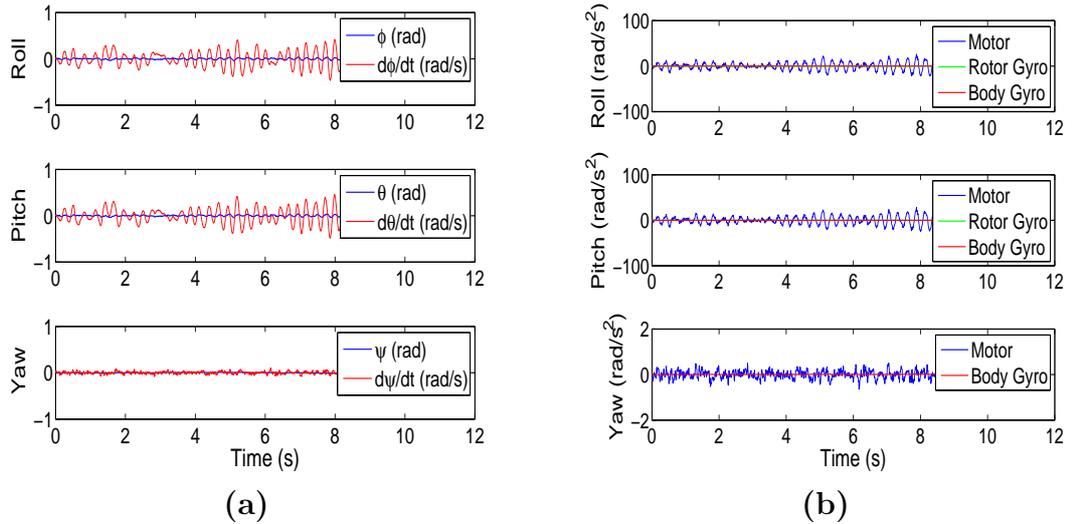


Figure 4.2: Results of hover simulation with zero initial conditions, showing (a) vehicle attitudes and (b) a comparison of the components of the dynamics.

the nonzero initial yaw rate. Increasing the initial states caused the nonlinearities to increase off-axis, but they were quickly corrected. The vehicle responded more quickly in impulse tests, in which a step input with a duration of one sample was added to the input vector. Since giving the same step input to all motors resulted in no net moment, the control input was set such that roll, pitch, and yaw moments were induced (Fig. 4.5). The system recovered from the impulses within 1.5 s. The attitude nonlinearities remained at least two orders of magnitude smaller than the motor moments, again maintaining previous assumptions.

The matrix \mathbf{K} in Section 3.3 as used in the above simulation was found by adjusting the \mathbf{Q} matrix and simulating the results. The measurement cost matrix was the focus since the attitude tolerances were arbitrary compared to the motor limits used in computing \mathbf{R} . For example, assuming $\mathbf{Q} = \mathbf{I}$ generated a response that took a minute to converge to equilibrium from nonzero initial conditions (Fig.

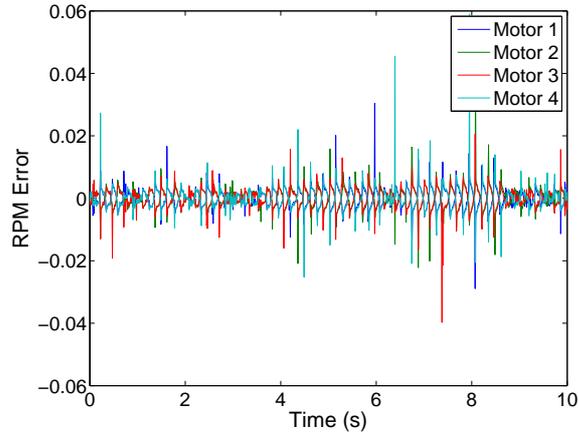


Figure 4.3: Error between predicted and simulated rotor states used to verify the Luenberger observer.

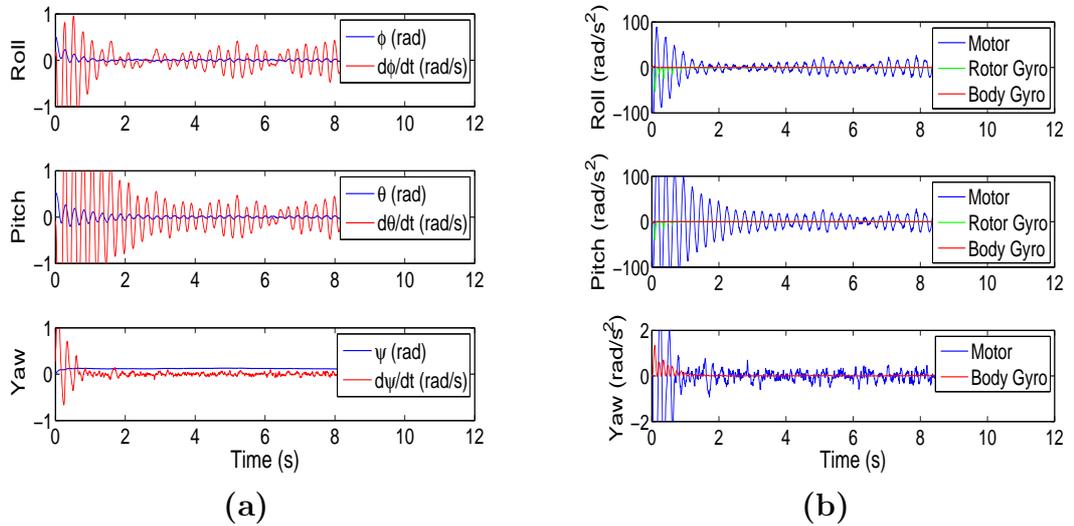


Figure 4.4: Results of hover simulation with $\underline{x}_0^T = [0.5 \ 0 \ 0.5 \ 0 \ 1 \ 0 \ 0 \ 0]$, showing (a) vehicle attitudes and (b) a comparison of the components of the dynamics.

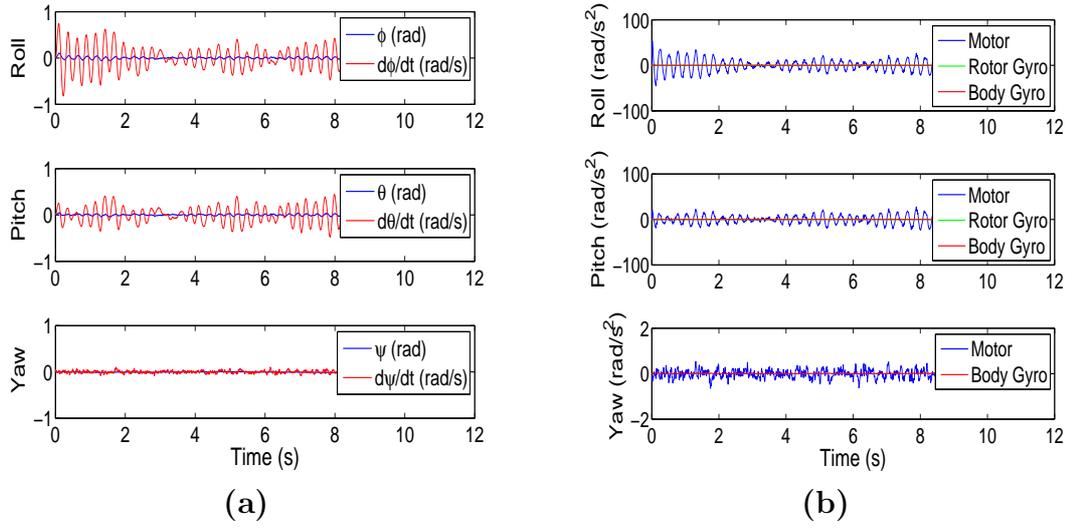


Figure 4.5: Results of hover simulation with impulse $\Delta \underline{u}^T = [50 \ -150 \ -50 \ 150]$, showing (a) vehicle attitudes and (b) a comparison of the components of the dynamics.

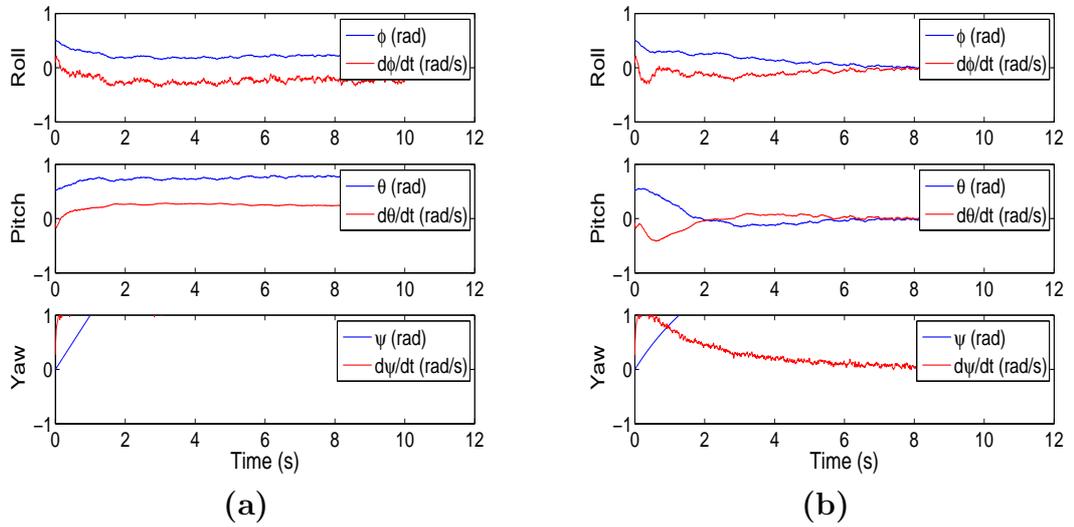
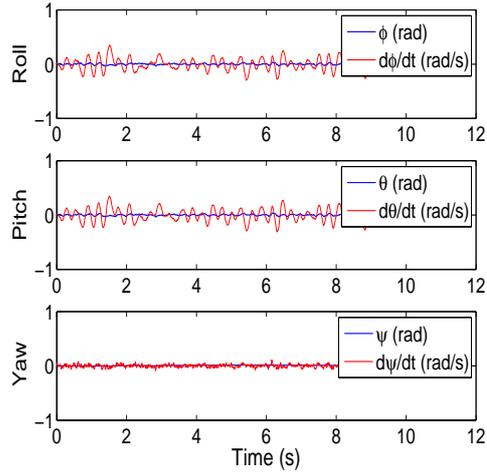
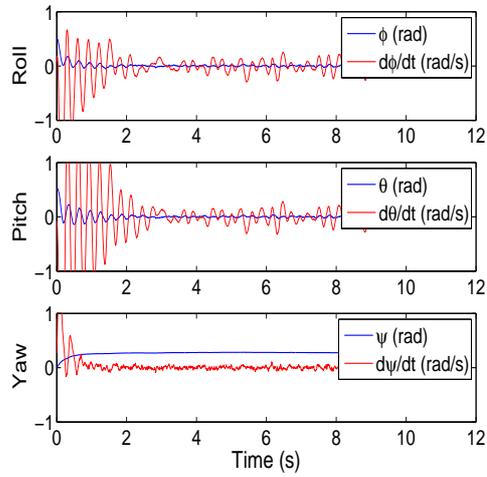


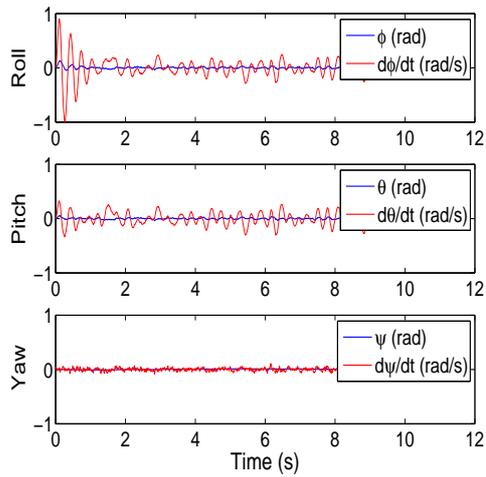
Figure 4.6: Attitude results of hover simulation with $\underline{x}_0^T = [0.5 \ 0 \ 0.5 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0]$ and controller tuned with (a) $\mathbf{Q} = \mathbf{I}$ and (b) $\mathbf{Q} = \text{diag}([1 \ 1 \ 1 \ 1 \ 1 \ 1 \times 10^{-8} \ 1 \times 10^{-8} \ 1 \times 10^{-8} \ 1 \times 10^{-8}])$.



(a)



(b)



(c)

Figure 4.7: Results of hover simulation with controller \mathbf{Q}_2 and (a) zero initial conditions, (b) $\underline{x}_0^T = [0.5 \ 0 \ 0.5 \ 0 \ 1 \ 0 \ 0 \ 0]$, and (c) an impulse $\Delta \underline{u}^T = [50 \ -150 \ -50 \ 150]$.

4.6a). Using the values in Section 3.3 for the rotor state references, but keeping the first five terms along \mathbf{Q} 's diagonal at unity, the simulated response converged much sooner, but well after five seconds (Fig. 4.6b). Ideally, the response would be faster to prevent large translational excursions. Reducing the attitude terms to ± 0.01 and the angular rates to ± 0.1 helped reduce the response time to less than 2 s, as well as had less oscillations than the original controller (Fig. 4.7b). The resulting measurement cost matrix using these values was

$$\mathbf{Q}_2 = \begin{bmatrix} 1 \times 10^4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 \times 10^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 \times 10^4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \times 10^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \times 10^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \times 10^{-8} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \times 10^{-8} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \times 10^{-8} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \times 10^{-8} \end{bmatrix} \quad (4.1)$$

The resulting controller performed similarly to \mathbf{Q} , though it experienced far reduced transients in all test cases. In all simulations, the controller using the gain matrix \mathbf{K}_2

generated using \mathbf{Q}_2 had greater state accuracy and did not oscillate nearly as much as the more aggressive controller (Fig. 4.7). Though the less aggressive controller appeared better suited in these simulations, the difference between the two controller became more apparent in position feedback simulation.

4.1.2 Position Control

Proper implementation of position control required modeling the translational vehicle dynamics in the simulation. The translational kinematics were neglected previously to emphasize the derivation of the on-board model-based control system for the quadrotor. The system used a proportional-integral controller to avoid complexity, but it required tuning. As mentioned previously, adjusting gains in simulation expedited free flight testing implementation of hardware. Also, damage to the vehicle was avoided or minimized in the initial flight tests. To tune the PI controller in simulation, a system model for translational dynamics must be formulated. According to Castillo, et al [8], the translational dynamics for the quadrotor was expressed as follows:

$$m\ddot{x}(t) = -(C_T\rho AR^2)(W_1(t) + W_2(t) + W_3(t) + W_4(t)) \sin(\theta(t))$$

$$m\ddot{y}(t) = (C_T\rho AR^2)(W_1(t) + W_2(t) + W_3(t) + W_4(t)) \cos(\theta(t)) \sin(\phi(t))$$

$$m\ddot{z}(t) = -(C_T\rho AR^2)(W_1(t) + W_2(t) + W_3(t) + W_4(t)) \cos(\theta(t)) \cos(\phi(t)) + mg$$

The above equations were in the body reference frame. Once the equations were implemented in the simulation, it was possible to construct the position control loop

around the attitude simulation. To do so, the external loop was regarded as a pilot adding inputs to the attitude control law. Thus, the control law in Equation 3.37, using the discrete gain matrix \mathbf{K} , became

$$\underline{u}_k = \underline{u}_{p,k} - \mathbf{K}\underline{x}_k \quad (4.2)$$

where $\underline{u}_{p,k}$ were the inputs computed by the PI algorithm at sample k . Vehicle motion in the x- and y-direction depended on the pitch and roll of the vehicle, respectively. By using the components of the vertical thrust in the x- and y-axes, the desired translation was achieved. As such, the inputs for motors 1 and 3 controlled motion along the x-axis, and motors 2 and 4 governed motion along the y-axis. Net thrust along the z-axis affected vertical translation and were added to all motor inputs. As a result, the discrete control law describing \underline{u}_p was

$$\underline{u}_{p,k} = \begin{bmatrix} K_{i,x}x_k + K_{p,x}\dot{x}_k + K_{i,z}z_k + K_{p,z}\dot{z}_k \\ -K_{i,y}y_k - K_{p,y}\dot{y}_k + K_{i,z}z_k + K_{p,z}\dot{z}_k \\ -K_{i,x}x_k - K_{p,x}\dot{x}_k + K_{i,z}z_k + K_{p,z}\dot{z}_k \\ K_{i,y}y_k + K_{p,y}\dot{y}_k + K_{i,z}z_k + K_{p,z}\dot{z}_k \end{bmatrix} \quad (4.3)$$

Since z-position and rate affected all motor inputs, the gains $K_{i,z}$ and $K_{p,z}$ were tuned first, followed by tuning the x-axis gains $K_{i,x}$ and $K_{p,x}$. Assuming symmetry, the values for $K_{i,y}$ and $K_{p,y}$ were assumed to be equivalent to their x-axis counterparts. The gains were tuned using the Ziegler-Nichols method for PI gain tuning [4]. Before starting, the simulation was set up such that all initial conditions were zero, and any

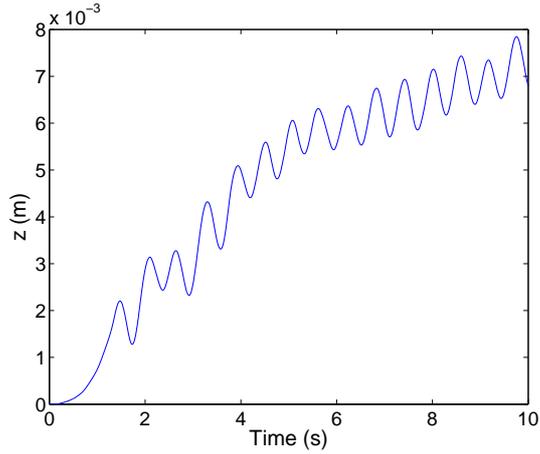


Figure 4.8: An example of the unstable oscillation induced by increasing the proportional gain for the z -axis.

additional disturbances neglected, with the exception of sensor noise. Once set, the simulation was started with $K_{p,z}$ set to an arbitrary value that will send the system response into unstable, sustained oscillations (Fig. 4.8). Then, the lowest value for the gain to send the system into instability was found using binary search.

Table 4.1: Position Control Gains

	\mathbf{K}	\mathbf{K}_2
$K_{i,x}$	60	$40 \mu s/m$
$K_{p,x}$	40	$28 \mu s \cdot s/m$
$K_{i,y}$	60	$40 \mu s/m$
$K_{p,y}$	40	$28 \mu s \cdot s/m$
$K_{i,z}$	132	$146 \mu s/m$
$K_{p,z}$	68	$66 \mu s \cdot s/m$

The period of the oscillations was measured and used to help calculate the values for $K_{i,z}$ and $K_{p,z}$ using the formulas

$$K_p = 0.45K_u \quad (4.4)$$

$$K_i = 1.2K_p/T_u \quad (4.5)$$

where K_u was the unstable gain and T_u was the period of the oscillations. Gains $K_{i,x}$ and $K_{p,x}$ were found similarly, with the final values associated with both control gain matrices shown in Table 4.1. In hover, the simulation showed the quadrotor maintaining its position as intended (Fig. 4.9 and 4.10). The vehicle's oscillations in hover visibly disturbed the position of the vehicle, but the quadrotor stayed within an $0.1 \times 0.1 \times 0.005$ m area about the origin. There appeared to be little difference between the two controllers thus far. Initializing the vehicle at a nonzero attitude did not adversely affect the vehicle (Fig. 4.11 and 4.12) and neither did an impulse (Fig. 4.13 and 4.14). The more aggressive controller appeared to perform slightly better than the other in this instance. Flight paths generated using the **K** controller exhibited less error, as did its attitudes despite transients in the angular rates. Interactions between the autopilot and the attitude controller were observed, but these did not destabilize the vehicle in either attitude or position. With the two control algorithms tested in simulation, these were then implemented in physical systems.

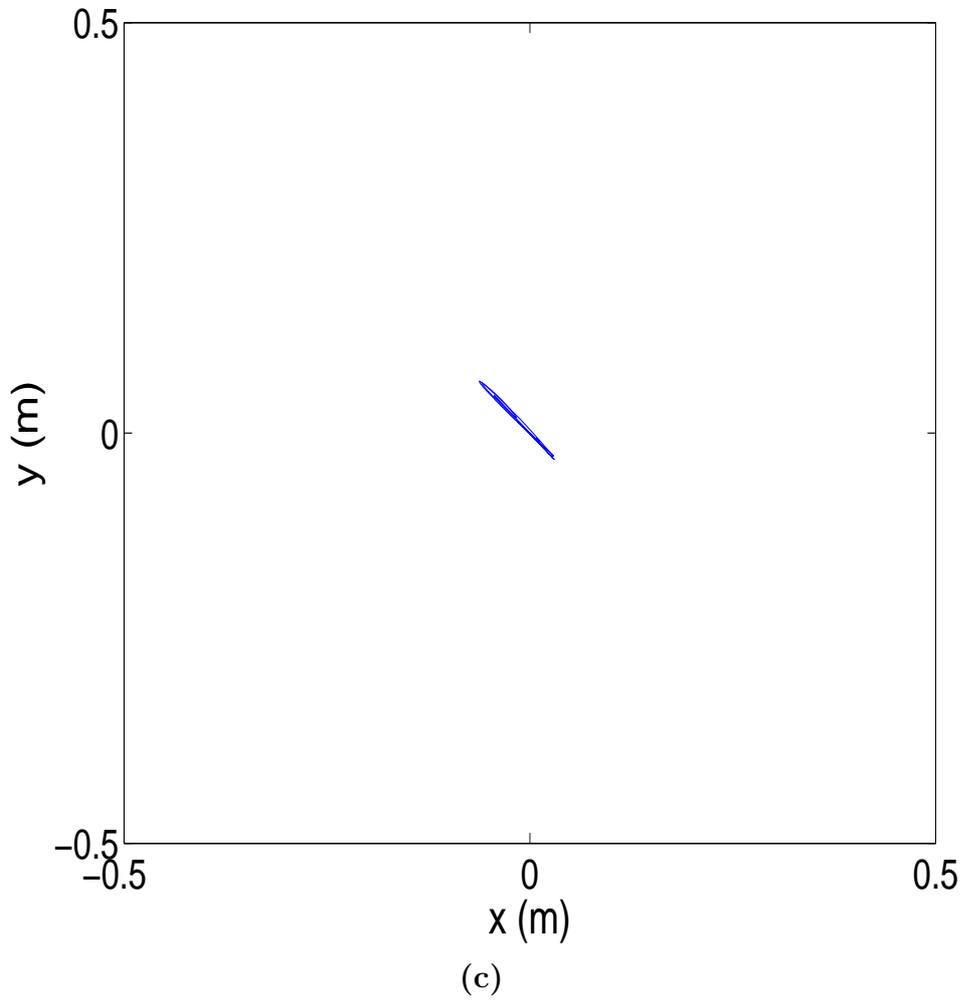
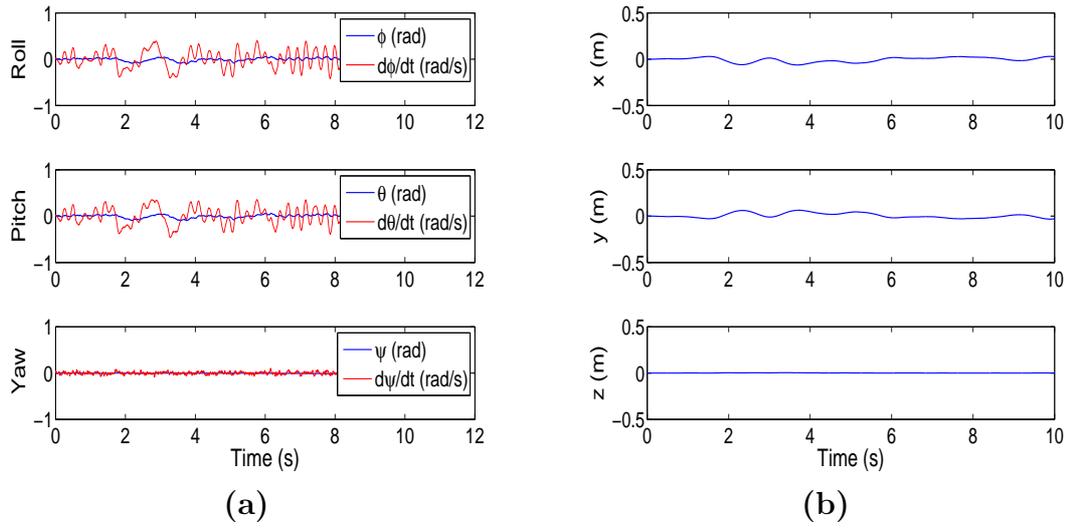


Figure 4.9: Results of hover simulation with position control and zero initial conditions, showing (a) vehicle attitudes, (b) translation, and (c) flight path.

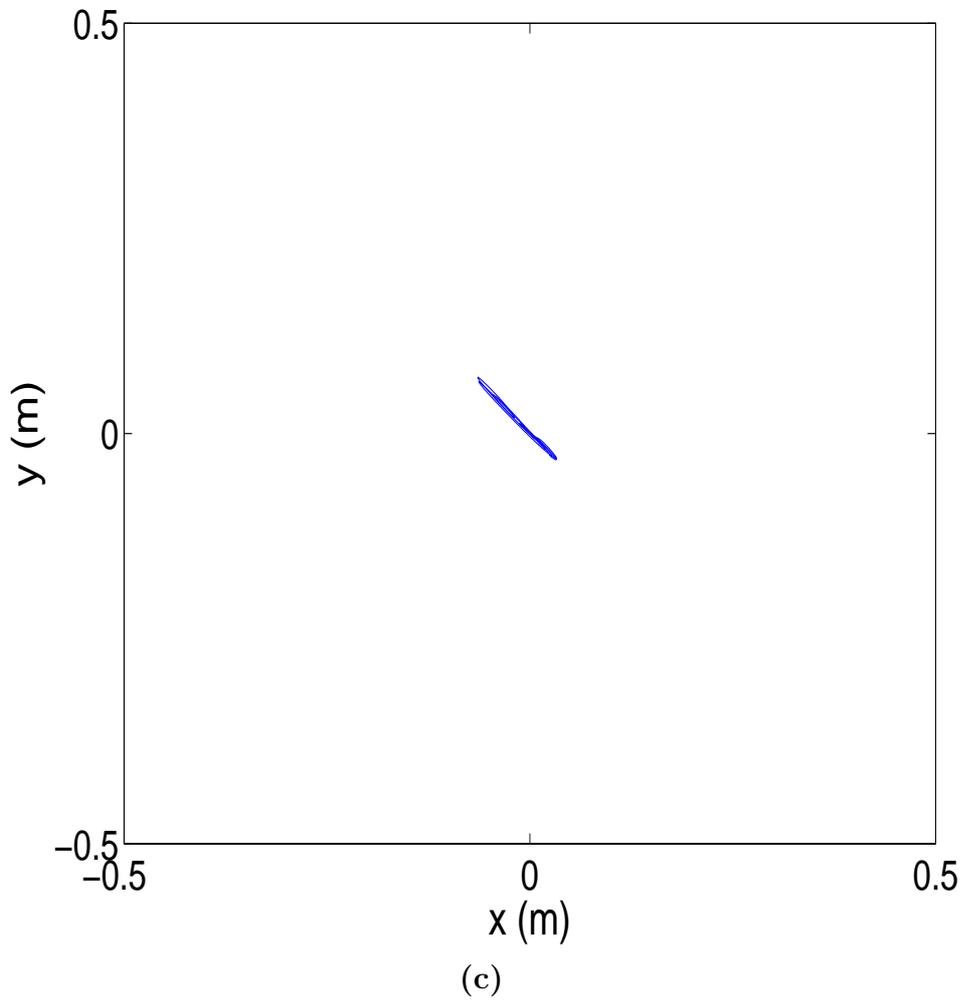
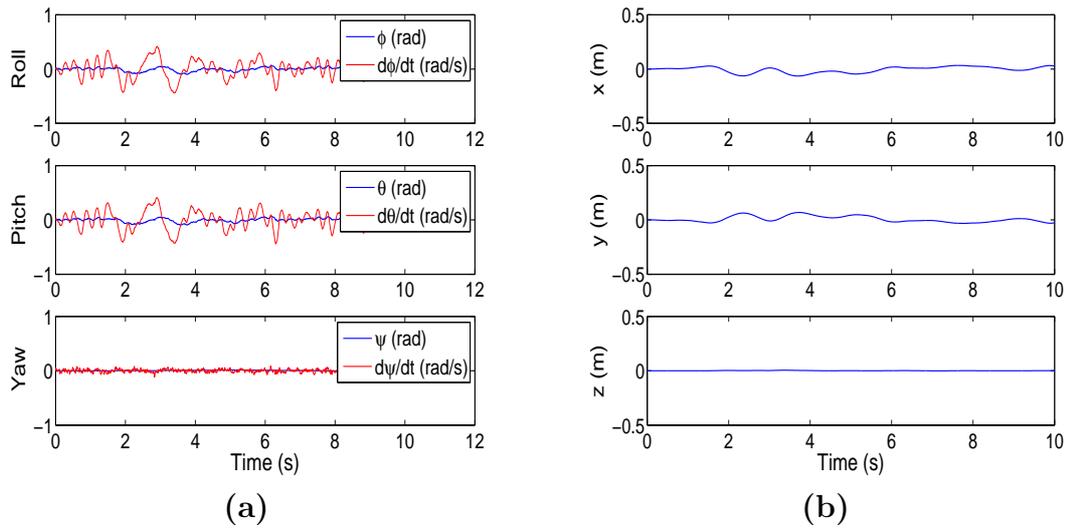


Figure 4.10: Results of hover simulation with position control, \mathbf{K}_2 controller, and zero initial conditions, showing (a) vehicle attitudes, (b) translation, and (c) flight path.

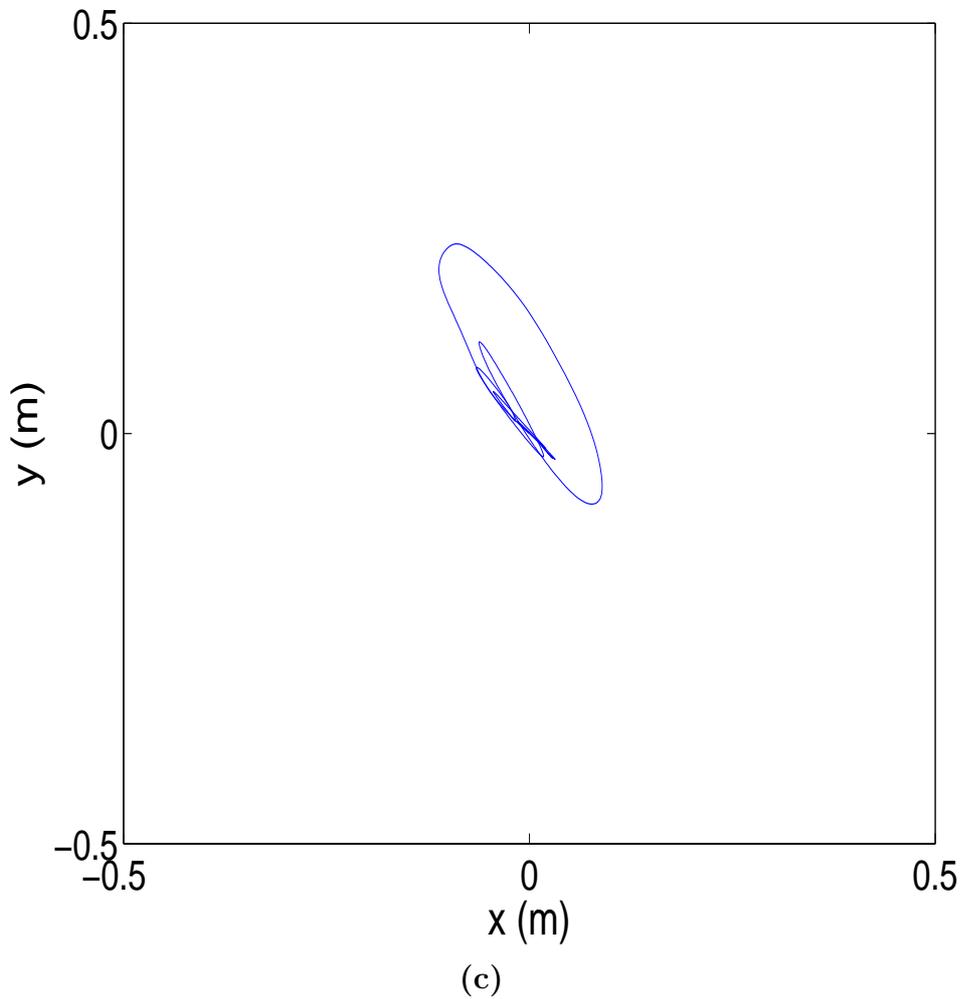
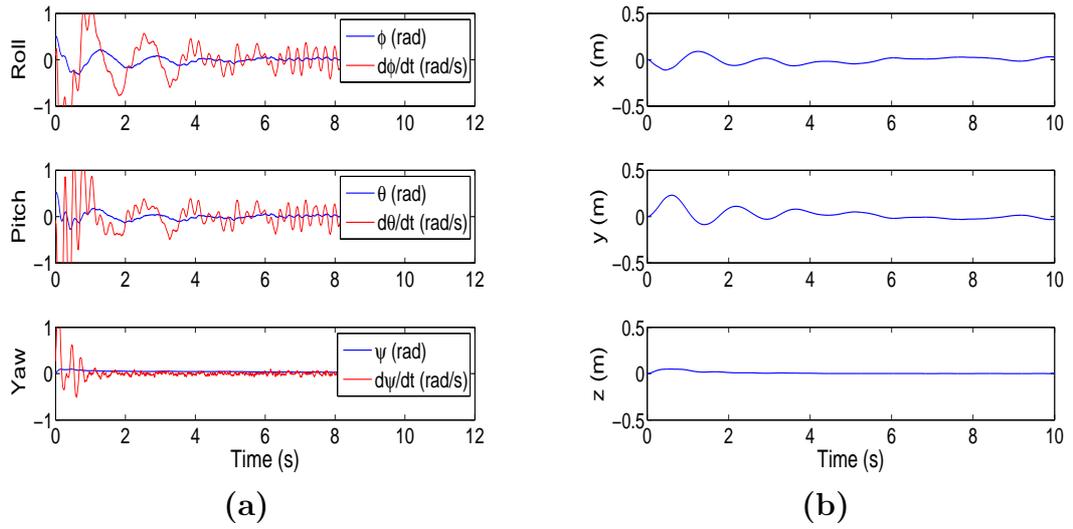


Figure 4.11: Results of hover simulation with position control and $\underline{x}_0^T = [0.5 \ 0 \ 0.5 \ 0 \ 1 \ 0 \ 0 \ 0]$, showing (a) vehicle attitudes, (b) translation, and (c) flight path.

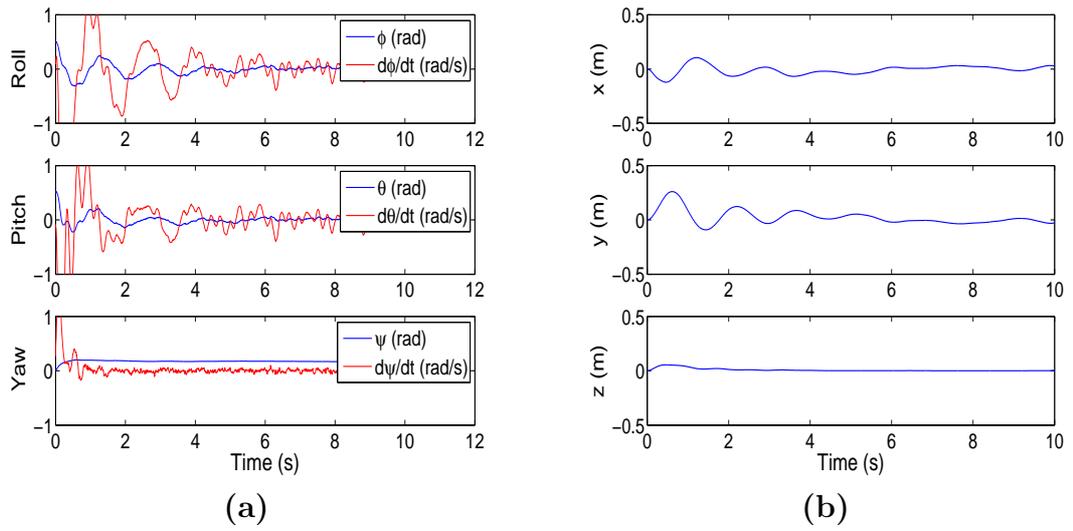


Figure 4.12: Results of hover simulation with position control, \mathbf{K}_2 controller, and $\underline{x}_0^T = [0.5 \ 0 \ 0.5 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0]$, showing (a) vehicle attitudes, (b) translation, and (c) flight path.

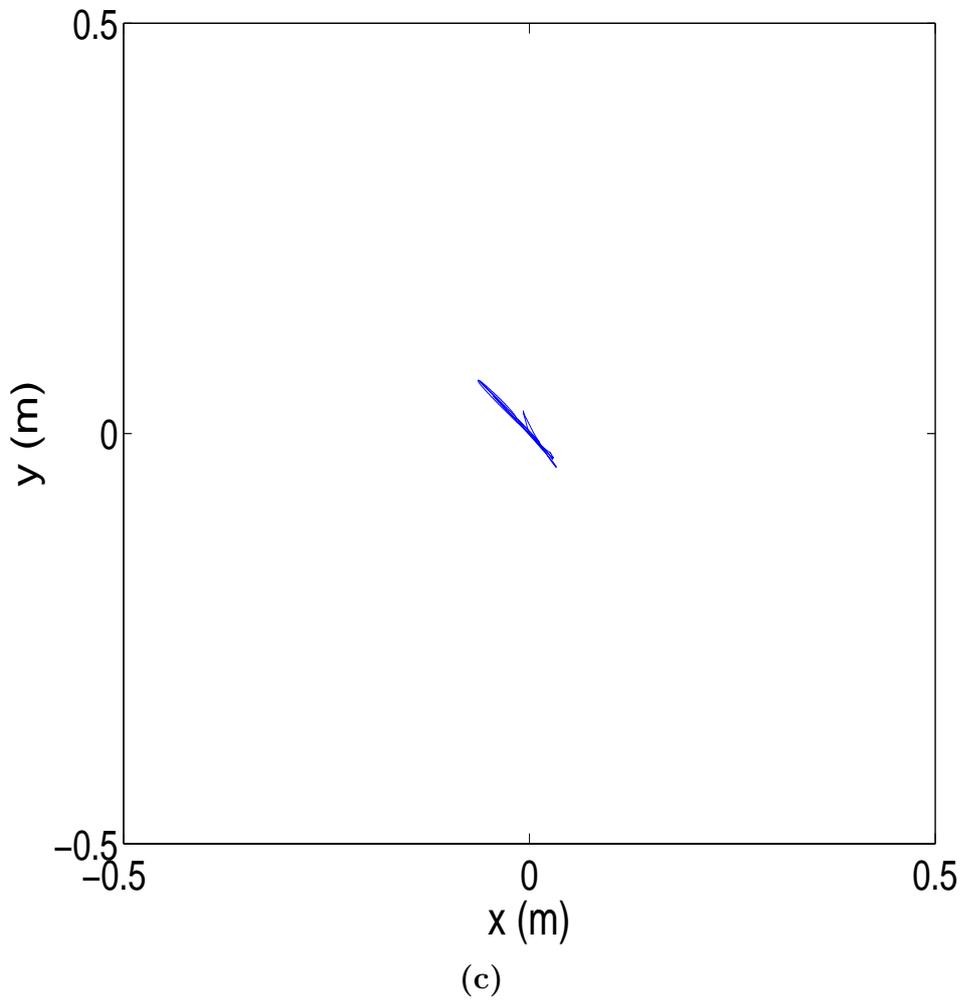
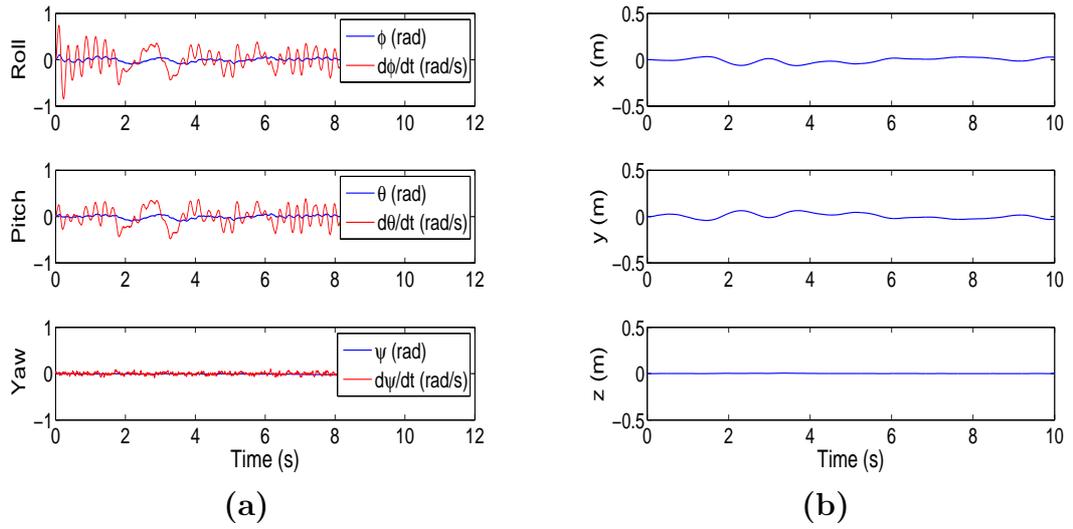


Figure 4.13: Results of hover simulation with position control and impulse $\Delta \underline{u}^T = [50 \ -150 \ -50 \ 150]$, showing (a) vehicle attitudes, (b) translation, and (c) flight path.

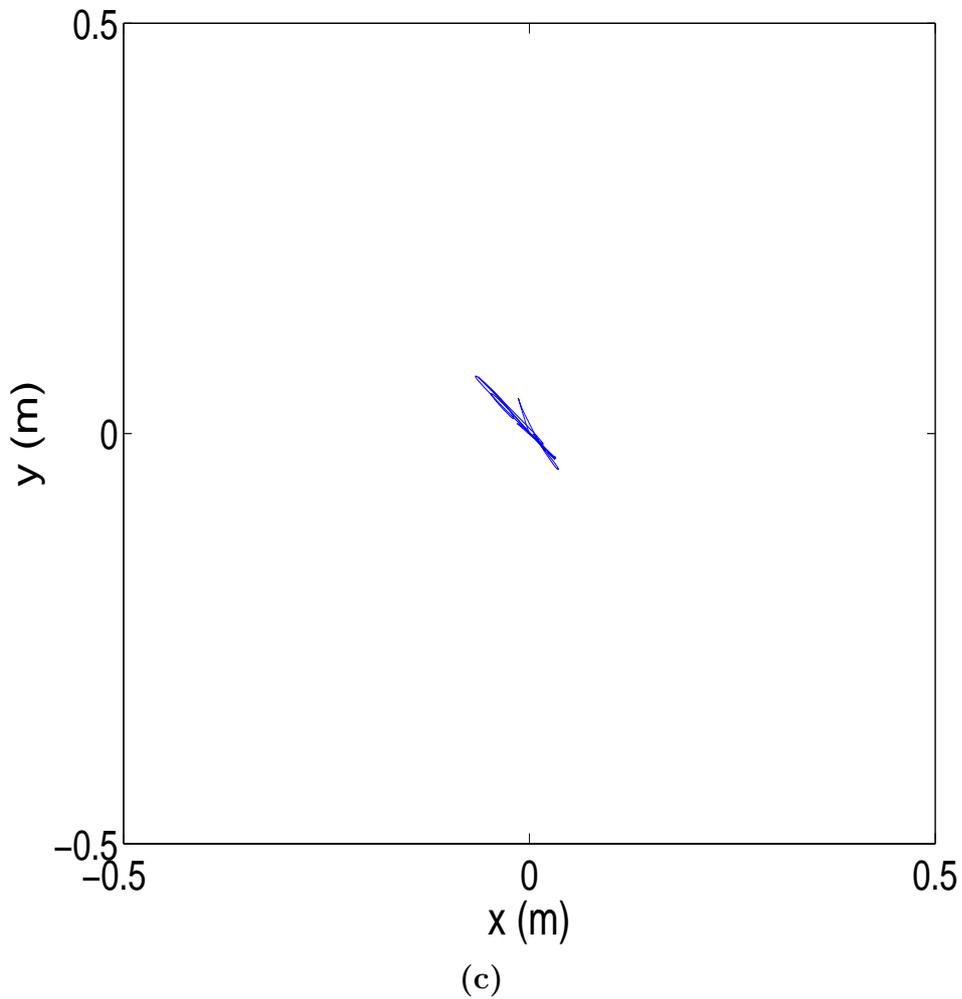
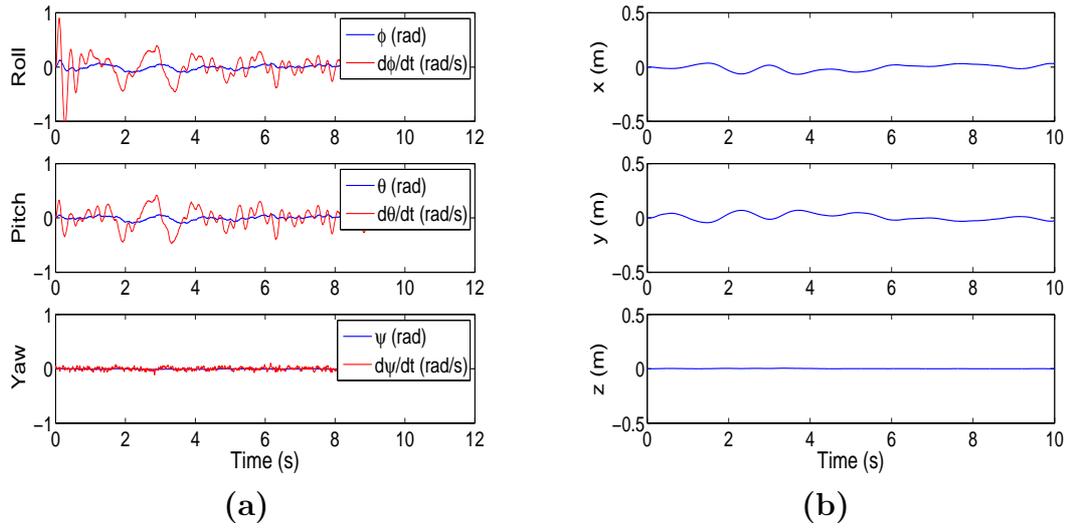


Figure 4.14: Results of hover simulation with position control, \mathbf{K}_2 controller, and impulse $\Delta \underline{u}^T = [50 \ -150 \ -50 \ 150]$, showing (a) vehicle attitudes, (b) translation, and (c) flight path.

4.2 Flight Testing

Having determined via simulation that the gains computed in Sections 3.3 and 4.1.2 allowed the quadrotor to operate in stable hover while maintaining its position, the physical vehicle was tested and the efficacy of the simulations verified. First, the control methods were implemented on their respective systems. The quadrotor then underwent a series of tests to determine its performance as well as the reliability of the simulation, provided that the software and hardware were cooperating.

4.2.1 Implementation and Testing in Vicon

The attitude control and estimation algorithms were coded into the Arduino microcontroller as described in Chapter 2. As for the position control system, the PI controller was added to a VI in LabView on a laboratory PC. The VI communicated with other programs written by AVL (UMD) researchers to extract pertinent state information from the Vicon position system (Fig. 4.15). During initialization, the program zeroed the initial conditions relative to the vehicle's starting position, then subtracted the current state data from reference conditions set by the user. In effect, the PC user could fly the quadrotor by changing the reference conditions, though the only reference changed during testing was vehicle altitude. The new state information was then used to compute the control inputs as in Equation 4.3, input to the quadrotor via bluetooth, and used the control law from Equation 4.2. The program wrote the control inputs from the PC as well as the states observed by the Vicon system to a file for post-processing in Matlab. After software and

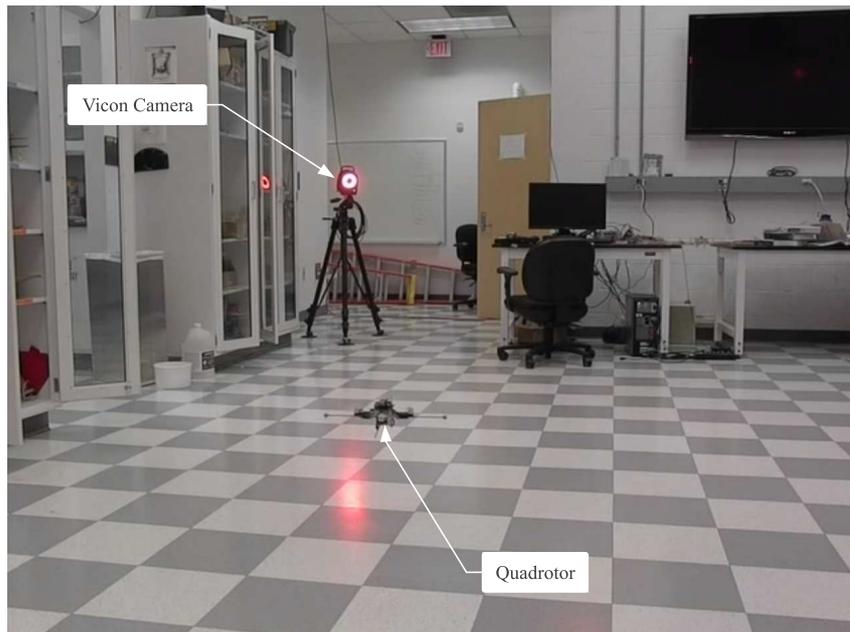


Figure 4.15: Laboratory setup for quadrotor flight testing in Vicon (one of eight cameras shown).

hardware communication was confirmed, the vehicle was tested.

4.2.2 Quadrotor Flight Testing in Vicon

Before any testing could take place, Vicon was calibrated and the quadrotor airframe modeled in the program Tracker. Silver retro-reflective balls were placed on the ends of the fore, left, right pylons, and on top of the bluetooth module to create a triangular model in the software (Fig. 2.1). Once Tracker identified and followed the vehicle around the test area adequately, the LabView VI was opened and the vehicle placed in the middle of the experimental space. With the quadrotor as level as possible on the floor, the vehicle and Labview were activated in an arbitrary order. For ten seconds, the quadrotor initialized the motors as well as averaged the sensor measurements to establish a level reference attitude. The x- and y-position

references were zeroed in the LabView program, and the altitude (z) reference was set to -1 m, which kept the motor inputs as low as possible to prevent a sudden start-up. Once the bluetooth connection was established properly, the microcontroller was ready, and the motor safety disengaged via the PC and the motors turned on. The quadrotor received a chain of 8-bit input channels encoded in a serial stream, so it got motor inputs from 0 to 255. The 0 bit acted as a header for the quadrotor to begin reading the data, while 1 acted as the safety signal; any value greater than 1 was interpreted as a motor command. With the safety off, the motors started and the z -reference indicator on the PC increased until the vehicle barely began to slide along the ground. Upon reaching this condition, the desired altitude was quickly set to 0.5 m and the quadrotor jumped to the desired height. In stable hover, the quadrotor flew at its designed equilibrium condition and stability experiments were performed. First, the MAV flew in hover for 30 s without interruption. At the end of 30 s, noting that the vehicle's operation appeared stable, the operator left the PC and imparted an impulse on the x and y axes individually by tapping the ends of the safety pylons (Fig. 4.16a). Each tap was followed by ten seconds without interruption to observe the behavior of the vehicle. Once impulse testing was complete, the z -reference was gradually lowered gently to land. The quadrotor's performance was also tested in the presence of a wind disturbance. After taking the quadrotor up into level hover, a box fan was directed at the MAV and turned on (Fig. 4.16b). The airflow was unsteady but averaged and tested at 2 m/s. Each test was completed with three trials to demonstrate repeatability (Fig. 4.17). However, only one of each test was selected for display in this document. Once testing completed,

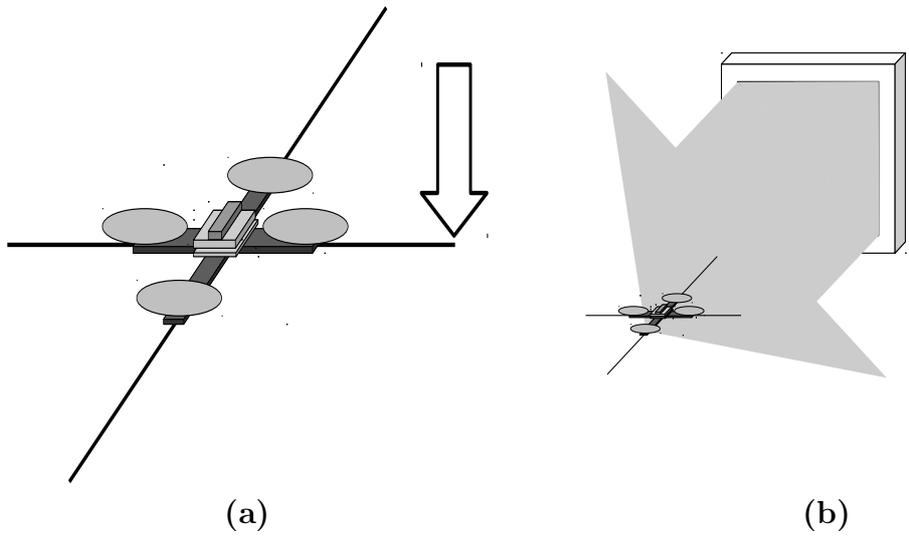
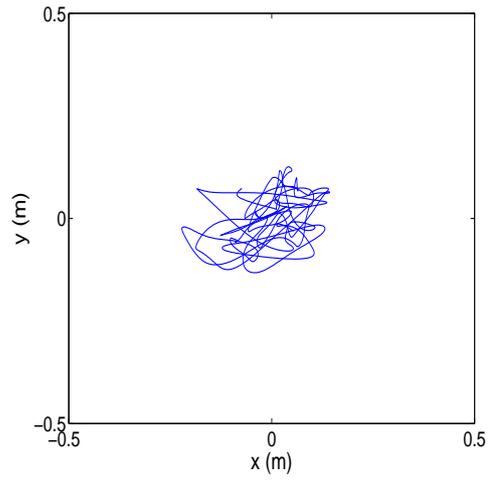
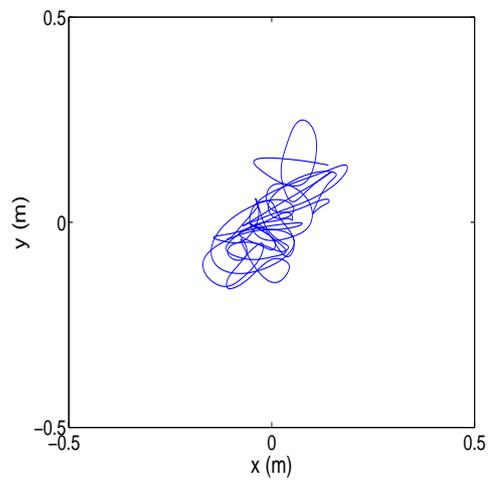


Figure 4.16: Diagrams depicting tests conducted during flight, which include (a) imparting an impulse on a pylon and (b) subjecting the vehicle to a wind disturbance.

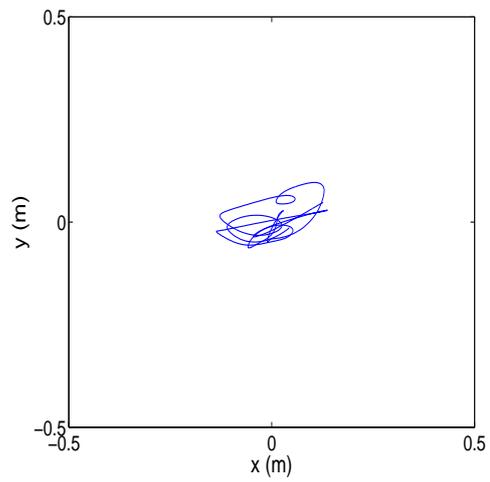
the flight data was analyzed.



(a)



(b)



(c)

Figure 4.17: Results of three flight tests with position control in undisturbed hover.



Figure 4.18: Video frames from the quadrotor impulse flight test.

4.2.3 Quadrotor Flight Test Results

Despite the assumptions made in implementing a linear control algorithm on the nonlinear quadrotor system, the vehicle successfully achieved level hover. As seen in Figures 4.20 and 4.19, the vehicle experienced some oscillations in each axis, visually manifesting as a rolling oscillation, similar to a spinning top precessing. This behavior appeared in simulation as well (Fig. 4.2), so the phenomenon was expected. However, the reason for this transient was still under investigation. A possibility was that the sensor noise was too large and the measurement bandwidth too small such that aliasing occurred. Because the control algorithm used state feedback, any results of aliasing could appear in the inputs and excite the system. Similar behavior was noted in the AVL MicroQuad, which also suffered from sensor noise issues, though there were also problems with the motor dynamics as well. Most remarkably, however, was the increased accuracy of hover in the \mathbf{K} controller over \mathbf{K}_2 despite the unattractive transients in simulation. Though the vehicle oscillated slightly in these axes, its stability and station keeping accuracy became readily apparent when given an impulse (Figs. 4.22, 4.21, and 4.18). Any transient from the impulse quickly died and the quadrotor returned to hover almost immediately. This performance was predicted as well, in which the vehicle returned to equilibrium

within 2 s. Disturbance testing proved just as promising (Fig. 4.24 and 4.23), though more so for the \mathbf{K} controller. In both instances, the vehicle shifted from its reference position but maintained level hover. However, controller \mathbf{K}_2 diverged approximately 0.5 m from its original position, whereas the \mathbf{K} controller only shifted a maximum of 0.3 m. Changes in reference condition were expected, since the position control used PI with the proportional term using velocity, and thus the inputs became constant in the absence of a term for integrated position, as opposed to position itself. Still, the precision of the second controller was exceptional, as foreshadowed in the position-hold simulations. Overall, the more aggressive controller computed in Section 3.3 performed best when coupled with automated position control, but both control gain matrices resulted in a stable vehicle.

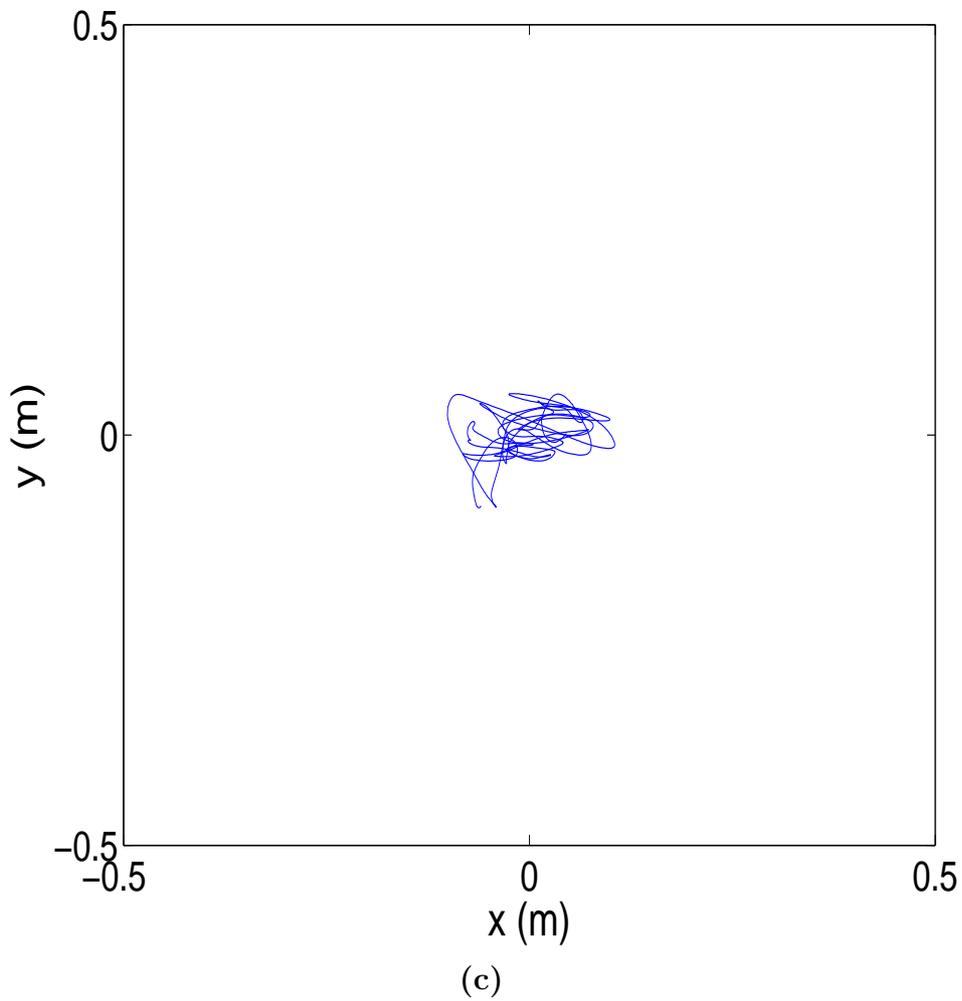
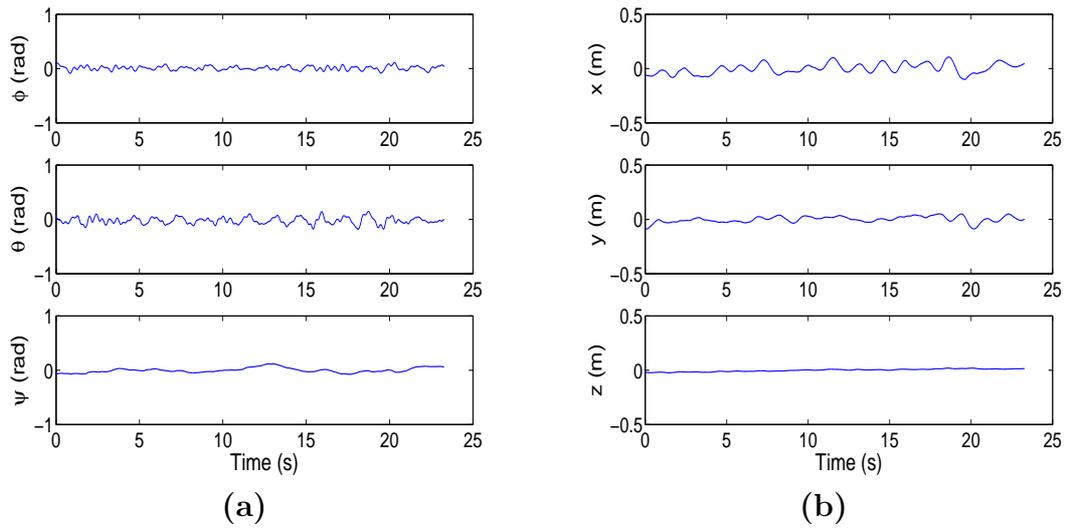


Figure 4.19: Results of flight test with position control in undisturbed hover, showing (a) vehicle attitudes, (b) translation, and (c) flight path.

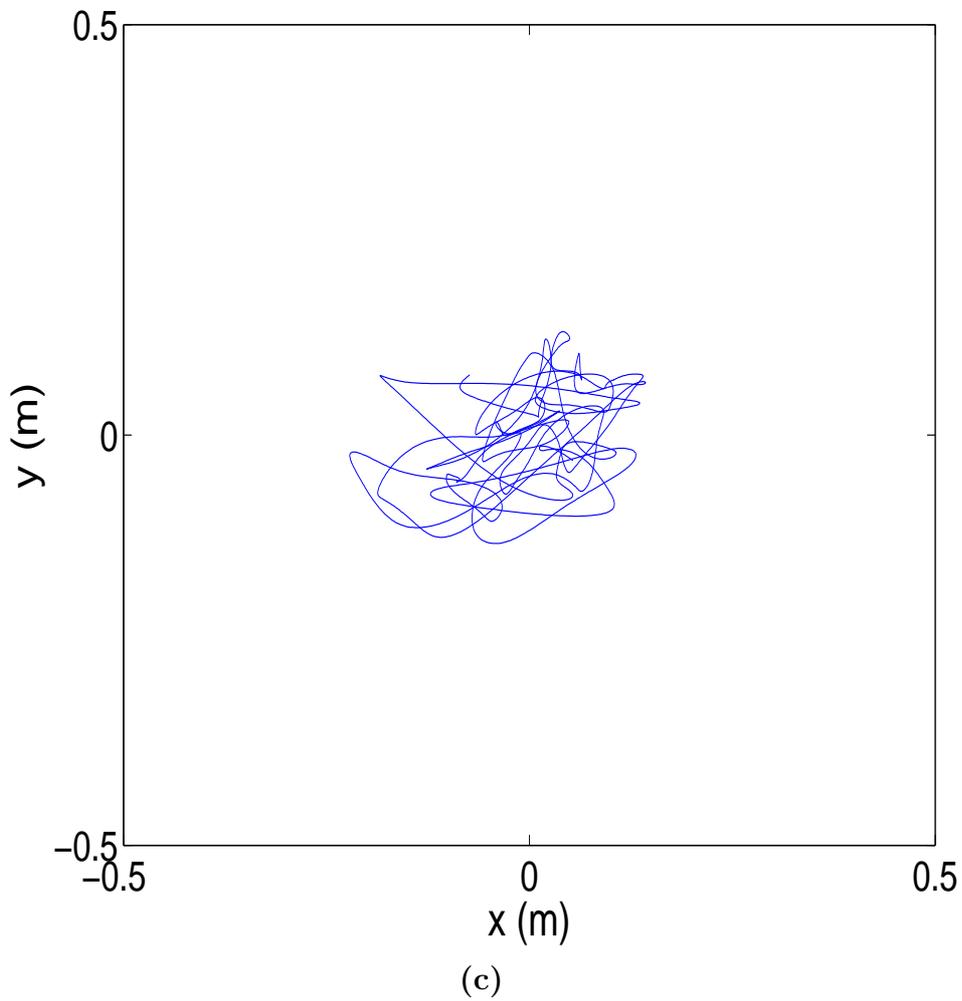
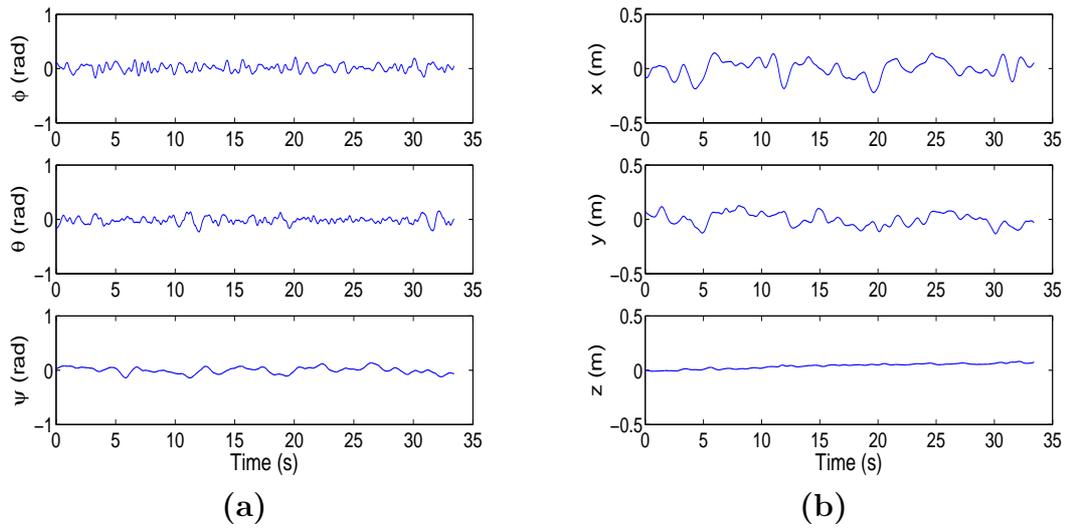


Figure 4.20: Results of flight test with position control and \mathbf{K}_2 controller in undisturbed hover, showing (a) vehicle attitudes, (b) translation, and (c) flight path.

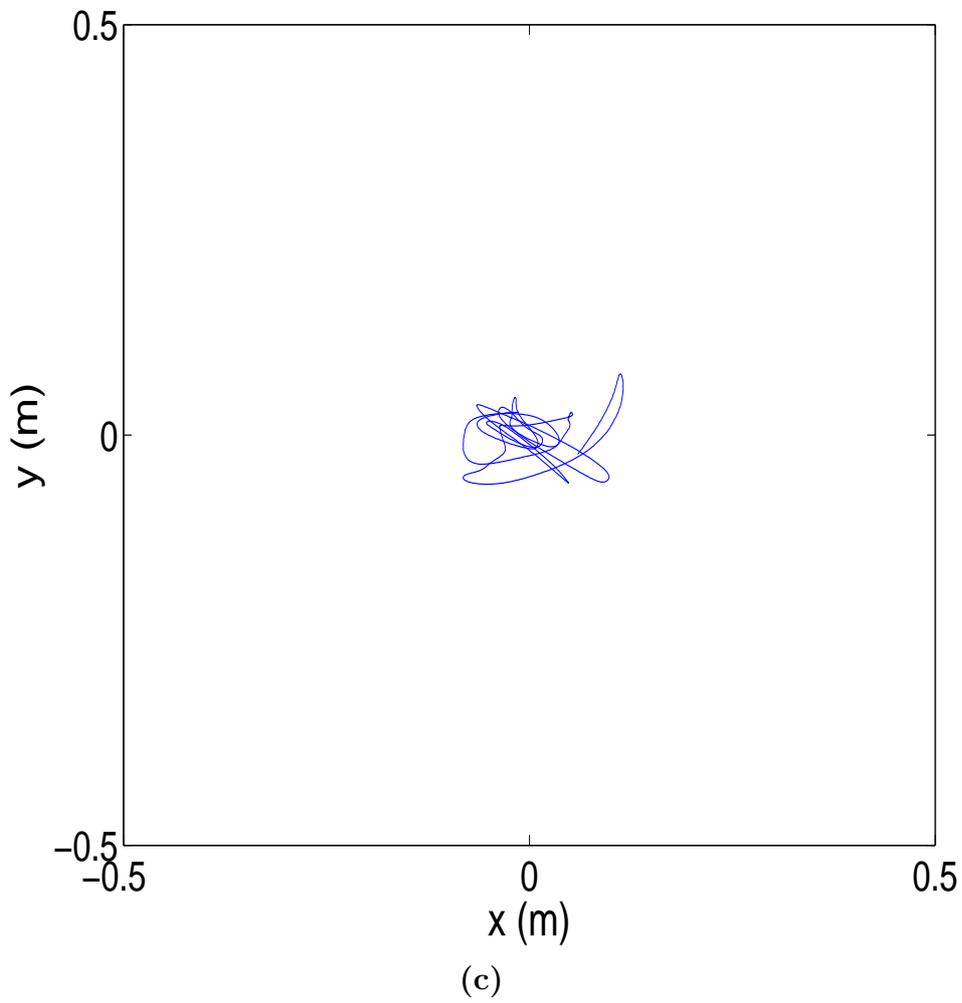
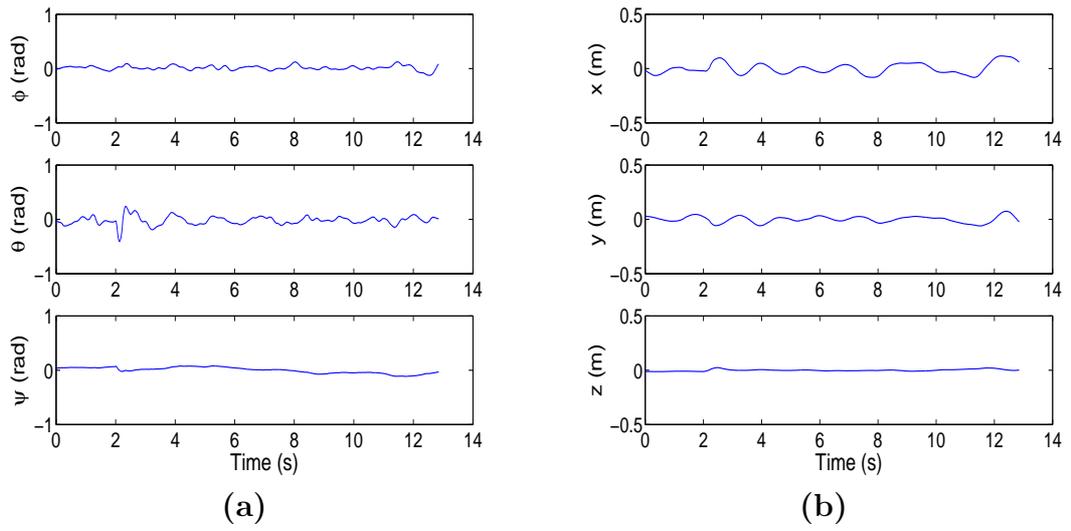


Figure 4.21: Results of flight test with position control and impulse on x-axis pylon, showing (a) vehicle attitudes, (b) translation, and (c) flight path.

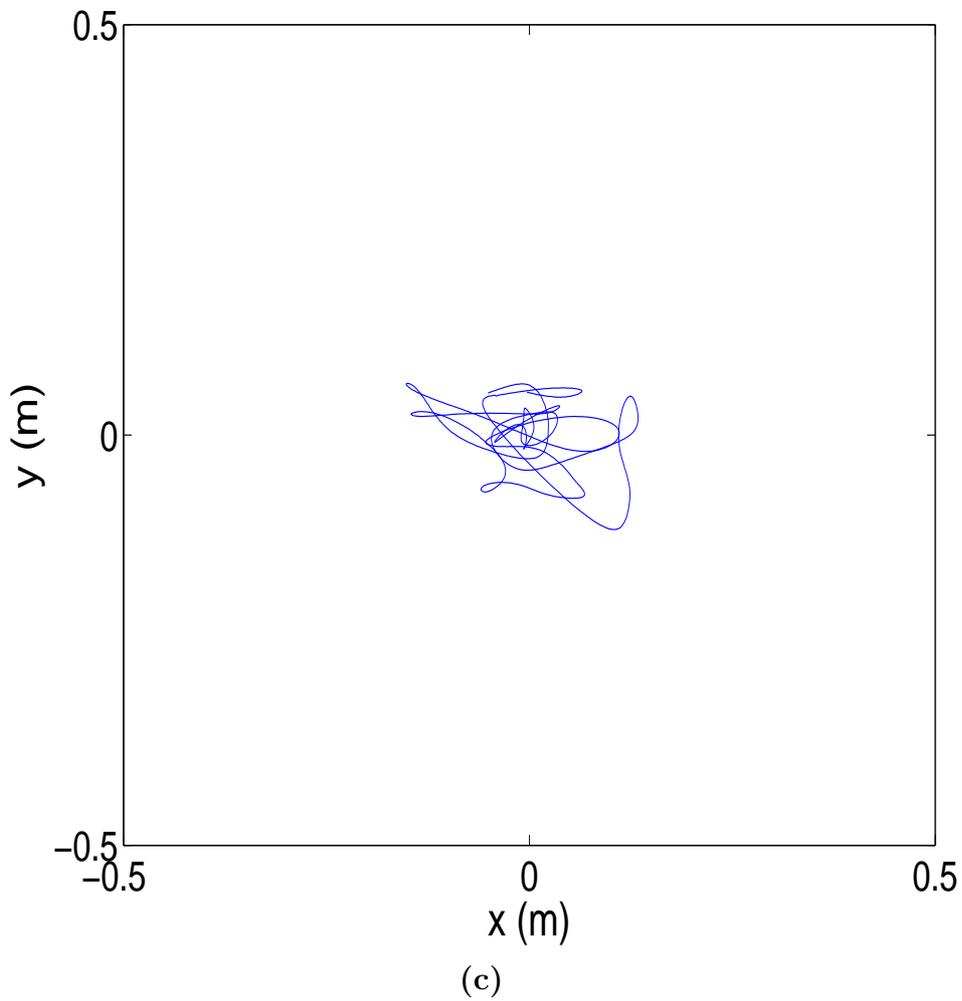
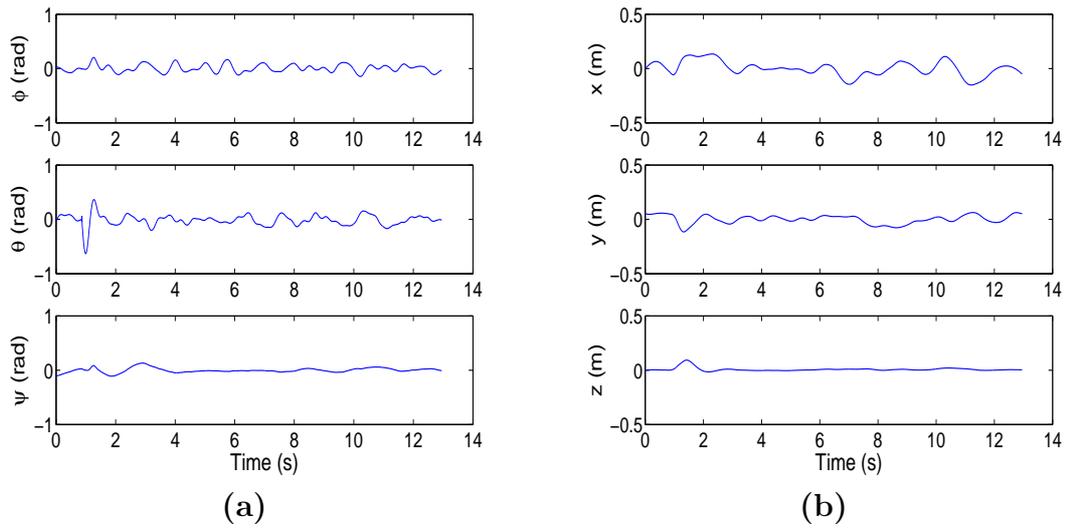


Figure 4.22: Results of flight test with position control, \mathbf{K}_2 controller, and impulse on x-axis pylon, showing (a) vehicle attitudes, (b) translation, and (c) flight path.

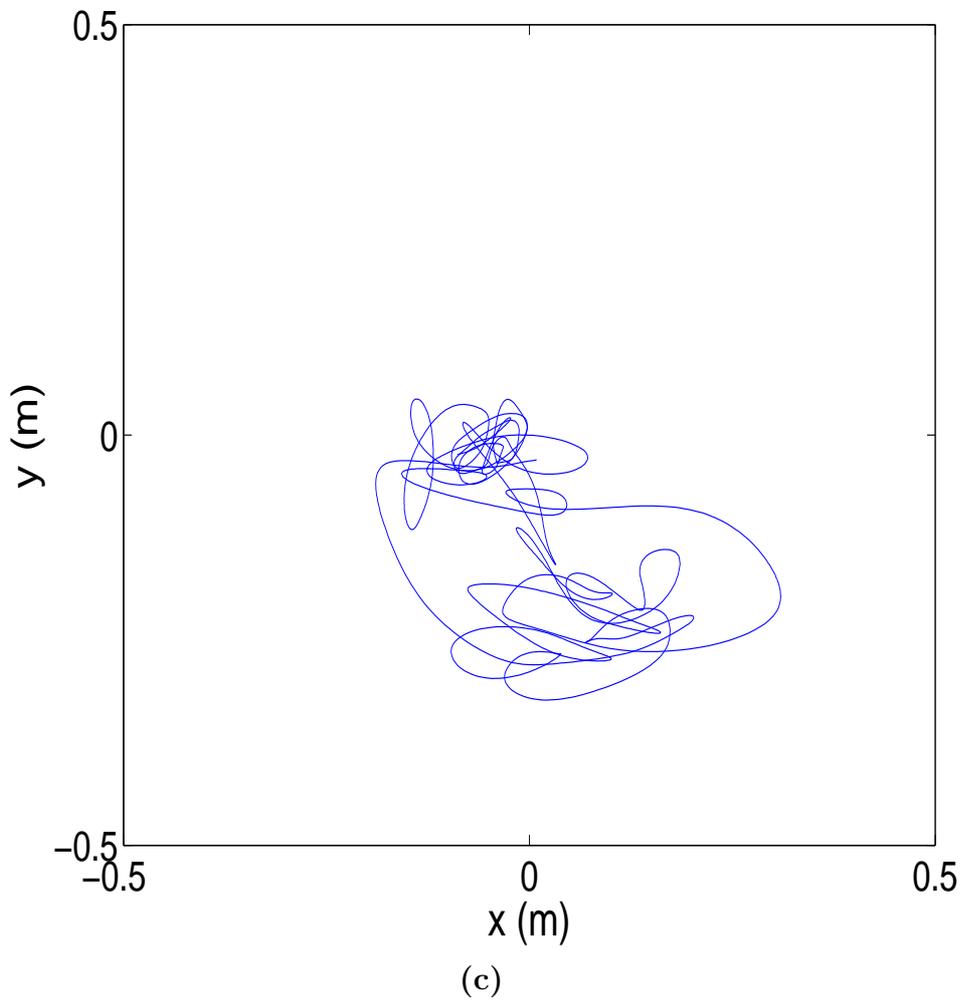
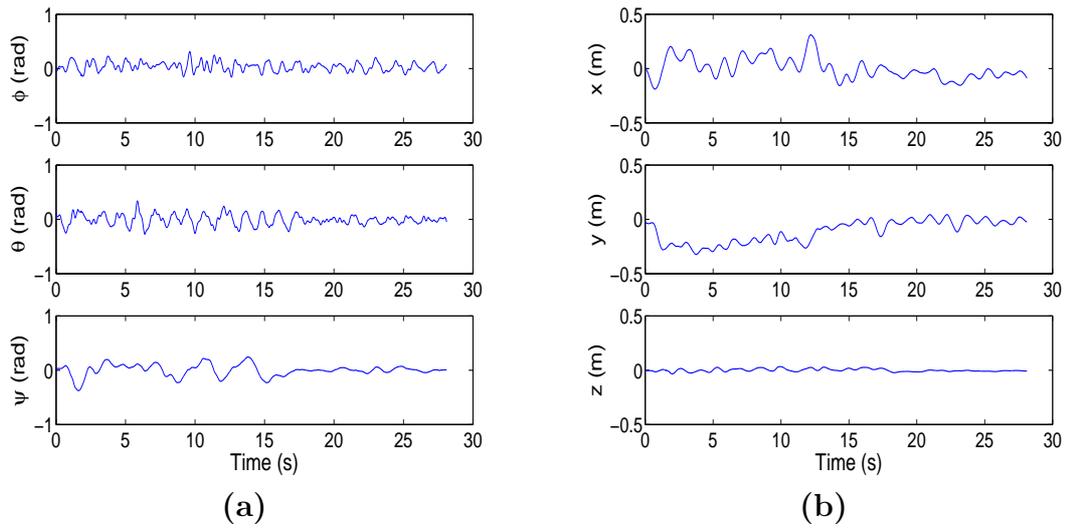


Figure 4.23: Results of flight test with position control and a 2 m/s wind in the y -direction, showing (a) vehicle attitudes, (b) translation, and (c) flight path.

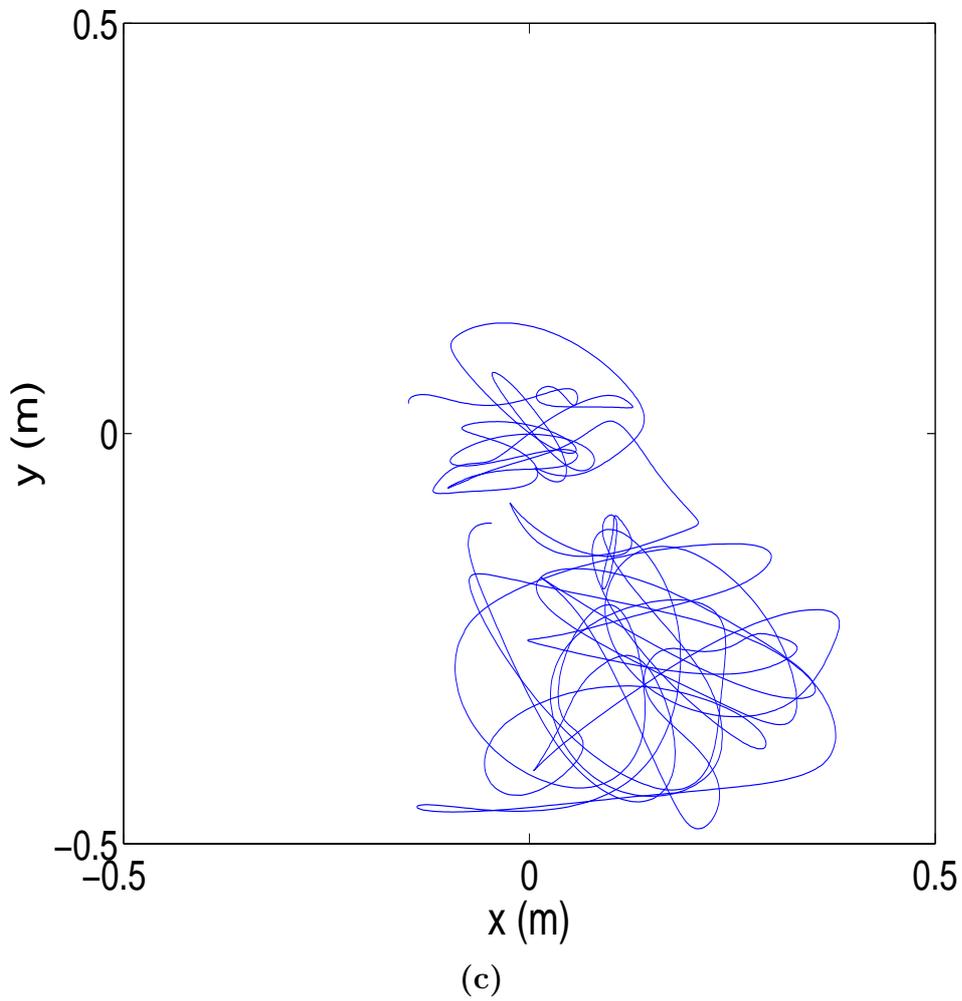
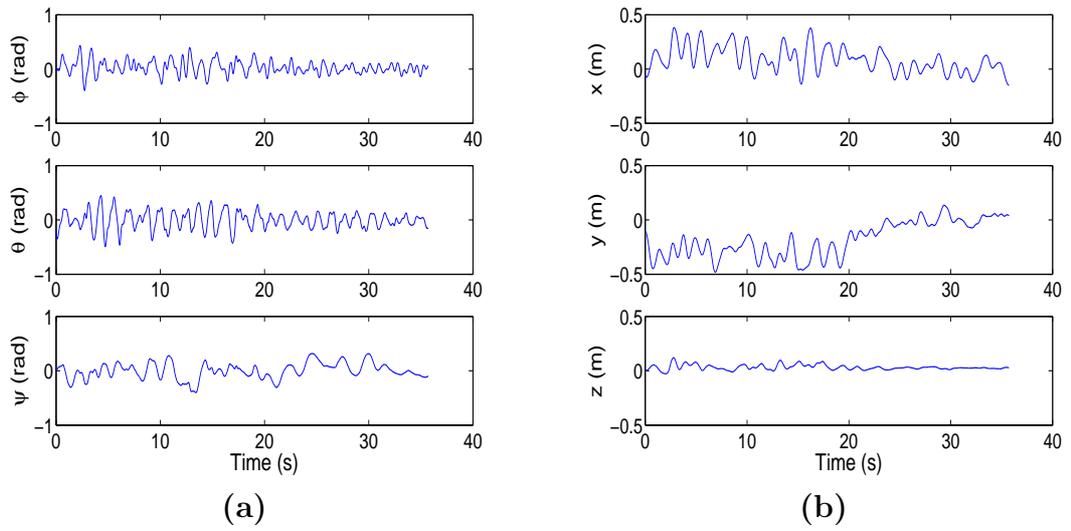


Figure 4.24: Results of flight test with position control, \mathbf{K}_2 controller, and a 2 m/s wind in the y-direction, showing (a) vehicle attitudes, (b) translation, and (c) flight path.

4.3 Concluding Remarks

Several conclusions were drawn from the quadrotor's performance in flight testing. First, the vehicle was capable of maintaining stable hover in both the absence and presence of disturbances. At best, the quadrotor maintained an attitude of ± 0.1 rad and its lateral position within a 0.1 m square. With a pitch impulse, the quadrotor returned to equilibrium while staying within the 0.1 m area and its aforementioned ± 0.1 rad attitude. With an unsteady wind disturbance averaging 2 m/s, the vehicle deviated from equilibrium by 0.3 m, but remained steady in its new position. In all instances, the vehicle returned to its designated position and attitude. Additionally, the quadrotor performed more accurately with an aggressive on-board controller coupled with the Vicon auto-pilot and an aggressive on-board controller. The Vicon auto-pilot provided an excellent means for flight testing. The system kept test conditions consistent between trials and ensured some repeatability. The PI controller successfully helped the quadrotor station-keep, though its altitude hold drifted because of battery voltage drop. The success of the PC-based autopilot and the quadrotor's embedded control algorithm was a testament to the utility of control formulation and tuning in simulation. Vehicle flight behavior resembled that predicted by the Simulink simulation, and the gains computed and tested required very little extra tuning to guarantee operation of the physical system. The simulation's efficacy was also an indication that the assumptions made in control algorithm and the dynamics formulation was appropriate for the quadrotor system in hover. Linearizing the nonlinear dynamics and using small angle approximations,

as well as subsequently assuming that the net rotor torques are close to zero, all contributed to simplifying the control problem while leading to a stable operating vehicle.

Chapter 5

Conclusions

The process of developing and testing the quadrotor described in the thesis demonstrated some of the trade-offs inherent in simplifying a vehicle's mechanical design. Keeping the number of moving and extraneous parts to a minimum helped ensure expedient production and reduced sources for failure. For a quadrotor, this guaranteed the necessity of computer augmented stability and required some complexity in the algorithm employed. In overcoming the challenges of quadrotor design and operation, some conclusions were drawn.

1. **Linearization of the nonlinear quadrotor attitude dynamics about level hover using perturbation method and small angle assumption produced a valid linear model for hover.**

Linearizing about zero attitude obfuscated a large amount of potentially pertinent information in the attitude dynamics. Doing so eliminated all nonlinearities introduced by gyroscopic effects and produced a linear system model that had marginally stable attitudes. As discussed in Section 4.1.1, however, simulations demonstrated that the nonlinear components of the attitude dynamics were negligible compared to the motor contribution, thus dismissal was justified.

2. **The motor dynamics dominated system behavior in hover.**

Testing in Chapter 2 revealed that these dynamics were not to be ignored, since the rise time for the motors were as long as 0.3 s. For a larger quadrotor with higher inertias, this might not be as much of an issue, but for smaller quadrotors such as the one under study, neglecting the motor dynamics could introduce instability to the system.

3. An LQR controller supplemented with a Luenberger Observer to estimate rotor states effectively stabilized the quadrotor.

Instead of adding complexity to the vehicle structure and circuitry, a Luenberger observer estimated the rotor states, thereby keeping the rotor dynamics in the system model used to compute the control gain matrix with LQR. In doing so, model uncertainty was reduced, as well as the theoretical delay that would be added in neglected the motor dynamics. Simulations demonstrated more than reasonable accuracy in the predictions. Granted, implementation of the observer on an embedded system took up a considerable amount of memory, especially if the matrix was represented as a floating point number. Saving the matrix to program memory instead of the EEPROM or SRAM avoided this issue but only worked because the system model was assumed to be LTI. The assumptions held, however, as demonstrated both in simulation and in flight testing, as the quadrotor maintained hover despite impulse and wind disturbances.

4. The simulation suitably predicted vehicle behavior and could be used to reliably tune the system, producing a flying, stable vehicle.

Using Simulink, the vehicle's behavior was observed before the control system was implemented on the physical vehicle. The simulation took into account the interactions between the various systems and assumptions, such as using a linear discrete controller in a system with nonlinear continuous dynamics, and displayed results that closely resembled those seen in flight testing. For instance, the oscillations predicted in the simulated vehicle response appeared in flight as well. Additionally, tuning the position control system using the simulation resulted in a working, stable vehicle that required minimal tuning in flight testing. In other words, a control system could be designed, tested, and tuned in the simulation, then implemented on the vehicle with positive results. During flight testing, adjustment to either the attitude control gain matrix \mathbf{K} or the position control gains was usually to improve performance, as opposed to ensuring operation.

5.1 Future Work

In designing the quadrotor, several possibilities for future pursuit have risen that deserve some attention. Since the quadrotor's dynamics in hover are dominated by motors, it may be possible to improve system performance by using high-torque motors that improve the rotor response. Alternately, reducing rotor inertia could lead to similar results, though perhaps not as dramatically as using a new set of motors. Another means to improving vehicle performance is the utilization of more accurate digital sensors; the Arduino's 10-bit resolution limits the accuracy of sensor

measurements and, coupled with the current sensors' sensitivity, may be responsible for the quadrotor's oscillatory response in hover. The microcontroller is I²C-ready, so using higher-resolution sensors is a definite possibility. With these improvements, it could be possible to increase the quadrotor's stability and make it an ideal platform for experimenting with other algorithms and payloads. For example, the payload capacity of the quadrotor could be used to test an additional sensor suite. Also, quadrotor and MAV testing could be streamlined by using the simulation to predict and tune the vehicles for use in Vicon. The simulation could be modified into a modular design to facilitate use with other systems. Alternatively, using the simulation and quadrotor dynamics, a design methodology could be developed for determining an optimal structure and controller given certain starting constraints, such as an off-the-shelf motor/rotor combination and battery. The algorithm could design a quadrotor by minimizing nonlinearities as well as motor effects by adjusting the vehicle's inertia. Visa-versa, the motors, rotors, and battery could be selected and placed based on the size and operating constraints of the vehicle. Overall, the quadrotor is an excellent test platform.

Bibliography

- [1] Abdallah, C. T., Cerone, V., Dorato, P., *Linear Quadratic Control: An Introduction*, Krieger Publishing Company, Malabar, FL, 2000.
- [2] Altug, E., Ostrowski, J., Mahony, R., Control of a Quadrotor Helicopter Using Visual Feedback, IEEE International Conference on Robotics & Automation, Washington, DC, May 2002.
- [3] "Arduino Pro Mini 328 - 5V/16MHz," Sparkfun Electronics. (<http://www.sparkfun.com/products/9218>).
- [4] Balasubramaniam, P., "Control, Computation and Information Systems," First International Conference on Logic, Information, Control and Computation, Gandhigram, India, February 25-27, 2011.
- [5] Borenstein, J., Ojeda, L., Kwanmuang, S., Heuristic Reduction of Gyro Drift in IMU-based Personnel Tracking Systems, SPIE Defense, Security + Sensing Conference, Orlando, FL, April 13-17, 2009.
- [6] Bouabdallah, S., *Design and Control of Quadrotors with Application to Autonomous Flying*, PhD Dissertation, cole Polytechnique Fdrale de Lausanne, Lausanne, Switzerland, 2007.
- [7] Bouabdallah, S., Noth, A., Siegwart, R., "PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor," IEEE/RSJ International Conference on Intelligent Robots and Systems Proceedings, September 2004.
- [8] Castillo, P., Lozano, R., and Dzul, A., "Stabilization of a Mini Rotorcraft with Four Rotors," *IEEE Control System Magazine*, December 2005, pp. 45-55.
- [9] "CrazyFlie quadrotor description," Daedalus Project, June 21, 2010. (<http://www.daedalus.nu/2010/06/crazyflie-quadcopter-description/>)
- [10] Domingues, J., "Quadrotor prototype," Master Thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, Lisboa, Portugal, 2009.
- [11] "Draganfly.com Industrial Aerial Video Systems & UAVs," Draganfly Innovations, Inc. (<http://www.draganfly.com>).
- [12] Grocholsky, B., Stump, E., and Kumar, V., "An Extensive Representation for Range-Only SLAM," International Symposium on Experimental Robotics, Rio de Janeiro, Brazil, July 7-10, 2006.

- [13] Harrington, A., *Optimal Propulsion System Design for a Micro Quadrotor*, Master's Thesis, University of Maryland, College Park, MD, 2011.
- [14] Hoffman, G., et al, The Stanford Testbed of Autonomous Rotorcraft for Multi Agent Control (STARMAC), The 23rd IEEE Digital Avionics System Conference, 2004.
- [15] Hoffmann, G., et al, "Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment," AIAA Guidance, Navigation, and Control Conference Proceedings, 2007.
- [16] Hrishikeshavan, V., *Experimental Investigation of a Shrouded Rotor Micro Air Vehicle in Hover and in Edgewise Gusts*, PhD Dissertation, University of Maryland, College Park, MD, 2011.
- [17] Hu, C., Qinghu Meng, M., Xiaoping Liu, P., "Observer-Based LQR Control of Shaping Process of Automobile Belt," 5th World Congress on Intelligent Control and Automation Proceedings, Hangzhou, China, June 15-19 2004.
- [18] Kendoul, F., et al, "Real-Time Nonlinear Embedded Control for an Autonomous Quadrotor Helicopter," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 4, July-August 2007, pp. 1049-1061.
- [19] Leishman, J., *The Breguet-Richet Quad-Rotor Helicopter of 1907*. (<http://www.glue.umd.edu/~leishman/Aero/Breguet.pdf>)
- [20] Leishman, J., *Principles of Helicopter Aerodynamics*, 2nd ed., Cambridge Aerospace Series, Cambridge University Press, New York, NY, 2006.
- [21] McMichael, J. and Francis, C. M., USAF, "Micro Air Vehicles Towards a New Dimension in Flight," *U.S. Department of Defense Weapons Systems Technology Information Analysis Center (WSTIAC) Newsletter*, Vol. 1, No. 13, January 2000. (<http://wstiac.alionscience.com/pdf/Vol1Num1.pdf>)
- [22] Miller, D., *Open Loop System Identification of a Micro Quadrotor Helicopter from Closed Loop Data*, Master's Thesis, University of Maryland, College Park, MD, 2011.
- [23] Niculescu, S., Fu, M., Li, H., "Delay-Dependent Closed-Loop Stability of Linear Systems with Input Delay: An LMI Approach," 36th IEEE Conference on Decision and Control Proceedings, San Diego, 1997.
- [24] Nise, N., *Control Systems Engineering*, 5th ed., John Wiley & Sons, Inc., Hoboken, NJ, 2008.

- [25] Patrick, B., *Optic Flow Based Station-Keeping and Wind Rejection For Small Flying Vehicles*, Master's Thesis, University of Maryland, College Park, MD, 2010.
- [26] Pereira, J., Bawek, D., Westfall, S., Design and Development of a Quad-Shrouded-Rotor Micro Air Vehicle, AHS 65th Annual Forum, Grapevine, TX, May 27-29, 2009.
- [27] Sanner, R., "Linear System Dynamics," University of Maryland, Fall 2008, Lecture.
- [28] Skogestad, S. and Postlethwaite, I., "Multivariable Feedback Control: Analysis and Design," John Wiley & Sons, Ltd., Chichester, UK, 1996.
- [29] Spooner, S., ed, "The Oehmichen Helicopter," *Flight*, Vol. 13, No. 9, March 3, 1921, p. 160.
- [30] Welch, G., and Bishop, G., An Introduction to the Kalman Filter, Department of Computer Science, University of North Carolina, Chapel Hill, NC, 2006.
- [31] Weng, K., and Mohamad, S., Design and Control of a Quad-Rotor Flying Robot For Aerial Surveillance, 4th IEEE Student Conference on Research and Development, 2006.