3-9-2009

Model-Based Control Using Model and Mechanization Fusion Techniques for Image-Aided Navigation

Constance D. Hendrix

Model-Based Control using Model and Mechanization Fusion Techniques for Image-Aided Navigation

THESIS

Constance D. Hendrix, Captain, USAF

AFIT/GE/ENG/09-19

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GE/ENG/09-19

Model-Based Control using Model and Mechanization Fusion Techniques for Image-Aided Navigation

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Constance D. Hendrix, BSEE, MBA

Captain, USAF

March 2009

AFIT/GE/ENG/09-19

Model-Based Control using Model and Mechanization Fusion Techniques for Image-Aided Navigation

Constance D. Hendrix, BSEE, MBA
Captain, USAF

Approved:

Lt Col Michael J. Veth, PhD (Chairman) — Date: 9 MAR 09

Dr. John F. Raquet (Member) — Date: 9 MAR 09

Lt Col Gregory J. Toussaint, PhD (Member) — Date: 06 MAR 2009

AFIT/GE/ENG/09-19

Abstract

Unmanned aerial vehicles are no longer used for just reconnaissance. Current requirements call for smaller autonomous vehicles that replace the human in high-risk activities. Many times these activities are performed in GPS-degraded environments. Without GPS providing today's most accurate navigation solution, autonomous navigation in tight areas is more difficult. Today, image-aided navigation is used and other methods are explored to more accurately navigate in such areas (e.g., indoors). This thesis explores the use of inertial measurements and navigation solution updates using cameras with a model-based Linear Quadratic Gaussian controller. To demonstrate the methods behind this research, the controller will provide inputs to a micro-sized helicopter that allows the vehicle to maintain hover.

A new method for obtaining a more accurate navigation solution was devised, originating from the following basic setup. To begin, a nonlinear system model was identified for a micro-sized, commercial, off-the-shelf helicopter. This model was verified, then linearized about the hover condition to construct an Linear Quadratic Regulator (LQR). The state error estimates, provided by an Unscented Kalman Filter using simulated image measurement updates, are used to update the navigation solution provided by inertial measurement sensors using strapdown mechanization equations. The navigation solution is used with a reference signal to determine the position and heading error. This error, along with other states, is fed to the LQR, which controls the helicopter. Research revealed that by combining the navigation solution from the INS mechanization block with a model-based navigation solution, and combining the INS error model and system model during the time propagation in the UKF, the navigation solution error decreases by 20%. The equations used for this modification stem from state and covariance combination methods utilized in the Federated Kalman Filter.

Acknowledgements

First and foremost, I owe a large debt of gratitude to my husband for all the understanding, encouragement, and support while pursuing my EE masters degree. Secondly, I like to thank Lt Col Michael Veth, my thesis advisor. Thanks for stating the obvious, when it wasn't obvious to me, and for the infectious enthusiasm for micro-air vehicles. In addition, there were several colleagues and professors that I'd like to thank for being there to talk out ideas when stuck in that thesis rut; they are 1lt Jeff Gray, Capt Jason Bingham, Capt Mike Ciampa, Lt Col Gregory Toussiant, Dr. Meir Pachter, and Dr. John Racquet. Furthermore, Maj. Michael Stepaniak's guidance on modeling quadrotors was extremely helpful and much appreciated in effort to start future work. Last, but certainly not least, I'd like to thank 1Lt Ryan Carr from AFRL/RB. 1Lt Carr was an instrumental part of hardware testing the LQR controller. I worked with him as the first customer to the MAV IFF in integration and testing of a MAV controller for autonomous operation. AFRL/RB is doing great work by making it possible to confirm a portion of the design before the final configuration was ready.

Constance D. Hendrix

Table of Contents

Page

Page

List of Figures

List of Tables

List of Symbols

Symbol		Page
P	State covariance matrix	15
K	Kalman Gain	17
R	Measurement noise covariance matrix	17
P_{res}	Residual covariance	17
h	Output nonlinear function	19
L	Number of states	20
λ	UKF scaling parameter	20
α	UKF spread parameter	20
β	UKF tuning parameter	20
κ	UKF shape parameter	20
W_{0m}	UKF nominal state estimate weight	21
W_{0c}	UKF nominal state covariance weight	21
W_{ukf}	UKF non-nominal sigma point weight	21
$\bar{x}$	Nominal state	21
χ	Sigma points	21
$P_{\hat{z}\hat{z}}$	Measurement estimate covariance	22
P_{xz}	UKF Cross correlation matrix	23
$P_{\hat{z}\hat{z}}$	UKF Cross correlation matrix	23
J	Quadratic cost function	24
X	State weighting matrix	24
U	Input weighting matrix	24
X_f	Accuracy weighting matrix	24
G_c	LQG optimal feedback controller gain	25
u^*	LQG optimal control output	25
K_e	Motor electro-magnetic force constant	35
R_m	Motor resistance	35
L	Motor inductance	35
K_t	Motor torque constant	35

Symbol		Page
b	Viscous damping ratio	35
$M_{inertia}$	Moment of inertia	36
I	Motor armature current	36
V	Motor voltage	36
R	Motor resistance	36
ω	Motor angular velocity	36
L	Motor inductance	36
τ_{load}	Drag torque	36
r_{blade}	Blade length	38
m	mass	38
u_1	Throttle	39
u_2	Rudder	39
u_3	Pitch command	39
u_4	Roll command	39
FB	Flybar	39
Λ	Diagonal matrix	47
Δv^b	Change in velocity	53
$\Delta\theta^b_{ib}$	Change in attitude	53
P_{wgs}	Initial WGS 84 MAV location	54
P^e	Initial ECEF MAV location	55
a^n	MAV acceleration, nav frame	55
v^n	MAV velocity, nav frame	55
p^n	MAV position, nav frame	55
θ^n_{nb}	MAV attitude, nav frame	55
ω^b_{nb}	MAV attitude rates	55
a^b	Accelerometer bias estimate	55
b^b	Gyroscope bias estimate	55
θ^b_{nb}	Current MAV attitude, Euler angle vector	55

Symbol		Page
Ω_{ie}^{n}	Skew symmetric of Earth's rate, navigation frame	55
w_{a}^{b}	Accelerometer noise	56
w_{b}^{b}	Gyroscope noise	56
$w_{a\_bias}^{b}$	Accelerometer bias	56
$w_{b\_bias}^{b}$	Gyroscope bias	56
$\delta\hat{x}$	Error state estimate	59
Δv^{b}	Differential velocity, raw INS	87
$\Delta\theta_{ib}^{b}$	Differential attitude, raw INS	87
Θ	Root-mean-square	89
S	Input noise intensity	90
I	Raw INS noise intensity	90

List of Abbreviations

Abbreviation		Page
UAV	Unmanned Aerial Vehicle	1
DoD	Department of Defense	1
MAV	Micro-sized Aerial Vehicle	1
GPS	Global Positioning System	1
LQG	Linear Quadratic Gaussian	2
VTOL	Vertical Take-Off and Landing	2
EKF	Extended Kalman filter	2
UKF	Unscented Kalman filter	3
AFIT	Air Force Institute of Technology	3
INS	Inertial Navigation System	3
6DOF	Six Degrees-Of-Freedom	3
MIT	Massachusetts Institute of Technology	4
ECEF	Earth-Centered, Earth-Fixed	6
NED	North East Down	7
DCM	Direction Cosine Matrix	9
WGS	World Geodetic System	9
pdf	Probability density function	15
SPKF	Sigma Point Kalman Filter	19
LQR	Linear Quadratic Regulator	25
SLAM	Simultaneous Localization and Mapping	27
SIFT	Scale Invariant Feature Transform	27
AFRL	Air Force Research Laboratory	28
IFF	Indoor Flight Facility	28
PPM	Pulse-Position Modulation	28
ANT	Advanced Navigation Technology	31
FKF	Federated Kalman Filter	58

Abbreviation		Page
CSV	Comma-Separated-Variable	65
VI	Virtual Instrument	65
DC	Direct Current	76
RSS	Root-Sum-Square	89
RMS	Root-Mean-Square	89
PI	Proportional-plus-Integral	102
PWM	Pulse-Width Modulation	114

Model-Based Control using Model and Mechanization Fusion Techniques for Image-Aided Navigation

I. Introduction

The successful demonstration of the RQ-1 Predator Unmanned Aerial Vehicle (UAV) introduced a new way to conduct warfare. These relatively low-cost drones were initially used to perform reconnaissance missions, loitering for up to 24 hours [1]. The operators of the system controlled the vehicle from a ground control station several miles from the area of interest. This stand-off capability allowed the mission to be performed in high-risk areas of operation without endangering the lives of onboard pilots or losing high-cost aircraft. The RQ-1's role was quickly expanded to include offensive air-to-ground engagement using Hellfire missiles. With the configuration of missiles, the designation changes to the MQ-1 Predator. There is no doubt that the Predator was the first-mover in the world of UAVs; but, with first-movers, come fast-followers. UAVs are now a viable consideration for today's military to fill current capability gaps (e.g., mine detection; signals intelligence, precision target designation, etc.) [23]. In the DoD's *Unmanned Systems Roadmap 2007-2032*, many implementations are being considered in effort to "invest in new equipment, technology, and platforms for the forces, including advanced combat capabilities" in terms of UAVs [23]. Not only is the mission of the UAV being expanded, but new operational environments are also being explored. One example of a new operational environment is the urban environment. Many unmanned systems are currently in development to operate within this challenging environment; these systems include ground robots and micro-sized aerial vehicles (MAVs). The urban environment poses a unique challenge for navigation in that the most accurate navigation solution to date, Global Positioning System (GPS), is often degraded or denied, especially inside

[1] http://www.af.mil/factsheets/factsheet.asp?fsID=122

buildings or underground. Alternatives to GPS are being explored to make the use of MAV's in the urban environment a reality.

1.1 Purpose

The purpose of this thesis is to develop a model-based Linear Quadratic Gaussian (LQG) controller design to control a MAV when GPS is denied. Because this design is model-based, a system would need to be chosen before a model is developed. Logically, the MAV would need to stop, look around, and change directions in a worst-case setting which heavily constricts movement due to walls, furniture, etc. The system chosen to meet this requirement is a vertical take-off and landing (VTOL) aircraft, such as a helicopter. Furthermore, an inertial navigation system and eventually cameras will be used to calculate a navigation solution in the absence of GPS. The image processing portion of this effort will not be undertaken, and is assumed to be available for integration at a later date. The innovative portion of this design is to create and test a method for combining system and inertial models to provide a more accurate solution of the vehicle's position and heading. This design, if successful, could be leveraged to help meet future requirements of today's warfighters.

1.2 Previous Work

Designing MAV controllers while considering the complexities of the urban environment is not cutting-edge work. Many papers have been published featuring the use of inertial and/or vision navigation of a micro-sized helicopter [7] [20] [19]. One example is detailed in an effort conducted by Allen Wu, Eric Johnson, and Alison Proctor from the Georgia Institute of Technology [1].

Wu et al. argued that in 2005 researchers had only "begun seriously investigating the application of vision sensors in inertial navigation" [1]. They investigated using an Extended Kalman Filter (EKF) to process measurement updates derived from image processing to correct for accelerometer and gyroscope drift that is inherent in inertial navigation systems. This corrected solution would serve as the

navigation solution for the controller. The crux of their design was to demonstrate the performance of the vision-based EKF through simulations using GPS readings as as truth. Although, the paper proved the design to be successful, the EKF is not the only state estimation method used for nonlinear filtering. A more recent Kalman filtering method has been devised that captures higher-order nonlinearities in a nonliner model. This effectiveness of this filter, unscented Kalman filter (UKF), has been tested many times against the EKF and repeatedly shown to provide superior performance. Two such studies were performed by First Lieutenant Sedat Ebcin from AFIT, who used the UKF with vision-aided inertial navigation [3], and Rudolph van der Merwe and Eric A. Wan from Oregon Health & Science University, whose work integrated the UKF study with an implementation on a MAV [20].

First Lieutenant Sedat Ebcin, conducted research on the UKF and its use with a tightly-coupled, image-aided, inertial navigation system (INS) [3]. His accomplishment was a follow-on to an earlier AFIT effort to fuse image and inertial navigation information using an EKF. Image measurements depicting range information to selective features were used in a feedback configuration to provide state estimates in order to correct the INS trajectory. The simulation was performed using a Monte Carlo analysis approach, followed-on by an experiment using binocular vision to calculate a trajectory inside a building. The results concluded that the UKF addressed the "destabilizing effects of linearization errors" found to be characteristic of the EKF, thus provided a notable improvement in the estimate of the navigation states during simulation and test [3]. Likewise, Rudolph van der Merwe and Eric A. Wan investigated the deficiencies found using the the EKF, which is considered by some the industry standard for nonlinear filtering, with integrated navigation system platforms [20]. The thrust of their work was to prove the UKF's, otherwise known as a Sigma-Point Kalman filter, superior performance in state estimation to the EKF in navigation by loosely-coupling a GPS receiver with an INS. Their stated points of query were limited to: 1) six degrees-of-freedom (6DOF) accuracy, 2) GPS latency resolution (to test a "sensor latency compensation technique"), and 3) closed-loop

control. When constructing the algorithm for simulation, a reduced nonlinear mathematical model was used, specific to the hardware to be later tested. Two types of UKFs were analyzed, the square-root UKF and the square-root central difference Kalman Filter. The square root approach was used to provide numerical stability to the calculations which is known to be problematic due to rounding-off, typical of most computer systems. The results of their efforts support their hypothesis of the UKF's superiority. Their conclusions were supported by simulation and hardware experimentation using an instrumented X-Cell-90 helicopter, created by MIT's *Laboratory for Information and Decision Systems*. Not only have many studies been performed comparing the UKF to the EKF using a micro-sized helicopter, LQG control has also been utilized in controlling the same type of vehicle.

Zhe Jiang, Jianda Han, Yuechao Wang, and Qi Song, from the Chinese Academy of Sciences, developed an LQG controller using a UKF for state estimation to simulate a helicopter maintaining a hover in a feedback configuration [9]. Their effort is one example of a simple design using LQG control techniques to control a highly nonlinear system. Another example is an effort made by John C. Morris, Michiel van Nieuwstadt, and Pascale Bendotti, from Caltech [10]. This group designed an LQG controller based on a nonlinear helicopter model to maintain hover. The design followed the basic steps for LQG control; however, mostly focused on the system identification process. One concern stated in the paper centered around the high degree of uncertainty in the yaw axis performance of their helicopter. The helicopter model used considers a tail rotor to control the yaw motion. Their paper suggests that an asymmetry in accusation could be the cause, and this effect was not captured in the system model. Plainly stated, "it is much easier to yaw in the direction opposite to the rotation direction of the main rotor" [10]. As a result, they reiterated the importance of modeling the dynamics of the system when implementing a model-based controller.

Numerous lessons learned can be gleaned from the many references previously mentioned. The concepts of LQG control of a hovering vehicle using INS and a UKF

with vision updates have all been previously accomplished. The concept of combining models to provide a better navigation solution for flight control is an area of control that is seemingly untouched. The work performed in this paper will incorporate information garnered from these previous works, along with others, to implement a new strategy in model-based control. This thesis will cover background supporting the methodology, the methodology and design, simulation and hardware results analysis, and conclusions. The first step is to understand the concepts behind the design.

II. Background

THIS chapter provides an introduction to the ideas and concepts behind the research and design effort presented in this paper. The objective of this thesis is to build a model-based LQG controller for a micro-sized helicopter. The controller will utilize a system model/inertial navigation integration method to help the helicopter maintain a hover condition. The background supporting the design is introduced in a way that each new concept builds upon the previous. The concepts to be covered in order of occurrence are: coordinate and transformation systems, Inertial Navigation Systems, Kalman filtering techniques, Linear Quadratic Gaussian controllers, Vision-Aided Navigation, and the Vicon System.

2.1 Coordinate Systems and Transformations

In navigation, the chosen coordinate reference determines the way position information is calculated and conveyed. Whether it be degrees in latitude and longitude or height in kilometers above a defined ellipsoid, having a reference standard between systems reduces errors in navigation. The following coordinate frames of reference are commonly used in navigation: inertial, Earth, navigation, and body frame [30]. An inertial coordinate system is defined as a non-accelerating, non-rotating coordinate system [32]. In navigation, the Earth-Centered inertial coordinate system represents a coordinate system which the axes are pointed to fixed stars. For navigation with respect to the Earth, the Earth-centered, Earth-fixed is the more logical frame of reference to be used.

2.1.1 Earth-centered, Earth-fixed. The Earth-centered, Earth-fixed (ECEF) frame of reference, otherwise known as the Earth frame, is a rotating, right-hand coordinate system [30]. This frame of reference uses the coordinates x, y, and z with the origin located at the Earth's center of mass. The ECEF frame rotates on the z-axis at the same rate as the Earth's rotation, allowing the x-axis to be fixed at the crossing of the prime meridian and the equator. The y-axis also protrudes out the equator, orthogonal to the x-axis and the z-axis, as displayed in Figure 2.1.

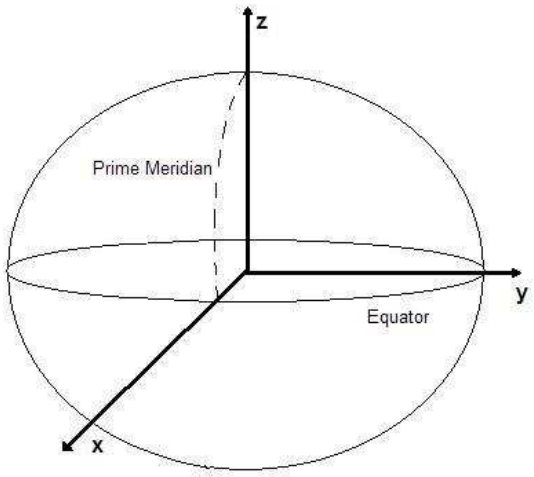


Figure 2.1: Earth-centered, Earth-fixed Coordinate System. A right-hand, Earth-centered, rotating coordinate system used in navigation.

A point referenced in this coordinate system is expressed in meters or kilometers. Applications using the ECEF frame include, but are not limited to, navigation over short distances, such as missile navigation, and over long distances, such as navigation with GPS. Another frame of reference used extensively for navigating is the navigation frame [30].

2.1.2 Navigation. The local geographic navigation frame, otherwise known as NED (North East Down) frame, is a rotating frame of reference with its origin located at the navigation system (see Figure 2.2) [30]. Its positive x-axis points to true north, positive y-axis points east, and positive z-axis points down. The x-y plane is always tangent to the Earth's surface. This frame of reference is a moving plane used extensively with Inertial Navigation Systems [25]. To determine the position in the NED frame, the raw inertial measurement data is typically resolved in the body frame.

2.1.3 Body. The body frame is a rotating frame of reference with a defined point of origin located somewhere near the body's center of mass. In Figure 2.3, the body is an aircraft, with the positive x-axis pointing out the nose of the aircraft, the

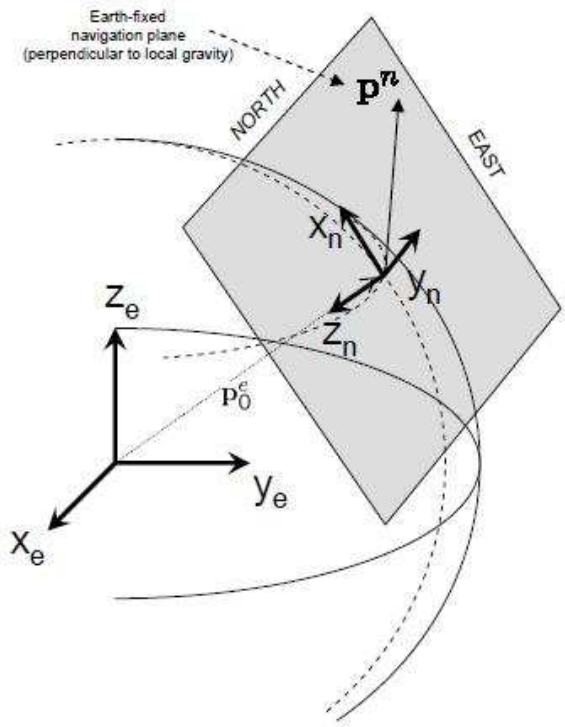


Figure 2.2: Local Geographic Navigation Frame in Relation to ECEF [33]. The navigation frame is a right-hand, rotating coordinate system, with the xy-plane tangent to the Earth's surface and centered at the end of vector p^n.

positive y-axis pointing out the right wing, and the positive z-axis pointing out the bottom, each axis being orthogonal to the others. This frame of reference is used to quantify roll, pitch, and yaw to be used for attitude calculations. Roll is defined as a rotation of the rigid body about the x-axis, while pitch and yaw describes the rotations about the y and z axes, respectively. The angles of rotation, also known as Euler angles, under particular circumstances are used to describe this change in attitude. The Euler angles, ϕ , θ, and ψ, represent roll, pitch, and yaw angles, respectively.

The Euler angles, defined by the order of rotation, are used to transform position vectors from one reference frame to another. The order of rotation, called (3,2,1), is used in aircraft navigation [26]. It requires a rotation about the z-axis first, then a

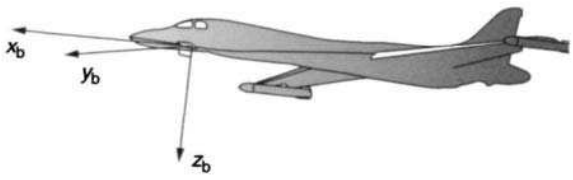


Figure 2.3: Body Frame of Reference [30]. A rotating frame of reference with the center of mass of a rigid body depicting the origin, commonly used in inertial navigation.

rotation about the y-axis, then finally a rotation about the x-axis. This sequence of events defines the relationship between the navigation frame and the body frame and is required to accurately produce a body-to-nav DCM (direction cosine matrix). A DCM is a linear transformation used to convert position information from one frame of reference to another. The body-to-nav DCM used to transform a position vector from the body frame of reference to the navigation frame of reference is shown in Equation (2.1) [30].

$$C_b^n = \begin{bmatrix} cos\psi cos\theta & cos\psi sin\theta sin\phi - sin\psi cos\phi & cos\psi sin\theta cos\phi + sin\psi sin\phi \\ sin\psi cos\theta & sin\psi sin\theta sin\phi + cos\psi cos\phi & sin\psi sin\theta cos\phi - cos\psi sin\theta \\ -sin\theta & cos\theta sin\phi & cos\theta cos\phi \end{bmatrix} \quad (2.1)$$

Coordinate reference frames are not the only consideration when defining standards in navigation. Reference ellipsoids and gravity models are referenced also when calculating a navigation solution. A common reference system helps eliminate errors when computing relative position. The most common standard used in the United States is the World Geodetic System (WGS) 84.

2.1.4 WGS 84. The WGS 84 is a standard by which a gravity model, reference ellipsoid, and navigation coordinate system is defined. The coordinate frame employs coordinates, λ, l, h, representing geodetic latitude, geographic longitude, and height above the WGS 84 ellipsoid, respectively. These coordinates are based on a

reference defined by an ellipsoid depicting the approximate shape of the earth, the Earth's axis of rotation, and specific points of longitude. For latitude, 0° is referenced at the equator, while ± 90° latitudes indicate locations approximate to the north and south poles. For longitude, the earth is divided into slices from 0° to 360°, starting and ending at the prime meridian located in Greenwich, England. This reference system was developed for use with the GPS and eventually became a "de facto" international standard [21].

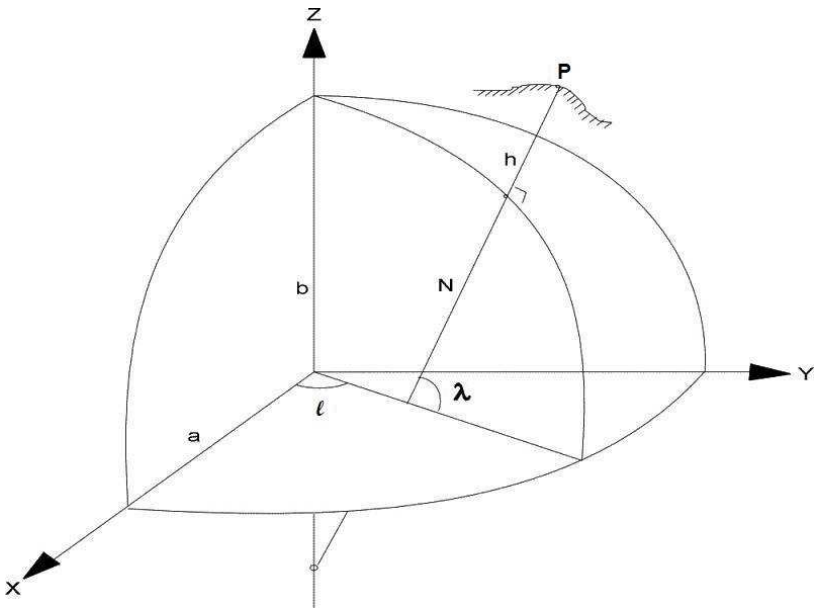


Figure 2.4: Ellipsoidal Coordinates with λ = Geodetic Latitude and l = longitude[1]. Describes the variables used with WGS 84 in defining ECEF coordinates, depicted by position "P".

The relationship between the WGS 84 and the ECEF coordinate frames is shown pictorially in Figure 2.4, and depicted numerically through Equations (2.2a) - (2.2e) in terms of the prime vertical of curvature (N), h, flattening (f), ellipsoid equatorial radius (a), ellipsoid polar radius (b), latitude (λ), and longitude (l) [32].

[1]http://www.gloposys.de/Global%20Positioning%20System%20CD/Abbildungen/Abb-3-11.jpg

$$x = (N + h)cos(\lambda)cos\ (l) \tag{2.2a}$$

$$y = (N + h)cos(\lambda)sin\ (l) \tag{2.2b}$$

$$z = [1 - (f(2 - f))N + h]sin(\lambda) \tag{2.2c}$$

$$N = \frac{a}{\sqrt{1 - (f(2 - f))^2 sin^2(\lambda)}} \tag{2.2d}$$

$$f = \frac{a - b}{a} \tag{2.2e}$$

The DCM used to transform vectors in the ECEF frame to the Navigation frame using WGS 84 coordinates is shown in Equation (2.3) [32].

$$C_e^n = \begin{bmatrix} -sin\lambda\ cos\, l & -sin\lambda\ sin\, l & cos\lambda \\ -sin\, l & cos\, l & 0 \\ -cos\lambda\ cos\, l & -cos\lambda\ sin\, l & -sin\lambda \end{bmatrix} \tag{2.3}$$

These DCMs can also be used together to perform additional coordinate systems tranformations. For example, to convert from body frame of reference to the ECEF frame of reference, a simple matrix transform and multiplication using previously defined DCMs are required: $C_b^e = C_e^{n\,T} C_b^n$. To recap, coordinate reference frames are useful in understanding the various ways position can be calculated, transmitted, and conveyed in different navigation systems. In this research the primary instrument used for producing the navigation solution is the Inertial Navigation System.

2.2 Inertial Navigation Systems

Inertial Navigation Systems (INS) are instrumental in navigation today. The concept behind these systems is to keep track of position, velocity, and attitude from only knowledge of the starting point by measuring translation acceleration and rate of change in attitude. Unlike other popular navigation systems, INS systems are completely passive and do not require external signals; therefore, the system cannot

be jammed or spoofed. INS systems are divided into two categories: gimbaled and strapdown. Strapdown systems constitute over 90% of INS systems currently used and will be the focus of this paper [26]. These systems use software to calculate the navigation solution using sensory inputs. Their mechanizations define how these solutions are calculated, and are tailored dependent upon their intended use. The strapdown mechanization used in this thesis is the local geographic navigation frame mechanization, as shown in Figure 2.2 [30]. The sensor package used consists of six rigidly mounted sensors: three gyroscopes, placed orthogonally with respect to each other and aligned with the the body's x, y, and z-axes, sensing rotation rates in pitch, roll, and yaw; and three accelerometers, also mounted in an orthogonal triad, sensing motion in the x, y, and z-directions of the body. To summarize, accelerometers are used to determine velocity and position, while the gyroscopes are used to determine attitude and attitude rates. With this in mind, several basic tasks are executed within the system to derive the required information for navigation. The sequence of these tasks are interpreted from Figure 2.5, and explained in the following text.

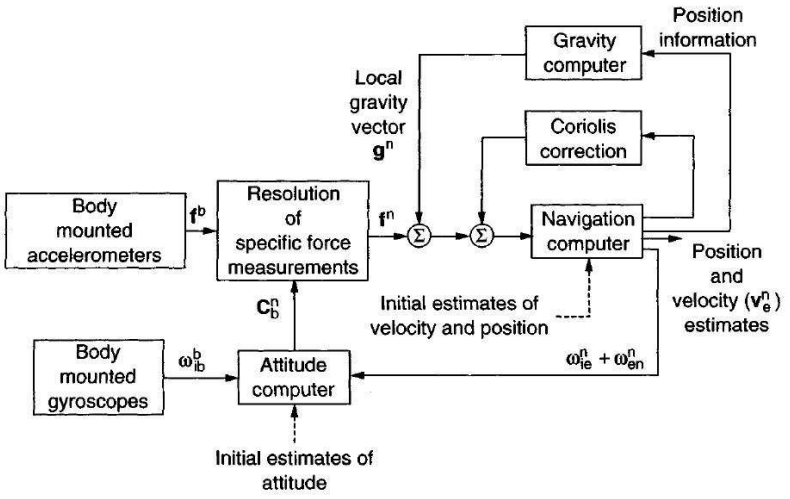


Figure 2.5: Strapdown INS - Local Geographic Navigation Frame Mechanization [30]. Used to calculate position, velocity, and attitude in Strapdown INS Systems used for long distance travel.

The body-mounted accelerometers shown in Figure 2.5 measure a specific force, designated f^b. The specific force accounts for the 3D acceleration, to include gravity,

and is easily translated to the navigation frame or the Earth frame of reference using transformation matrices discussed in the previous section. These matrices are formulated from aircraft's attitude produced by the body-mounted gyroscopes, which is later discussed. Once f^n or f^e is determined, acceleration in the corresponding frame of reference can be easily calculated by adding the gravity component (g), derived from a gravity model, in the commonly known "navigation equation", referenced in Equation (2.4) [30]:

$$a^n = C_b^n f^b + g^n \tag{2.4}$$

This equation is the basis for inertial navigation. The DCM used to convert f_b to the navigation frame of reference is calculated by knowing the body's attitude rates. These body rates (ω_{ib}^b) come from the three gyroscopes shown in Figure 2.5 [30]. First, ω_{ib}^b is used to calculate the body rate with respect to the navigation frame [30]:

$$\omega_{nb}^b = \omega_{ib}^b - C_n^b \left[\omega_{ie}^n + \omega_{en}^n\right] \tag{2.5}$$

The components ω_{ie}^n and ω_{en}^n constitute the navigation frame rates, and are referenced, along with the other corrections, in Table 2.1. This equation can be manipulated into an equation which describes the propagation of the DCM, where Ω_{nb}^b is the skew symmetric form of the body rate in the navigation frame ω_{nb}^b [30]:

$$\dot{C}_b^n = C_b^n \Omega_{nb}^b \tag{2.6}$$

This particular calculation captures attitude corrections; however, the corrections to the acceleration derived from the navigation equation, Equation (2.4), still need to be identified. These corrections come from the gravity computer and the coriolis correction blocks in Figure 2.5, and are summarized in Table 2.1.

Table 2.1: Inputs for Error Correction in Navigation [30]. These inputs are used to correct the navigation state for systems using local geographic frame mechanization.

Input	Description	Type of Correction
ω_{ie}^{n}	"The Earth's rate with respect to the inertial frame" in the nav frame - used to subtract out the effects of the Earth's rotation	Attitude
ω_{en}^{n}	"The turn rate of the navigation frame with respect to the Earth" in the nav frame - provides an angular correction of the nav frame due to the curvature of the Earth	Attitude
$\omega_{ie} \times (\omega_{ie} \times r)$	Centripetal acceleration - used to define the local gravity vector by adjusting the local mass attraction (gravity) for its effects	Acceleration
$(2\omega_{ie}^{n} + \omega_{en}^{n}) \times v_{e}^{n}$	The coriolis acceleration, where v_{e}^{n} is the ground speed in the nav frame - approximates an error caused by an effect of a body moving "over the surface (air or water) of a rotating earth"	Acceleration

To summarize, the final solution with corrections can be expressed in Equation (2.7). The term f^{b} represents the specific force vector provided by the accelerometers. This vector is translated to the navigation frame using the information provided by the gyroscopes. Several previously mentioned corrections are made, then finally the local gravity vector is added to the acceleration in the down direction.

$$a_{e}^{n} = C_{b}^{n} f_{b} - [2\omega_{ie}^{n} + \omega_{en}^{n}] \times v_{e}^{n} + g - \omega_{ie} \times [\omega_{ie} \times r] \tag{2.7}$$

With this final acceleration term, a_{e}^{n}, the position can be calculated by integrating twice then adding the required input of initial estimates. While understanding strapdown INS mechanization is essential when integrating an INS system into a navigation controller, state estimation is also a critical component in reducing errors due to stochastic inputs. The type of state estimator to be used with the controller in this effort is the Kalman filter.

2.3 The Kalman Filtering Techniques

Kalman Filters are optimal, recursive state estimators used to estimate system states in the presence of random inputs. The most common examples of random signals, also termed as non-deterministic signals, are disturbances and noise. The traditional Kalman Filter accomplishes this task through the propagation and update of a linear stochastic dynamics model. This model can be represented by a differential equation in terms of the system states (x), inputs (u), and process noise (w) as shown in Equation (2.8), where A, B and G are matrices describing the relationship of these terms. A difference equation, Equation (2.9), can be used in place of the differential equation for implementation in discrete-time using equations found in Table 2.2, where Δt is the sample time step and Q is the process noise covariance matrix. In the difference equation, Φ is identified as the state transition matrix, and B_d is the input transition matrix. These matrices are used to relate the current state, input, and noise to the state at the next time step, t_{i+1}. Also, with Kalman filtering, discrete measurements are taken to perform updates. These measurements (y) are expressed in terms of the states and measurement noise (v) in Equation 2.10, where H is the output transition matrix.

$$\dot{x}(t) = Ax(t) + Bu(t) + Gw(t) \tag{2.8}$$

$$x(t_{i+1}) = \Phi(t_i)x(t_i) + B_d(t_i)u(t_i) + w(t_i) \tag{2.9}$$

$$y(t_i) = Hx(t_i) + v(t_i) \tag{2.10}$$

The predicted statistics of this model constitute the mean, $\hat{x}$, and the associated uncertainty, which is captured in the covariance matrix, P. Because the traditional Kalman filter produces state estimates based on the most likely value of the state corresponding to the mode/mean of its Gaussian probability density function (pdf)

Table 2.2: Continuous to Discrete-Time Matrix Formulations [17]. These symbols and associated equations can be used to transform a differential equations to difference equations.

Symbol	Equation	Matrix
Φ	$e^{A\Delta t}$	State transition matrix
Q_d	$\int_{t_i}^{t_{i+1}} \Phi(t_{i+1},\tau)G(\tau)Q(\tau)G^T(\tau)\Phi^T(t_{i+1},\tau)d\tau$	Process noise intensity
B_d	$\int_0^{\delta t} \Phi(t_{i+1},\tau)B(\tau)d\tau$	Input matrix
G	G	Noise input matrix

derived from all past measurements and the stochastic dynamics model, it is considered an optimal estimator [17]. The state estimates are updated periodically to decrease the uncertainty due to the inevitable introduction of noise and disturbances in the system. The time period between updates is referred to as "propagation", as shown in Figure 2.6.

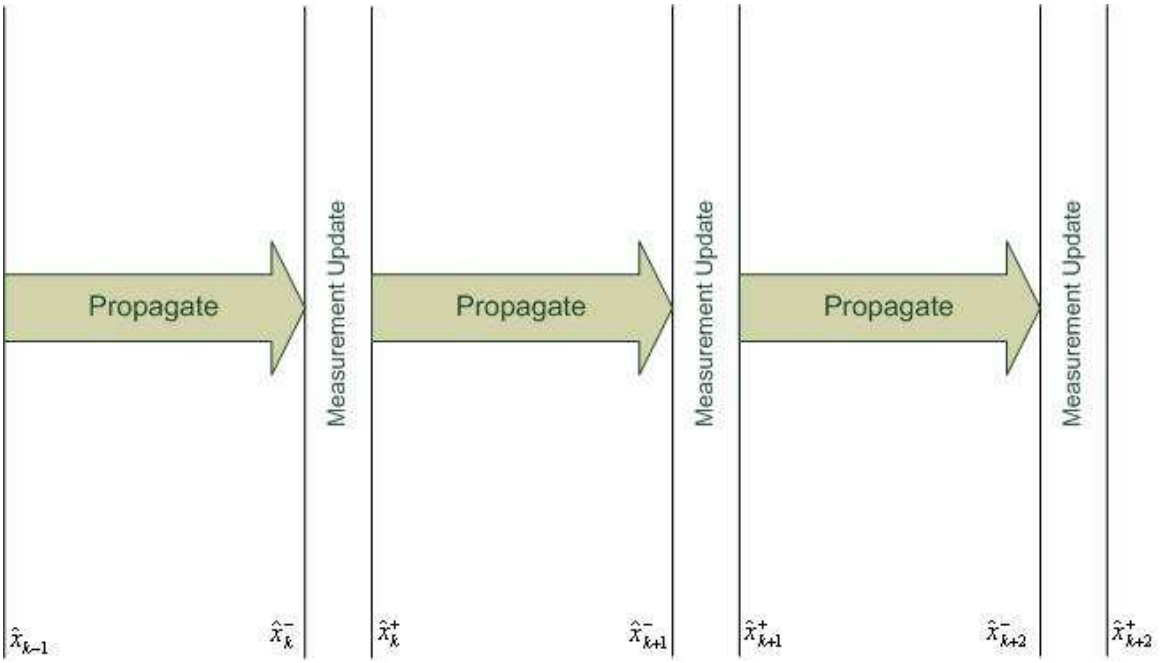


Figure 2.6: Kalman Filter Execution [32]. This iterative sequence is divided into two parts: propagation and measurement update. The minus superscript signifies the estimated state before update, and the plus signifies the estimated state after an update.

During propagation, the states are propagated forward using the system model with added uncertainty increasing with every time step. The following equations are

used to update the covariance and state estimate during propagation:

$$\hat{x}(t_i^-) = \Phi(t_{i-1})\hat{x}(t_{i-1}^+) + B_d(t_{i-1})u(t_{i-1}) \tag{2.11a}$$

$$P(t_i^-) = \Phi(t_{i-1})P(t_{i-1}^+)\Phi^T(t_{i-1}^-) + Q_d \tag{2.11b}$$

During the update, the difference between the measured (z_{meas}) and predicted values ($H\hat{x}(t_i^-)$), called the residuals, are calculated. The residual covariance is then used to formulate the Kalman gain, K, using Equation (2.12a). The Kalman gain determines the weighting of the measurement used in the calculation of the updated covariance and expected state values. The H matrix dictates the relationship between the states and the measurement, while the R matrix contains the covariance for the sensor noise [17]. The equations used to determine the Kalman gain and the residual covariance, P_{res}, are shown below:

$$K(t_i) = P(t_i^-)H^T P_{res}(t_i)^{-1} \tag{2.12a}$$

$$P_{res}(t_i) = HP(t_i^-)H^T + R \tag{2.12b}$$

The gain, output covariance matrix, and the residuals are then used to update the predicted mean ($\hat{x}$) and covariance (P):

$$\hat{x}(t_i^+) = \hat{x}(t_i^-) + K(t_i)\left[z_{meas}(t_i) - H\hat{x}(t_i^-)\right] \tag{2.13a}$$

$$P(t_i^+) = P(t_i^-) - K(t_i)HP(t_i^-) \tag{2.13b}$$

After the update, the covariance and mean propagate, thus repeating the sequence.

2.3.1 Extended Kalman Filter. The EKF is a type of Kalman filter used to account for nonlinearities in system and measurement models and can be con-

sidered the industry standard to reduce errors in nonlinear systems like the one in Equation (2.14) [15].

$$\dot{x}(t) = f(x, u, t) + Gw(t) \tag{2.14}$$

The EKF linearizes the nonlinear system and measurement models about the current operating condition. This linearization process produces a linear system model, only applicable during the next time step, which is applied to the traditional Kalman filter algorithm. The process for Extended Kalman filtering, like the traditional, is executed within two steps iteratively [15]:

1. Propagation
2. Measurement Update

Propagation. Time propagation is performed a little differently than the traditional Kalman filter. The state propagation equation, Equation (2.15), uses the integral of the nonlinear differential equation with respect to time, quantified using the current state estimate and inputs, to add to the previous state estimate.

$$x(t_i^-) = x(t_{i-1}^+) + \int_{t_{i-1}}^{t_i} f[\hat{x}(t/t_i), u(t), t]\, dt \tag{2.15}$$

The propagation of the state covariance, P, uses the equation referenced in Equation 2.11b. Before this equation can be used, the state transition matrix, Φ, must be realized. The EKF performs this task by linearizing the nonlinear system equation, $f(x, u, t)$, using only the first order of its Taylor series expansion, shown in Equation 2.16, then evaluated at the current condition. The discrete terms P and Q_d are then calculated using previously defined discrete-time conversion methods (see Table 2.2). These calculations occur each time step, making A, Φ, and Q_d time-varying.

$$A[t_i, \hat{x}(t_i^-)] \triangleq \left. \frac{\partial f[x, u, t_i]}{\partial x} \right|_{x=\hat{x}(t_i^-)} \tag{2.16}$$

Like the traditional Kalman filter, the covariance and state estimate continue to be propagated iteratively until a measurement update comes available.

Measurement Updates. During the measurement update for the EKF, Equations (2.12a) - (2.13b) are used with one possible constraint: unlike the traditional Kalman filter, the measurement model in terms of the states is not always linear. This issue is resolved by linearizing the nonlinear measurement model, h, about the current operating condition, shown in Equation (2.17).

$$H[t_i, \hat{x}(t_i^-)] \triangleq \left. \frac{\partial h[x, t_i]}{\partial x} \right|_{x=\hat{x}(t_i^-)} \tag{2.17}$$

The linearized result, H, is now available to update the state estimate and covariance [15]. From here, the state estimate and covariance are propagated forward until a new update is available, at which time the new linearized predictions are produced again based on the the new current estimates [15]. This explanation provided a brief summary of the EKF; however, it is not the only choice available in Kalman filtering for nonlinear systems. Another example of such a filtering technique is the Unscented Kalman filter.

2.3.2 Unscented Kalman Filter. Otherwise known as a Sigma Point Kalman Filter (SPKF), the UKF also attempts to estimate states of a nonlinear system. The major differences between the algorithm development of the EKF and the UKF is that instead of propagating the mean and covariance values, sigma points, describing the pdf, are propagated, transformed, and their statistics are used to update the mean and its uncertainty [32]. Several studies have shown that the UKF outperforms the EKF; however, the downfall is the longer computation time required to propagate sigma points [3] [34]. The UKF method is broken into three steps:

- Build Sigma Points
- Propagate Sigma Points
- Perform Measurement Update

Build Sigma Points. The UKF deals with nonlinearities in a system a better than the EKF. Where the EKF uses a first-order approximation for the system model, the UKF uses a higher-order approximation, which is a function of the unscented transform. The "unscented transformation" is a method of "calculating the statistics of a random variable which undergoes a nonlinear transformation" [12]. Both the EKF and UKF assume the pdf to be Gaussian; however, to define variations of the state pdf, the UKF uses sigma points instead of the mean and covariance to more precisely propagate and update. Each sigma point is propagated through the nonlinear function, then weighted, then used to calculate the state estimate and covariance. The number of sigma points used is one more than twice the number of states ($2L$+1). In determining the estimated mean and covariance, the engineer has a few parameters that are design specific [32]:

$$\lambda = \alpha^2(L + \kappa) - L \tag{2.18a}$$

$$p_s = L + \lambda \tag{2.18b}$$

$$W_{0m} = \frac{\lambda}{p_s} \tag{2.18c}$$

$$W_{0c} = W_{0m} + (1 - \alpha^2 + \beta) \tag{2.18d}$$

$$W_{ukf} = \frac{1/2}{p_s} \tag{2.18e}$$

The weighting's tuning parameters, λ, α, β, and κ, are used for scaling, changing the spread, tuning, and depicting the shape, respectively. The last three variables are used by the engineer to tune the filter. Typical parameter values for Gaussian pdfs are: $\kappa = 0$ and $\beta = 2$. The tuning parameter α value is typically selected between

1e-4 $\leq \alpha \leq 1$ [27]. After the tuning parameters are selected, the calculated weights are used to determine the estimated states and uncertainty. The weight is a function of the matrix. The weighting is defined by three values: W_{0m} - weighting for estimated mean for nominal sigma point, W_{0c} - weighting for estimated covariance for nominal sigma point, and W_{ukf} - remaining weight for all other sigma points related to mean ($\bar{x}$) and covariance, indicating the spread. But, before the weights can be applied, the sigma points (χ) need to be built [32],

$$\chi_0 = \bar{x} \tag{2.19a}$$

$$\chi_k = x_{mean} = \bar{x} - \left(\sqrt[c]{(L+\lambda)P_{xx}})\right)_k \tag{2.19b}$$

$$\chi_{k+L} = \bar{x} + \left(\sqrt[c]{(L+\lambda)P_{xx}}\right)_k \tag{2.19c}$$

where "k" represents the k^{th} sigma point and corresponds to the k^{th} column of the concatenated matrix, χ. The vector χ_0 is the mean, while the other vectors are χ_0 plus or minus the Cholesky square root of a weighted covariance direction. Usually each sigma point vector has a magnitude and direction different than the others. These sigma points will undergo a transformation using nonlinear models during the propagation and measurement update phases. This transformation is the unscented transformation previously discussed, which is the "basis of the UKF" [32].

Propagate Sigma Points. During propagation, the sigma points are transformed through the nonlinear propagation function, f. Each column in the matrix χ represent $2L+1$ vectors (χ_0 through χ_{2L}):

$$\chi(t_i) = f[\chi(t_{i-1}), u(t_{i-1})] \tag{2.20}$$

After the sigma point transformation, the mean and covariance of $\chi(t_i)$ are calculated using the following equations, where Q_d is the discrete-time process noise

covariance. The resultant state estimate and covariance are calculated as follows, where k symbolizes the k^{th} sigma point vector.

$$\hat{x}(t_i^-) = W_{0m}\chi_0(t_i) + \sum_{k=1}^{2L} W_{ukf}\chi_k(t_i) \tag{2.21a}$$

$$P_{xx}(t_i^-) = W_{0c}\left(\chi_0(t_i) - \hat{x}(t_i^-)\right)\left(\chi_0(t_i) - \hat{x}(t_i^-)\right)^T + \sum_{k=1}^{2L} W_{ukf}\left(\chi_k(t_i) - \hat{x}(t_i^-)\right)\left(\chi_k(t_i) - \hat{x}(t_i^-)\right)^T + Q_d(t_i) \tag{2.21b}$$

This propagation occurs every Δt, the defined time step, until a measurement is available [32].

Perform Measurement Update. When a measurement becomes available, the filter performs an update. A new set of sigma points needs to be calculated because of the addition of process noise after the recent propagation. The calculations shown in Equations (2.19a) - (2.19c) are used with $\hat{x}^-$ to determine the new sigma points. Next, a prediction of the measurements is made based on these sigma points. This prediction is accomplished by transforming the sigma points through the nonlinear measurement function, h. The resultant sigma points, Z_i, are then used to calculate the measurement prediction, $\hat{z}_k$, and its associated uncertainty, $P_{\hat{z}\hat{z}}$, as shown in Equations (2.22a) - (2.22c).

$$\hat{z}(t_i) = W_{0m}Z_0(t_i) + \sum_{k=1}^{2L} W_{ukf}Z_k(t_i) \tag{2.22a}$$

$$P_{\hat{z}\hat{z}0}(t_i) = W_{0c}(Z_0(t_i) - \hat{z}(t_i))(Z_0(t_i) - \hat{z}(t_i))^T \tag{2.22b}$$

$$P_{\hat{z}\hat{z}}(t_i) = P_{\hat{z}\hat{z}0}(t_i) + \sum_{k=1}^{2L} W_{ukf}(Z_k(t_i) - \hat{z}(t_i))(Z_k(t_i) - \hat{z}(t_i))^T + R \tag{2.22c}$$

The updated mean and covariance become,

$$\hat{x}(t_i^+) = \hat{x}_k^- + K(z_{meas} - \hat{z}_k) \tag{2.23a}$$

$$P_{xx}(t_i^+) = P_{xx}(t_i^-) - K P_{\hat{z}\hat{z}}(t_i) K(t_i)^T \tag{2.23b}$$

using new equations, different from the traditional Kalman filter, where the Kalman filter gain, K, is calculated using the cross correlation matrix (P_{xz}) and the innovation covariance matrix ($P_{\hat{z}\hat{z}}$) [12].

$$K(t_i) = P_{xz}(t_i)(P_{\hat{z}\hat{z}}(t_i))^{-1} \tag{2.24a}$$

$$P_{xz}(t_i) = \sum_{i=1}^{2L} \Big[W^{0c}(\chi_0(t_i) - \hat{x}(t_i^-)) \qquad W^{ukf}(\chi_k(t_i) - \hat{x}(t_i^-)) \Big] * \Big[W^{0c}(Z_0(t_i) - \hat{z}(t_i)) \qquad W^{ukf}(Z_k(t_i) - \hat{z}(t_i)) \Big]^T \tag{2.24b}$$

After each update, the filter will propagate the state estimate and covariance until another measurement is available [32]. During this iterative process, the state estimate is also routed to other functions in the controller. The function that uses these states to determine the input to the plant is the Linear Quadratic Gaussian (LQG) Controller.

2.4 Linear Quadratic Gaussian (LQG) Controllers

The LQG control method represents a "systematic design of multi-variable control system using both deterministic and stochastic dynamic optimal control ideas" [2]. In the name, "linear" refers to its association with linear systems, "quadratic" due to the quadratic cost function, and "Gaussian" due to the Gaussian noise sources [32]. The LQG controller can be designed and applied to continuous, as well as discrete-time applications.

2.4.1 Continuous LQG Control Design. In the continuous-time case, the system model is described by the following differential equation [24],

$$\dot{x} = Ax + Bu + Gw \tag{2.25}$$

The states and the inputs of this system model are used in a quadratic cost function/performance measurement. The goal of LQG design is to create a controller by minimizing this cost function. The cost function used for continuous-time systems is given by,

$$J = \int_0^T (x^T X x + u^T U u)dt + x^T(T) X_f x(T) \tag{2.26}$$

where X (positive semi-definite matrix) and U (positive definite matrix) are weighting matrices which influence the cost on the state and on the inputs, respectively. The cost for each state and input will be determined by their expected values relative to each other and the relative cost of departure from their desired value. Furthermore, X_f is used to weight the accuracy of the state at final time, T. The selection of these matrices will vary based on the engineer's desired response of the system. One text suggest two simple guidelines [2]:

1. Make all weighting matrices diagonal.
2. Select large values for any variable required to be small in the time domain.

For example, if the input varies over a small value range, a higher weight on the U matrix would be appropriate; however if the input values are high, a higher weight could cause "transient behavior in the states to be more pronounced" [2]. Once the weighting matrices are selected, the matrix Riccati equation is used to find the covariance matrix, P:

$$-\dot{P} = A^T P + PA + X - PBR^{-1}B^T P \tag{2.27a}$$

$$P(T) = X_f \tag{2.27b}$$

The matrix, P, is then used to determine the LQR (linear quadratic regulator) gain, G_c:

$$G_c(t) = R^{-1}B^T P(t) \tag{2.28}$$

This gain is then used in a negative feedback configuration to control the system by modifying the following input, assuming that the desired state vector contains all zeros.

$$u^* = -G_c(t)x \tag{2.29}$$

This method is not only available in the continuous-time domain, but equations are also available in discrete-time.

2.4.2 Discrete LQG Control Design. For a discrete-time implementation, the idea is the same, but the equations do change. The system model is described by the following difference equation [32]:

$$x_{k+1} = \Phi_k x_k + B_{d\,k} u_k + w_k \tag{2.30}$$

The overarching goal of the LQG design is to minimize the following defined cost function J [16]. For discrete-time applications, this cost function is defined as,

$$J = \sum_{i=0}^{N} \frac{1}{2}[x^T(t_i)\, X(t_i)x(t_i) + u^T(t_i)U(t_i)u(t_i)] + \frac{1}{2}x^T(t_{N+1})X_f x(t_{N+1}) \tag{2.31}$$

where X, U, and X_f are weighting matrices that are identically defined and held to the same criteria as previously characterized for continuous-time. The selection of these matrices will once again vary based on the engineer's desired response of the system within the sampling period [16]. The optimal control, u^*, is a function of the output state estimate of the Kalman Filter and G_c^*,

$$u^*[\hat{x}(t_i^+), t_i] = -G_c^*(t_i)\hat{x}(t_i^+) \tag{2.32}$$

where G_c is the optimal feedback LQR controller gain given by the following equation, and the backward Riccati difference equation, K_c, is solved backwards with the terminal condition, $K_c(t_{N+1}) = X_f$ [16].

$$G_c^*(t_i) = \left[U(t_i) + B_d^T(t_i)K_c(t_{i+1})B_d(t_i)\right]^{-1}\left[B_d^T(t_i)K_c(t_{i+1})\Phi(t_{i+1},t_i)\right] \tag{2.33a}$$

$$\begin{aligned} K_c = {} & X(t_i) + \Phi^T(t_{i+1},t_i)K_c(t_{i+1})\Phi(t_{i+1},t_i) \\ & - \left[\Phi(t_{i+1},t_i)K_c(t_{i+1})B_d(t_i)\right]\left[U(t_i) + B_d^T(t_i)K_c(t_{i+1})B_d(t_i)\right]^{-1} \\ & \times \left[B_d^T(t_i)K_c(t_{i+1})\Phi(t_{i+1},t_i)\right] \end{aligned} \tag{2.33b}$$

The control input, u^*, is not only applied to the plant, it is also routed back as an input to the Kalman Filter, along with the measurement, y, as shown in Figure 2.7 [16]. One piece of the design missing from this figure is how the measurements are made going to the Kalman filter. These measurements can be taken from a variety of sensors, such as magnetometers, inertial sensors or cameras.

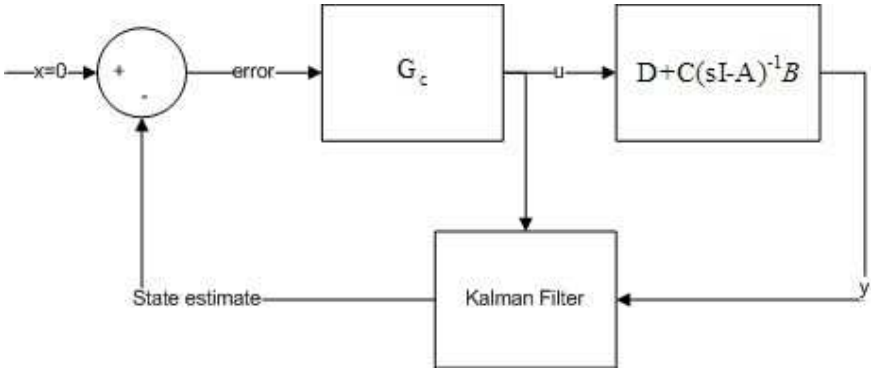


Figure 2.7: LQG State-Feedback Diagram [2]. This diagram corresponds with Equation 2.32, in deriving the optimal feedback control, when the desired states are zero.

2.5 Vision-Aided Navigation

In environments where GPS is degraded or denied, engineers are developing algorithms that take images from a camera and provide measurements for position and

attitude in navigation. One common example is Visual Simultaneous Localization and Mapping (SLAM). SLAM not only performs the image processing, but also performs the activity of mapping and map maintenance in exchange for maximum information usage. SLAM could perform the image processing by one of a number of techniques. Two mentionable techniques are: optic-flow and feature-tracking. Optic flow takes into account the entire image by creating a uniform sampling lattice. The algorithm analyzes the whole picture over time to determine location. The other example of localizing is the feature-tracking method. This method focuses on the features in an image, as opposed to the whole picture. One algorithm commonly used for feature-tracking is the Scale-Invariant Feature Transform (SIFT) [14]. To understand how images can be used to provide a navigation solution, the SIFT process is briefly discussed.

The SIFT method was developed by David G. Lowe of the University of British Columbia. This approach analyzes objects in an image and generates descriptors for these interest points in an effort to "extract distinctive invariant features" from the image portraying the area of navigation [14]. The classification of invariant relates to image scale and rotation of the features describing objects within the image. As the camera moves about the room, the SIFT algorithm generates descriptors in effort to determine the camera's relative location within a predefined inertial frame of reference. Specifically, additional processing is required to calculate position and attitude after feature matching. An example of this additional processing is the use of the Kalman filter [32].

Only a top-level understanding of vision navigation is required to accomplish the research in this thesis. This section provides a brief overview of how vision can be used onboard a MAV to calculate position and attitude; however, cameras can also be used outside the MAV to provide a very accurate solution for the purpose of risk mitigation testing. This type of system is available at AFRL and is used during hardware testing; the manufacturer of this particular system is Vicon.

2.6 Vicon System

The Vicon System is a real-time motion capture flight control system used in applications ranging from animation to tracking aerial vehicles. There are several labs established in academic/research environments using Vicon for unmanned aerial vehicle (UAV) control. Some of these include, but are not limited to, MIT, Georgia Institute of Technology, the Boeing Corporation, and the Air Force Research Laboratory (AFRL) [1] [2] [3]. For this particular application, this system allows the tracking and control of a "captured" object, such as a moving vehicle, with millimeter accuracy. "Captured" refers to the Vicon system identifying an object by the orientation of well-placed reflectors on a vehicle. When an object is captured, the system identifies the reflectors as being on a particular vehicle and uses them as reference points when finding center of mass and calculating position and attitude. A typical system setup requires a collection of motion capture cameras mounted and focused on a confined area and processing units that connect and synchronize the cameras to a desktop computer.

At AFRL/RB's Laboratory, Micro-Air Vehicle (MAV) Indoor Flight Facility (IFF), 36 four-megapixel cameras are housed in a 30'×30'×20' room. Nine cameras are mounted on each wall to sufficiently track captured objects as shown in an active screen shot in Figure 2.8. The overall purpose of the laboratory is to provide engineers an environment in which to test MAV controllers without integrating a payload on the vehicle. The setup of the Vicon System at AFRL is detailed in Figure 2.9. The data from the cameras is sent to a National Instruments real-time processor using an ethernet connection at 120 Hz. This data is then processed in real-time in Labview to provide position, velocity, attitude, and attitude rates to the controller. The controller uses this information to generate control signals at a pre-defined 50 Hz, which is converted to a PPM (pulse-position modulation) signal by a PPM generator. This

[1] http://acl.mit.edu/
[2] http://www.vicon.lt/
[3] http://www.vicon.com/company/releases/220108.htm

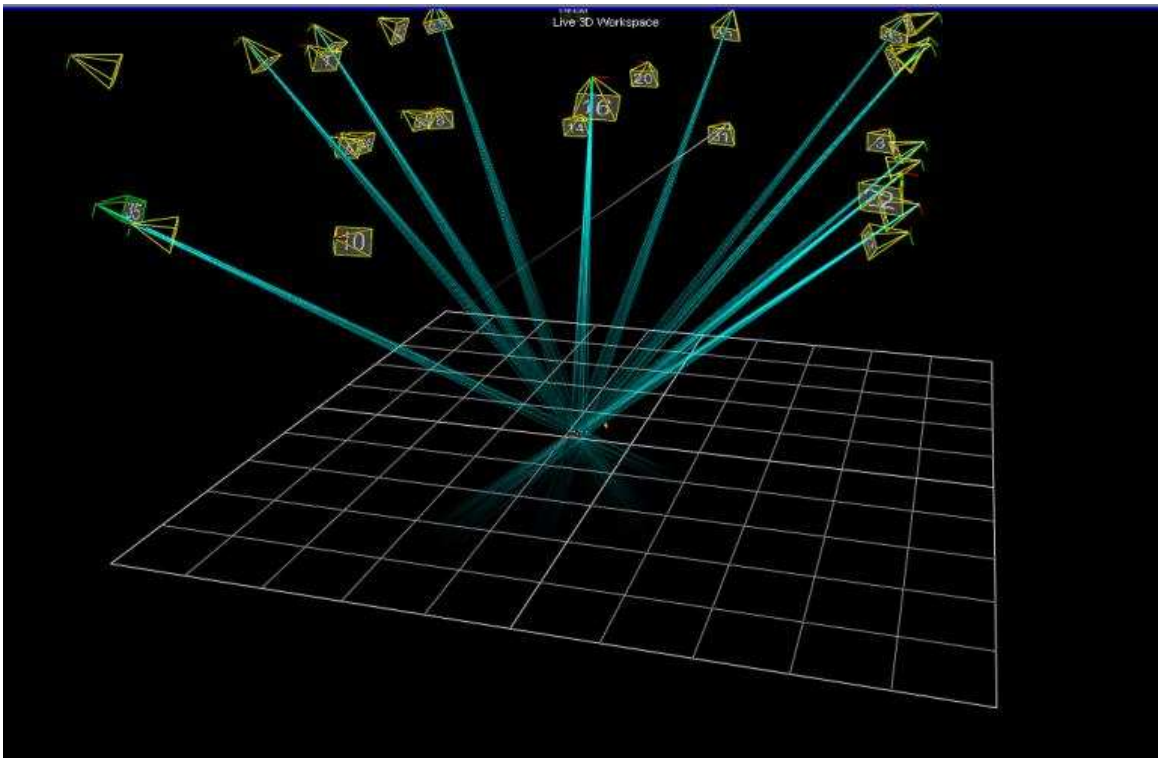


Figure 2.8: AFRL/RB MAV Lab Screen Shot. Several cameras are tracking a captured object located in the center of the room. The grid shown represents the floor, and the vehicle is defined by well placed reflectors on the body of the MAV.

PPM signal is then sent to the MAV's remote control through a cable connected to the trainer port. If the bypass switch on the controller is activated, the control signal from the PPM generator will be transmitted to the MAV in place of the control signals provided in manual operation. For a customer of AFRL's MAV IFF, this process is mostly transparent. Labview acts as the primary user interface, allowing engineers to integrate and monitor the execution of their controllers real-time [18].

Understanding the Vicon system is crucial to grasping the design of the upcoming hardware tests. In retrospect, the information presented in this chapter will be used in the research and design of the helicopter controller. In this chapter, coordinate and transformation systems, inertial navigation systems, Kalman filtering techniques, linear quadratic Gaussian controllers, vision-aided navigation, and the Vicon system were discussed to provide a background of the concepts behind the research support-

ing this thesis. Armed with these concepts, the methodology behind the design will be discussed and quantified in the next chapter.

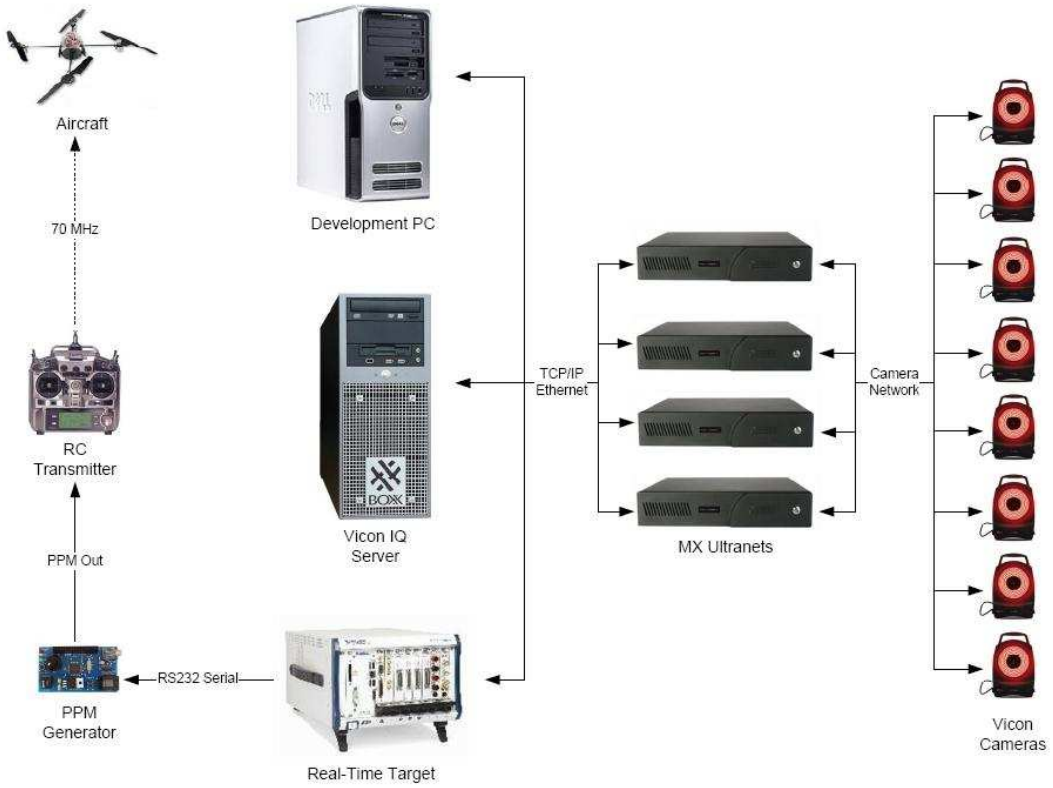


Figure 2.9: AFRL/RB MAV Lab Vicon Control System [18]. Data is collected by the cameras in real-time, sent to the development PC for processing, converted to control signals in Labview, then transmitted to the MAV through the remote control.

III. Methodology

BEFORE building a MAV controller, two decisions have to be made: select a MAV and select a controller functional schema. First and foremost, a MAV will need to be selected for the project. If the end goal is to have a fully autonomous air vehicle, several vehicles will need to be procured as back-ups to reduce the risk of delays in schedule in case a vehicle malfunctions or is damaged. Next, the vehicle will need to be tested to determine lift capacity since a payload with sensors and a microprocessor is in its future (assumed to be approximately 120 grams). This information is not typically available for commercial, off-the-shelf air vehicles so some risk of the choice not meeting requirements is expected. Finally, the vehicle should be small enough to move inside a building in all modes of flight: forward, backwards, laterally, and hover. One particular vehicle was readily available in AFIT's ANT (Advanced Navigation Technology) Center that met all requirements: the Walkera 53-1 four-channel radio remote-controlled, micro-sized coaxial helicopter, hereafter dubbed as "El Toro".

Next, an overall design schema for the controller is developed. Figure 3.1 shows the basic idea. The mechanization block represents INS mechanization, taking raw INS data from accelerometers and gyros to produce position, velocity and attitude, or system model mechanization, taking inputs from the controller to produce position, velocity, attitude and attitude rates, or a combination of both. In other words, the mechanization produces a nominal state vector. This information is subtracted from the corresponding measurements calculated from the camera images to produce a measurement error. This error is provided as the measurement input, along with the control signal, to the Kalman filter. The Kalman filter produces an error state estimate to the mechanization. The mechanization makes the necessary adjustments to the nominal states. A reference state vector representing the desired states is subtracted from the nominal states; the resultant error vector is multiplied by the LQR gain, G_c, then sent to the plant as the control signal. To adequately explain each step in the design process, this chapter is broken into the following sections:

- El Toro System Model

- Controller Design
- Stochastic Estimation
- Inertial Navigation
- System Model and INS Combination
- Final Design

The first step of this model-based control design is to create a mathematical model of El Toro.

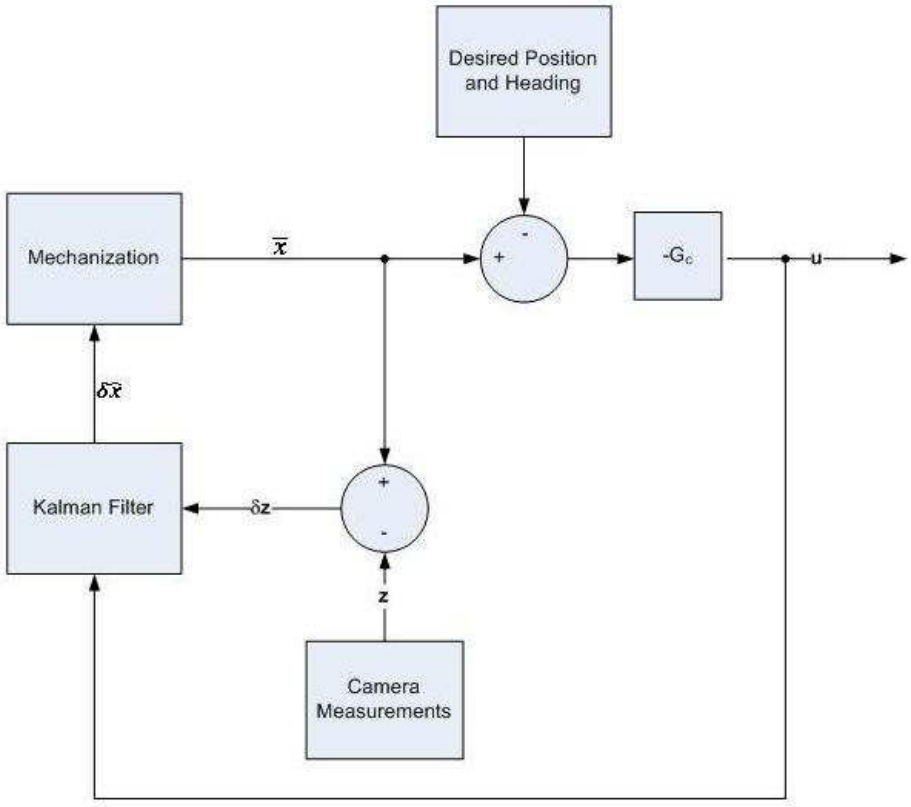


Figure 3.1: Controller Functional Diagram. This diagram represents the general design schema going into the helicopter control design process.

3.1 El Toro System Model

The *El Toro* helicopter (Figure 3.2) is actually a stripped-down and modified Walkera model 53-1. This particular micro-air vehicle is a counter-rotating design with the upper blade utilizing an inertial flybar for stabilization during changes in attitude in pitch and roll. The remote control provides a rudder and throttle with limits incorporated for pitch and roll.

Figure 3.2: Walkera 53-1 (El Toro) . This commercially-available, remote-controlled, micro-sized helicopter is selected for modeling and autonomous control.

Before developing a system model, a good understanding of the system's dynamics is essential. To aid in describing the dynamics of El Toro, a functional diagram was procured from a classroom discussion and modified as a reference for discussion (Figure 3.3). To begin, El Toro has two sets of blades: upper and lower. Each set is controlled by a different motor. The throttle and rudder controls determine the output of these motors. The throttle on the remote control is a notched lever that when moved up, the two motors gain angular velocity in unison. The increase in angular velocity causes the blades to spin at a faster rate, which, in turn, produces more lift. When the throttle lever is moved down, the motors spin down, producing the opposite effect.

The next three controls, rudder, pitch, and roll, are spring-loaded levers. With these three levers remaining in the neutral position, the helicopter stays level at a constant heading.

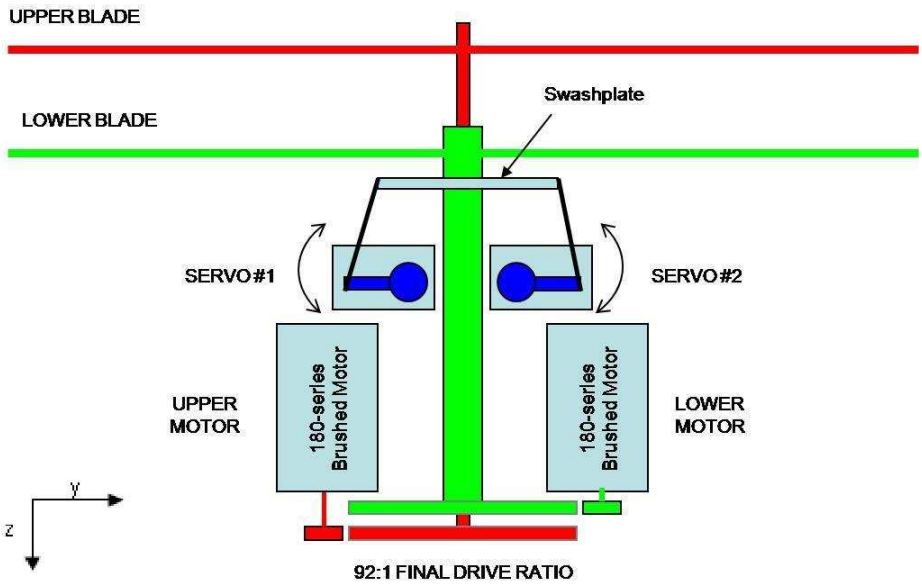


Figure 3.3: El Toro Functional Diagram [31]. Two brushed motors control the speed of the upper and lower blade sets, while the two servos control the swashplate, which in turn controls the attitude of the helicopter.

In contrast, the rudder produces a change in heading when its lever is moved from its default position. With no rudder input, both upper and lower blade sets move at the same angular speed, but in opposite directions. However, if there is a difference between these angular velocities, a rotation about the z-axis (body-frame) is produced. The magnitude and direction of this yaw motion depends on the length of time and the direction and amount of force on the lever is applied. In addition, the Walkera 53-1 includes a yaw stabilization circuit installed to counter any uncommanded yaw disturbances.

The final two controls on the remote control are: pitch and roll. These controls provide input to the two servos shown in Figure 3.3. These servos control the swashplate, which changes the attitude of the lower blades. Upon this change the upper blades follow suit, but only after a short delay due to the gyroscopic procession of the inertial flybar. The purpose of the flybar is to provide stability during these changes in cyclic. This overall action changes the attitude of the aircraft, thus causing a translation in the body's x and y-direction. Specifically, servos #1 and #2 rotate clockwise

for a positive roll, and counter-clockwise for a negative roll, tipping the swashplate from right to left. For pitch, the servos rotate in opposite directions. Servo #1 rotates counter-clockwise and servo #2 rotates clockwise for a positive pitch, tipping the swashplate up towards the front of the helicopter. For a negative pitch, servo #1 rotates clockwise and servo #2 rotates counter-clockwise, causing the swashplate to tip down towards the front of the aircraft. When the pitch and roll controls are released by the operator, the vehicle levels. As a final note, the vehicle requires continuous input from the operator to remain in hover condition.

In addition, the ANT Center provided the specifications for this vehicle in Table 3.1. These parameters were used to develop the nonlinear system model.

Table 3.1: El Toro Specifications [31]. These parameters are provided by the ANT Center and are paramount in developing the system model.

Vehicle mass (w/ battery):	320 grams
Blades (4 total):	22 cm length, 6.2 g each
Gearing:	92:1 final drive ratio
Maximum voltage:	7.4 VDC
Maximum drive current:	8 Amps
Motor constants:	$K_e = 0.026$ V-s
	$R_m = 0.42$ Ohms
	$L = 200$ uH
Motor/Drive Constants:	$K_t = 0.042\ N - \frac{m}{A}$
	$b = 140\ mg\frac{m}{s}$

3.1.1 Nonlinear System Model Derivation. As a preface to this section, much of the design of the *El Toro* model was produced as a result of previously accomplished graduate class projects with Capt Jason Bingham. Furthermore, as a preface to the original project, much of the background information on El Toro was developed at the ANT Center. As previously mentioned, the Walkera 53-1 has

two brushed motors. These motors are modeled using commonly-known dynamics equations [6]:

$$M_{inertia}\dot{\omega} = -b\omega + K_t I - \tau_{load} \tag{3.1a}$$

$$L\dot{I} = -RI - K_e\omega + V \tag{3.1b}$$

where $M_{inertia}$ is the moment of inertia, I is the motor current, V is the motor voltage, R is the motor resistance, ω is the motor angular velocity, L is the motor inductance, and τ_{load} is the motor torque that is created with acceleration. The remaining variables stem from the motor parameters defined in Table 3.1. Furthermore, the moment of inertia, $M_{inertia}$ (kg-m^2), was calculated using the moment of a rod with length of 44 cm (twice the blade length) and mass of 0.0124 kg:

$$M_{inertia} = \frac{0.0124(0.44)^2}{12} \tag{3.2}$$

The functions for lift (N) and torque load due to acceleration (N-m) in terms of the motor speed were previously derived by through experimentation:

$$F_{lift} = 0.0098\left(\frac{\omega^2}{125} - \frac{\omega}{4} - 1.17\right) \tag{3.3a}$$

$$\tau_{load} = -\frac{e^{\frac{\omega}{65}}}{122} \tag{3.3b}$$

The fusion of the motor, lift, and torque equations represent the dynamics of the brushed motor. These dynamics were translated to Simulink, shown in Figure 3.4, for future simulation.

Referring in Figure 3.4, the area outside the red box shows a series of computations that was devised to account for the effect of pitch and roll on lift. These are

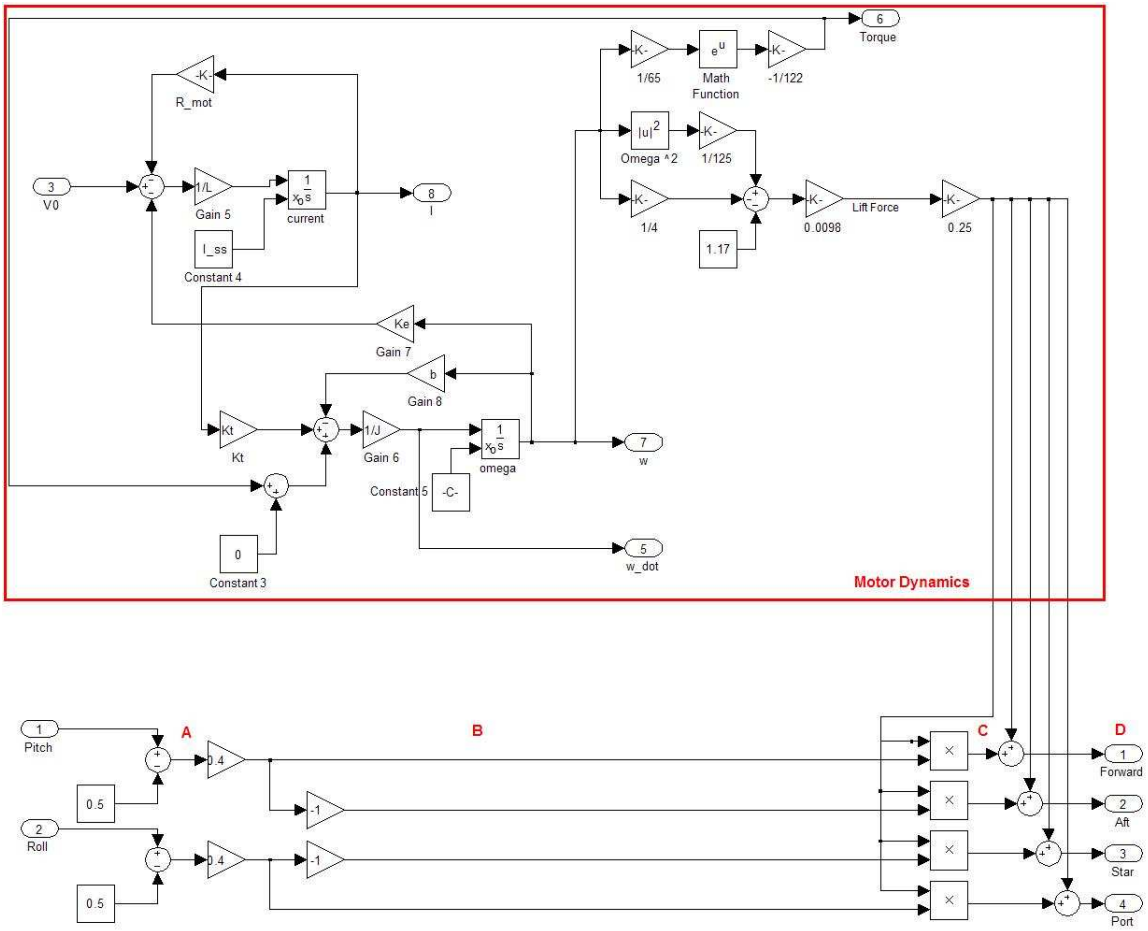


Figure 3.4: Motor Blade Simulink Diagram. The components within the red box model the brushed motor dynamics. The components outside the red box depict the effect pitch and roll have on lift.

Table 3.2: Blade Equations. This equations account for the effect of pitch and roll on total lift. Each column refers to a specific label shown in Figure 3.4.

	A	**B**	**C**	**D**
Forward	Pitch - 0.5	0.4A	$\frac{1}{4}F_{lift}B$	C + $\frac{1}{4}F_{lift}$
Aft	Pitch - 0.5	-0.4A	$\frac{1}{4}F_{lift}B$	C + $\frac{1}{4}F_{lift}$
Star	Roll - 0.5	0.4A	$\frac{1}{4}F_{lift}B$	C + $\frac{1}{4}F_{lift}$
Port	Roll - 0.5	-0.4A	$\frac{1}{4}F_{lift}B$	C + $\frac{1}{4}F_{lift}$

referred to as the blade equations. Table 3.2 provides a brief outline of the computations. The columns in this table represent the particular areas of interest outside the red box shown in Figure 3.4. The outputs of this figure account for the lift produced

on a disk representing rotating blades. Forward, aft, star, and port account for the areas on the disk in which the lift is produced. Understanding the blade equations starts with the pitch and roll inputs. The inputs of pitch and roll range from zero to one, with 0.5 generating no pitch or roll, and the extremes changing lift by 20%. The outputs of the upper and lower motor-blade models (each represented by Figure 3.4) are used to generate the forces and torques to be applied to the 6DOF Simulink model. The force and torque equations are as follows, where r_{blade} represents blade length, and subscripts 0 and 1 indicate the lower and upper blades, respectively:

$$F_x = -mg\ sin\theta \tag{3.4a}$$

$$F_y = mg\ sin\phi cos\theta \tag{3.4b}$$

$$F_z = mg\ cos\theta cos\phi - (Fwd_{total} + Aft_{total} + Star_{total} + Port_{total}) \tag{3.4c}$$

$$M_x = r_{blade}(Fwd_{total} - Aft_{total}) \tag{3.4d}$$

$$M_y = r_{blade}(Port_{total} - Star_{total}) \tag{3.4e}$$

$$M_z = (M_{inertia}\dot{\omega_0} - \tau_{load}) - (M_{inertia}\dot{\omega_1} - \tau_{load}) \tag{3.4f}$$

The inertia matrix used in the 6DOF model was derived from the measured mass (m) of the helicopter body, along with the measured physical dimensions, assuming a homogeneous solid: height (h) - 4 cm, width (w) - 7 cm, and depth (d) - 8 cm. The equations used to calculate the moment of each axis came from the standard inertia equation of a solid cuboid:

$$I_x = \frac{m}{12}(h^2 + w^2) \tag{3.5a}$$

$$I_y = \frac{m}{12}(h^2 + d^2) \tag{3.5b}$$

$$I_z = \frac{m}{12}(d^2 + w^2) \tag{3.5c}$$

The results make up the following inertia matrix:

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{3.6}$$

Finally, the voltage input to the motor-blade circuits were determined to be a function of the throttle (u_1) and rudder (u_2) commands:

$$V_0 = \frac{1}{2}(u_1 + u_2) \tag{3.7a}$$

$$V_1 = \frac{1}{2}(u_1 - u_2) \tag{3.7b}$$

The final nonlinear model in terms of the input (u) and states (x) was produced from this design:

$$\dot{x} = f(x, u) \tag{3.8}$$

The resulting input is a unitless vector containing throttle, rudder, pitch (u_3), and roll (u_4); the resulting states are representing in a vector (x) that contains position (meters), velocity($\frac{m}{s}$), attitude (radians), attitude rates ($\frac{rad}{s}$), the two motor angular velocities ($\frac{rad}{s}$) and current (amps), and the flybar attitude (radians); and the resulting output (y) is a vector representing the first twelve states. The subscripts 0 and 1 indicate the lower and upper blades, respectively, and FB signifies flybar states.

$$u = [u_1,\ \ u_2,\ \ u_3,\ \ u_4]^T \tag{3.9a}$$

$$x = [x,\ y,\ \ z,\ \dot{x},\ \dot{y},\ \dot{z},\ \phi,\ \theta,\ \psi,\ \dot{\phi},\ \dot{\theta},\ \dot{\psi},\ \omega_0,\ I_0,\ \omega_1,\ I_1,\ FB_\phi,\ FB_\theta]^T \tag{3.9b}$$

$$y = [x,\ \ y,\ \ z,\ \ \dot{x},\ \ \dot{y},\ \ \dot{z},\ \ \phi,\ \ \theta,\ \ \psi,\ \ \dot{\phi},\ \ \dot{\theta},\ \ \dot{\psi}]^T \tag{3.9c}$$

This nonlinear system model was also constructed in Simulink as shown in Figure 3.5, taking into account wind resistance, flybar effects, and remote control dynamics, as described in Table 3.3.

Table 3.3: El Toro Simulink Model Circuit Description. The sections in this table corresponds to blocks identified in Figure 3.5.

Block	Description
A	More accurately models the controls of El Toro
B	Models the effects of the flybar on the upper blade
C	Converts the effects of gravity from the navigation to the body frame of reference
D	Accounts for the decay in velocity ($\dot{x}$ and $\dot{y}$) and yaw rate ($\dot{\psi}$) once the pitch, roll, and rudder controls are released

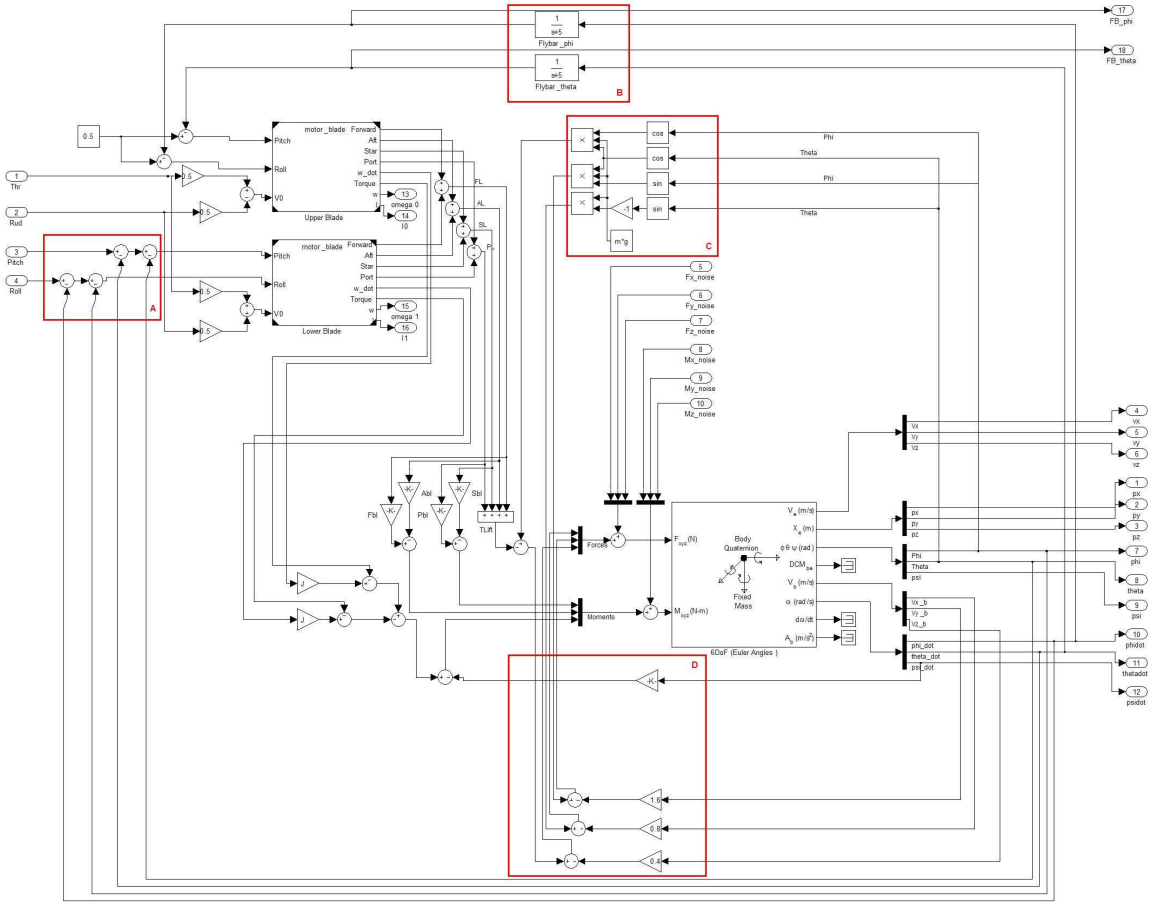


Figure 3.5: El Toro Simulink Model Diagram. This Simulink diagrams characterizes the nonlinear system mathematical model for the El Toro helicopter.

3.1.2 Model Reduction. The model in the previous subsection consists of 18 states. Model reduction is traditionally accomplished to simplify the controller design. The first 12 states are considered the standard kinematic states of an aircraft. The last six could be considered nontraditional and are considered for approximation in a lower number state vector. After investigation, it is determined that the motor angular velocity and current can easily be approximated, while the flybar states cannot. Therefore, the 18-state model is reduced to 14 states using the method described in this section.

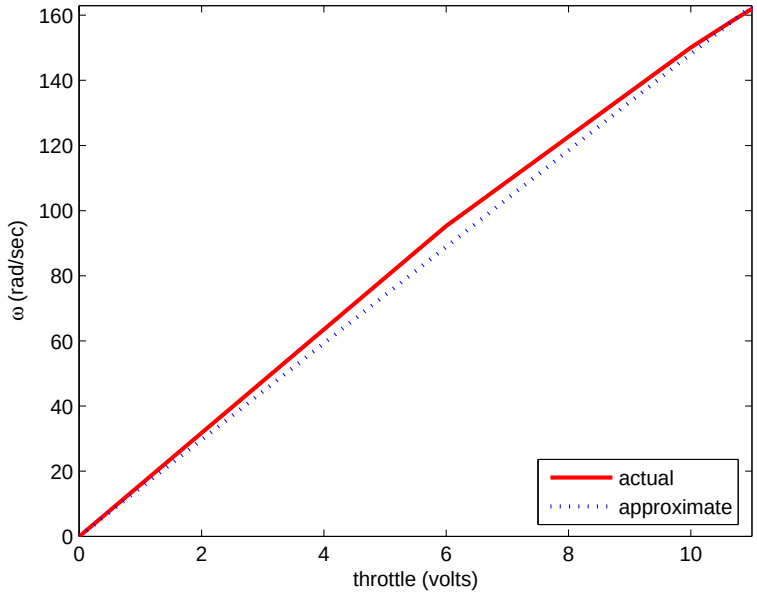
To approximate the response of these four states, each input was varied while watching the response from the upper and lower blade motors. Only the rudder and throttle affected the motors, as expected. Each state's output was plotted against the varying throttle and rudder. Equations for each of the four states to be eliminated were devised in terms of these two inputs and compared to the actual responses in Figures 3.6 and 3.7. Note, the responses are not identical, but are close. Since the controller will be built for hover, the equations match the closest at that point (quantified in the next section). The final equations for ω_0, ω_1, I_0, and I_1 are:

$$\omega_0 = 14.8139(u_1) - 11.39(u_2) \tag{3.10a}$$

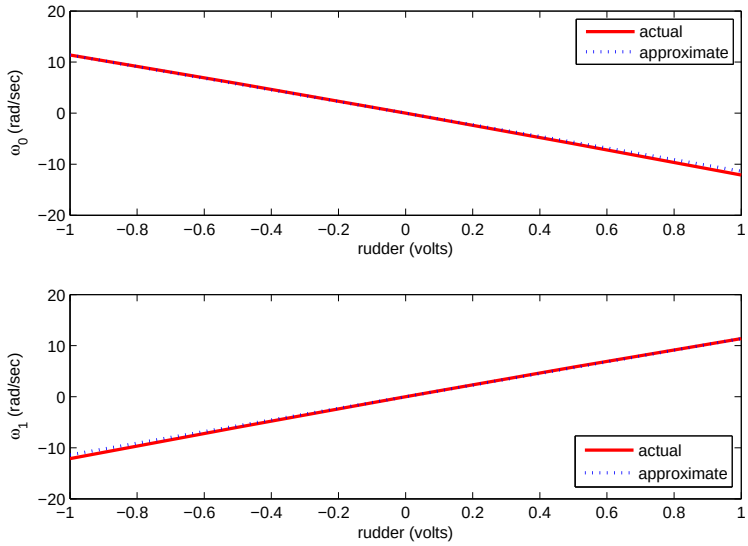
$$\omega_1 = 14.8139(u_1) + 11.39(u_2) \tag{3.10b}$$

$$I_0 = 1.0786(0.14 \times u_1)^2 + 0.4 - 0.4495(u_2) \tag{3.10c}$$

$$I_1 = 1.0786(0.14 \times u_1)^2 + 0.4 + 0.4495(u_2) \tag{3.10d}$$

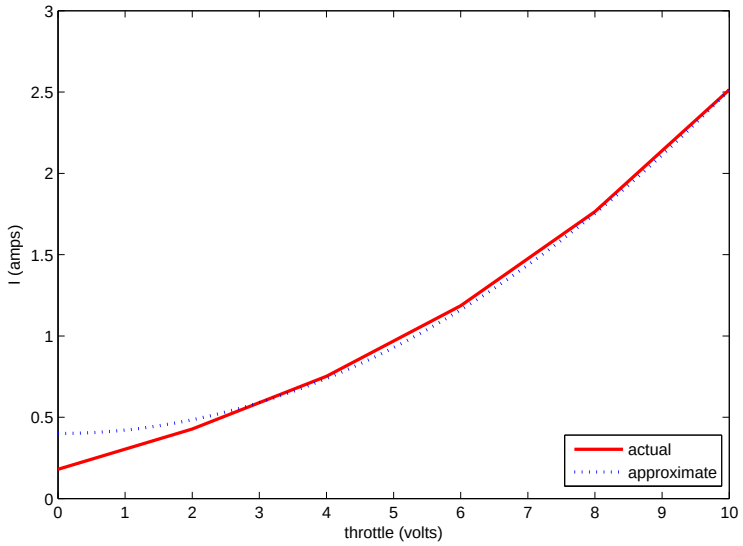


(a) Throttle Response

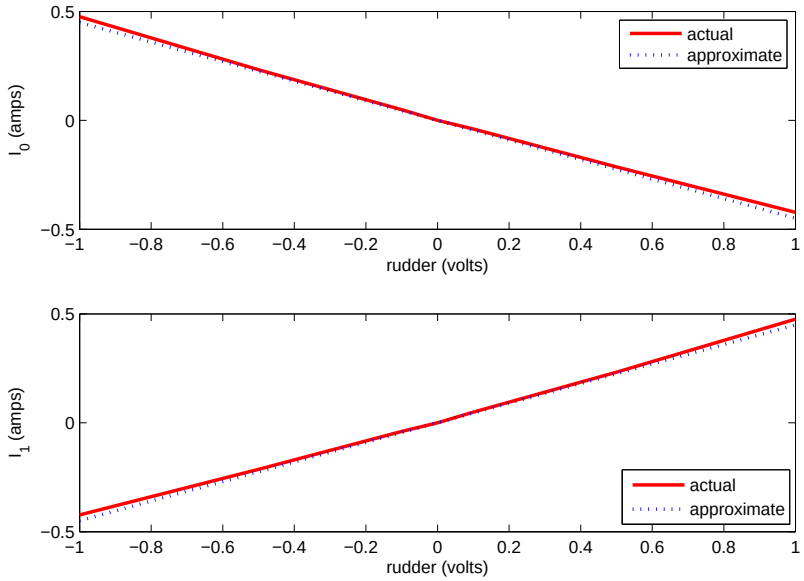


(b) Rudder Response

Figure 3.6: Motor Angular Velocity Approximation vs. Original Response. The actual response is the motor angular velocity response of the 18-state nonlinear model varying throttle and rudder independently. The approximate response is the motor angular velocity response using equations approximating the full-state response.



(a) Throttle Response



(b) Rudder Response

Figure 3.7: Motor Current Approximation vs. Original Response. The actual response is the motor current response of the 18-state nonlinear model varying throttle and rudder independently. The approximate response is the motor current response using equations approximating the full-state response.

3.1.3 System Model Linearization. In order to implement a linear quadratic Gaussian controller, first the nonlinear system model must be linearized. Using the full-state matrix for hover, the angular velocity, ω, was calculated by equating Equation (3.3a) to $\frac{1}{2}mg$. Furthermore, the motor voltage and current were calculated by substituting ω in Equations (3.1a) - (3.1b) and solving for the two respective unknowns. The resulting angular velocity, motor voltage, and motor current for hover was determined to be 158.41 $\frac{rad}{s}$, 5.2785 volts, and 2.76 amps, respectively. In the reduced-state model, this is not a consideration. For both models, the position, velocities, attitude, attitude rates, and flybar angles should all be zero at hover. Substituting in these values along with inputs for hover (mentioned below), the 14-state system model is linearized about the hover condition using the Jacobian method and now expressed by the following deterministic state-space model:

$$\dot{x} = \mathbf{A}x + \mathbf{B}u \tag{3.11a}$$

$$y = \mathbf{C}x + \mathbf{D}u \tag{3.11b}$$

The output, y, consists of the first 12 states; therefore, the $\mathbf{C}$ matrix is a 12×12 identity matrix with two additional columns of zeros to account for the last two unobservable states, and the $\mathbf{D}$ matrix is a 12×4 matrix of zeros because the output is not in terms of the input. The resultant matrices (shown in Appendix A) were verified through Simulink by using the Linear Analysis function located under Tools, Control Design (Control and Estimation Tools Manager). The default operating point was defined by the input levels for a hover condition (shown below), then synchronizing this default operating point to the model.

- Throttle - 10.694
- Rudder - 0
- Pitch Command - 0.5
- Roll Command - 0.5

After the linearization of the Simulink model was performed by the software, the results were compared with the mathematically derived result. The results were identical, which verified the linearization results. Furthermore, the transfer functions relating the four inputs to all twelve outputs were provided by the Simulink Linearization process and are detailed in the Appendix A; the input/output combos not included are equal to zero. The resultant pole/zero map, shown in Figure 3.8, shows system poles (rad/sec) from the derived characteristic equation:

$$s(s+396)(s+321)(s+166)(s+6.21)(s+5)(s+2.5)(s+1.25)(s+0.809) \quad (3.12)$$

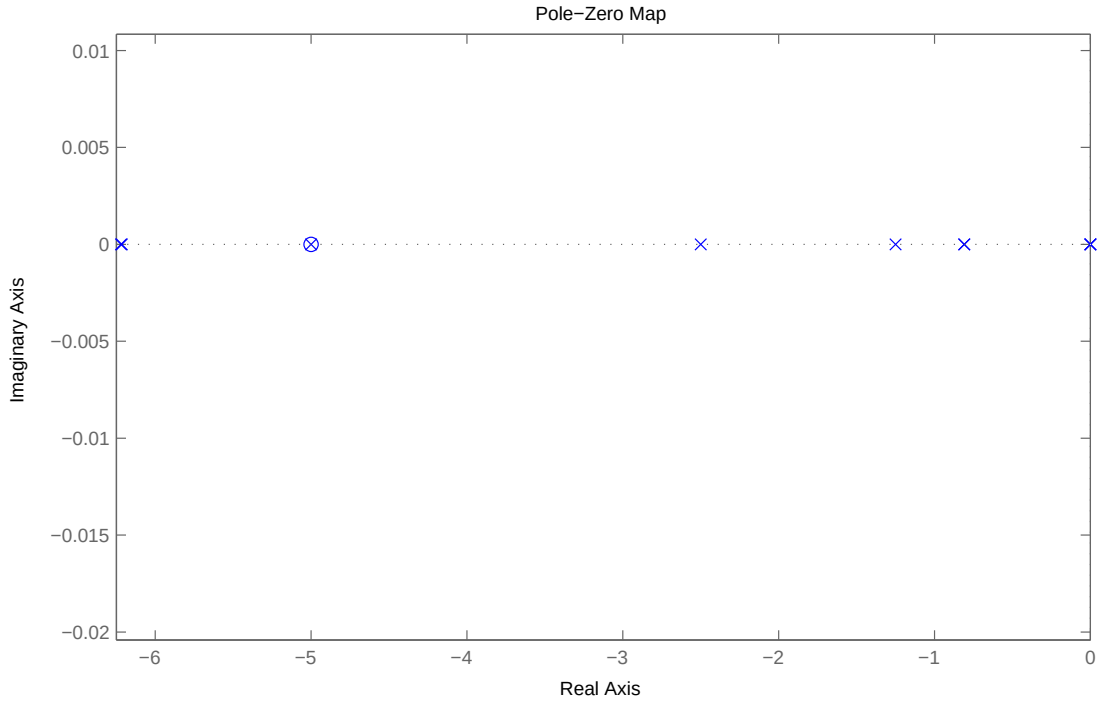


Figure 3.8: El Toro Pole Zero Map. These poles are the dominant poles associated with the El Toro linearized reduced-order system model about the hover condition.

3.1.4 Stochastic Noise Insertion. Moving away from the deterministic side, stochastic noise needs to be accounted for in the system model. Process noise and measurement noise amend the system difference equation as follows, with w representing the process noise, G representing the process noise transition matrix, and v representing the measurement noise.

$$\dot{x} = Ax + Bu + Gw \tag{3.13a}$$

$$y = Cx + Du + v \tag{3.13b}$$

Process noise, otherwise known as dynamics noise, accounts for the randomness that correspond to the dynamics of the system [17]. Measurement noise accounts for random signals that are inserted during the measurement process. It is important to account for these noise sources in order to determine the most probable output and state values. The process and measurement noise vectors for the El Toro System model are approximated using a Gaussian noise sequence with:

$$E[w_k] = 0 \tag{3.14a}$$

$$E[w_k w_i^T] = Q\delta t \tag{3.14b}$$

$$E[v_k] = 0 \tag{3.14c}$$

$$E[v_k v_i^T] = R\delta_{ik} \tag{3.14d}$$

$$E[w_k v_i^T] = 0 \tag{3.14e}$$

In other words, each of the noise sources generated are considered independent, white random variables. The 12 measurement noise sources, v, are added to the output vector, y, shown in Equation (3.13b). The six process noise sources are injected into the system model, as shown in Figure 3.5 (inputs 5-10), by adding Gaussian noise to the forces and moments prior at the input of the 6DOF model. The variables assigned

to this noise vector are:

$$w = [w_{Fx}, w_{Fy}, w_{Fz}, w_{Mx}, w_{My}, w_{Mz}]^T \tag{3.15}$$

In efforts to devise the noise transition matrix, G, these variables are added to the force and moment equations, Equations (3.4a) - (3.4f). The resulting $\dot{x}$ nonlinear system model was in terms of x, u, and w. The Jacobian (first derivative) was then taken with respect to w to produce G. The process noise matrix, G, becomes a function of θ, ϕ, and ψ. The numerical G matrix for hover (all angles equal to zero) is listed in Appendix A. This matrix was verified by linearizing the Simulink model (about hover) using the Control and Estimation Tools Manager, with w identified as inputs.

3.2 Controller Design

A LQR controller was created and used as the heart of the LQG controller design. The resulting gain matrix was calculated using a discrete-time solution due to the eventual integration with the Vicon System, which determines position and attitude at a 50 Hz rate and implementation unto the Blackfin microprocessor, manufactured by Analog Devices. The key to this controller design was determining the weighting matrices. The overarching goal of the controller is to maintain hover; therefore, the position and the heading were weighted higher than the remaining states. The state weighting matrix, X, is a diagonal matrix with the values correlated to the position and heading equal to 10, while all other states (except the motor states) equal to 1. The input weighting matrix, U, was assigned the identity matrix, as displayed below. Both equations use the symbol Λ to signify a diagonal matrix with diagonal elements listed in order of row and column. The units for the states are described in

the Section 3.1.1.

$$\mathbf{X} = \Lambda(10,\ 10,\ 10,\ 1,\ 1,\ 1,\ 1,\ 1,\ 10,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 1,\ 1) \tag{3.16a}$$

$$\mathbf{U} = \Lambda(1,\ 1,\ 1,\ 1) \tag{3.16b}$$

The LQR gain matrix was computed using the *dlqry* function in MATLAB with a 0.02 second time step, which applies the LQG techniques discussed in Chapter II, Section 2.4.2, for discrete-time systems. The 18-state model was used in the calculation of this 4×18 matrix. This controller was designed before the model reduction effort described in the previous section. The same exercise was performed with the reduced model with results showing only gain increases on the rudder control. The numerical result for the truncated full-state model can be viewed in Appendix B. This gain matrix is only one component on the overall design of the setup outlined in the introductory paragraph in Chapter III; the inputs to the gain matrix are the states estimated by the Kalman filter.

3.3 Stochastic Estimation

Since the world is not purely deterministic, stochastic estimation is used to better estimate the true state errors. Because the helicopter is best modeled using a nonlinear system of equations, the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF) are both good candidates for implementation. Since several studies have indicated that the UKF provides the better esimtate of the two (as previously discussed in Chapter II), the UKF is the choice for this design; however, the EKF was implemented while troubleshooting the UKF, thus both designs were included and compared to verify the UKF's performance. The state estimator was designed and simulated separately from the LQR controller. This is possible due to the separation property, which states that for LQG controllers using an optimal state estimator (Kalman Filter), the "feedback control matrix is independent of all

uncertainty" [16]; therefore, the deterministic controller can be designed and tested separately from the Kalman Filter.

Whole-valued states, as opposed to error states, were initially used to verify the performance of each filter. Also, the sampling time, Δt, is set to 0.02 seconds, for a discrete-time application. Finally, the reduced-state model was used in place of the full 18-state model.

3.3.1 Extended Kalman Filter. Following theory and equations from Section 2.3.1, the EKF was coded in an m-file using MATLAB.

Propagation. Since the state vector propagates between updates using the linearized system model in Equation 3.11, the A matrix, previously calculated in Section 3.1.3, is used to convert the system model to discrete-time, using the equations listed in Table 2.2. Matrices Φ_k and Q_d are then used to propagate the covariance between measurement updates. Both the state estimate and covariance propagation equations are identified in Section 2.3.1 and listed, for convenience, below:

$$x(t_i^-) = x(t_{i-1}^+) + \int_{t_{i-1}}^{t_i} \delta\dot{x} dt \tag{3.17a}$$

$$P(t_i^-) = \Phi_k(t_i) P(t_{i-1}^+) \Phi_k^T(t_i^-) + Q_d \tag{3.17b}$$

where process noise intensity matrix is $Q = \Lambda(0.01\ \frac{N^2}{s}$, $0.01\ \frac{N^2}{s}$, $0.01\ \frac{N^2}{s}$, 5e-6 $\frac{N^2-m^2}{s}$, 5e-6 $\frac{N^2-m^2}{s}$, 5e-6 $\frac{N^2-m^2}{s}$). If a measurement update is not available,

$$x(t_i^+) = x(t_i^-) \tag{3.18a}$$

$$P(t_i^+) = P(t_i^-) \tag{3.18b}$$

Measurement Update. For the EKF, the measurement update encompasses three main activities: measurement prediction, Kalman gain calculation,

and state estimate and covariance update. The measurement (z) and measurement estimate ($\hat{z}$), for the case of El Toro, is simply the first 12 states of $x(t_i^-)$ because these are the only observable states. Next, the Kalman gain calculation is performed. Once again, these equations are located and described in Section 2.3.1; however, they are listed again below for convenience, where the measurement noise intensity matrix is $R = \Lambda(0.1\text{ m}^2, 0.1\text{ m}^2, 0.1\text{ m}^2, 0.01\ \frac{m^2}{s^2}, 0.01\ \frac{m^2}{s^2}, 0.01\ \frac{m^2}{s^2}, 0.01\text{ rad}^2, 0.01\text{ rad}^2, 0.01\text{ rad}^2, 0.01\ \frac{rad^2}{s^2}, 0.01\ \frac{rad^2}{s^2}, 0.01\ \frac{rad^2}{s^2})$:

$$P_{res}(t_i) = HP(t_i^-)H^T + R \tag{3.19a}$$

$$K(t_i) = P(t_i^-)H^T P_{res}(t_i)^{-1} \tag{3.19b}$$

The state estimate and covariance update below results when an update is available.

$$x(t_i^+) = x(t_i^-) + K(t_i)\left[z_{meas}(t_i) - \hat{z}(t_i)\right] \tag{3.20a}$$

$$P(t_i^+) = P(t_i^-) - K(t_i)HP(t_i^-) \tag{3.20b}$$

3.3.2 Unscented Kalman Filter. The UKF takes a more complicated approach to propagation and measurement update when dealing with nonlinearities within a system. Sigma points represent the statistics of the state vector, x, where χ_0 is the mean, and remaining 28 sigma points, χ_i, for the reduced-state model, capture characteristics of the pdf. The sigma point equations, Equations (2.18a) - (2.19c), were used and derived from three tunable parameters: α=0.25, β=2, and κ=0. These sigma points are used to propagate through the nonlinear function and calculate the corresponding propagated/updated mean and covariance.

Propagation. Once the 29 sigma points are generated, each column vector is propagated through the nonlinear system equation. Due to the "stiff-

ness" of the system model, the small integration steps had to be used to integrate the nonlinear system equations. A stiff system is one that refers to a system where the ratio of the largest eigenvalue (λ_l) divided by the smallest eigenvalue (λ_s) is much greater than 1.

$$\frac{\lambda_l}{\lambda_s} >> 1 \tag{3.21}$$

To provide this level of integration, MATLAB's *ode15s* was used to integrate the nonlinear system model differential equation during propagation using MATLAB code and also during all Simulink simulations performed in this effort.

Next, the process noise transition matrix was calculated. This matrix is time varying and is defined in terms of attitude angles: ϕ, θ, and ψ. These angles correspond to the seventh, eighth, and ninth values of $x(t_i^+)$ and will be used to calculate G during each time step. Furthermore, the system model $\dot{x}$ is linearized about the current state in order to calculate the state transition matrix, Φ. Using these matrices, the equations from Section 2.3.2 and Table 2.2 are used to calculate the propagated state estimate and covariance. These equations are summarized/repeated below for convenience, with i signifying the time step number.

$$\Phi(t_i) = e^{A(t_i)\,\Delta t} \tag{3.22}$$

$$Q(t_{i-1}) = \frac{1}{2}\left[\Phi(t_i)G(t_i)\;Q\;G(t_i)^T\Phi(t_i)^T + G(t_i)\;Q\;G(t_i)^T\right] \tag{3.23}$$

$$\hat{x}(t_i^-) = W_{0m}\chi_0(t_i) + \sum_{k=1}^{2L} W_{ukf}\chi_k(t_i) \tag{3.24}$$

$$\begin{aligned} P_{xx}(t_i^-) = {} & W_{0c}\left[\chi_0(t_i) - \hat{x}(t_i^-)\right]\left[\chi_0(t_i) - \hat{x}(t_i^-)\right]^T + \\ & \sum_{k=1}^{2L} W_{ukf}\left[\chi_k(t_i) - \hat{x}(t_i^-)\right]\left[\chi_k(t_i) - \hat{x}(t_i^-)\right]^T + Q_d \end{aligned} \tag{3.25}$$

Measurement Update. When a measurement update comes available (every 0.5 seconds), new sigma points are built using the same method mentioned during the discussion on propagation. These sigma points are then transformed through the measurement equation below to become Z:

$$Z = H\chi \tag{3.26}$$

The measurement equation directly observes the first 12 state variables; therefore, the output transition matrix, H, is a linear function. The remaining equations used during the update were discussed in more detail in Chapter II, Section 2.3.2, and repeated below for convenience:

$$\hat{z}(t_i) = W_{0m} Z_0(t_i) + \sum_{k=1}^{2L} W_{ukf} Z_k(t_i) \tag{3.27a}$$

$$P_{\hat{z}\hat{z}0}(t_i) = W_{0c} \left[Z_0(t_i) - \hat{z}(t_i) \right] \left[Z_0(t_i) - \hat{z}(t_i) \right]^T \tag{3.27b}$$

$$P_{\hat{z}\hat{z}}(t_i) = P_{\hat{z}\hat{z}0}(t_i) + \sum_{k=1}^{2L} W_{ukf} \left[Z_k(t_i) - \hat{z}(t_i) \right] \left[Z_k(t_i) - \hat{z}(t_i) \right]^T + R \tag{3.27c}$$

$$\begin{aligned} P_{xz}(t_i) = & \left[W_{0c}(\chi_0(t_i) - \hat{x}(t_i^-)), W_{ukf}(\chi_k - \hat{x}(t_i^-)) \right] * \\ & \left[W_{0c}(Z_0(t_i) - \hat{z}(t_i)), W_{ukf}(Z_k(t_i) - \hat{z}(t_i)) \right]^T \end{aligned} \tag{3.27d}$$

$$K(t_i) = P_{xz}(t_i) P_{\hat{z}\hat{z}}(t_i)^{-1} \tag{3.27e}$$

$$\hat{x}(t_i^+) = \hat{x}(t_i^-) + K(t_i) \left[z_{meas} - \hat{z}(t_i) \right] \tag{3.27f}$$

$$\hat{P}_{xx}(t_i^+) = \hat{P}_{xx}(t_i^-) - K(t_i) P_{\hat{z}\hat{z}}(t_i) K(t_i)^T \tag{3.27g}$$

3.4 Inertial Navigation

Looking back to Figure 3.3, the mechanization block to generate a nominal trajectory could use one of two methods: system model mechanization using throttle, rudder, pitch, and roll inputs, or INS mechanization using raw INS data from an accelerometer and gyro as inputs. The former mechanization is discussed in Sec-

tion 3.1, and can be described using the nonlinear equation and Simulink model block in Figure 3.9.

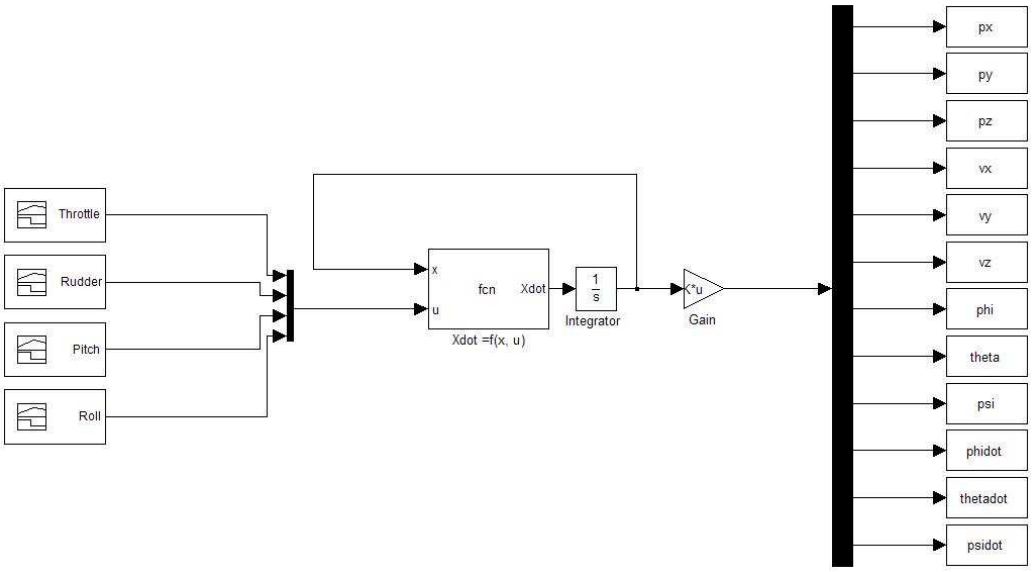


Figure 3.9: El Toro System Model Mechanization. The embedded Simulink function contains the nonlinear system model, which takes inputs and states to generate a trajectory along with attitude results.

This system model mechanization can be replaced using INS strapdown mechanization equations, resulting in differing levels of accuracy. As a note, code used for the explanation and simulation of Strapdown mechanization and INS error propagation equations was previously generated by the ANT Center, and only slightly modified for this effort.

3.4.1 Strapdown Mechanization. The application of a strapdown INS dictates the mechanization employed to derive position and attitude from raw INS data: Δv^b and $\Delta\theta^b_{ib}$, previously annotated as a^b and ω^b_{ib}. The mechanizations consider different coordinate systems and are chosen based on the application. The mechanization chosen for the INS mechanization block in this effort is the "local geographic navigation frame mechanization", which considers a rotating Earth, and references the local geographic navigation frame of reference discussed in Section 2.1 [30].

Because hover is the goal of this thesis, gravity is modeled as a constant. To determine the constant value of gravity, AFRL's MAV IFF was used as the most like area of operation. Google maps was used to determine the approximate location of the facility using WGS 84 information:

Latitude 39.79088 ° N

Longitude 84.0917 ° W

Ellipsoidal height 230 m

This coordinate will hereafter be referred to as the initial WGS 84 MAV location (P_{wgs}), which will not change when hovering. The gravity component was calculated using gravity model constants defined by WGS 84 and outlined in Table 3.4 [35]. Although this exercise is arguably irrelevant, it produced values that are operationally representative.

Table 3.4: Gravity Model Constants. These variables are used to calculate gravity using Equation 3.28.

Symbol	Definition	Value
$a1$	gravity model constant	9.7803267715
$a2$	gravity model constant	0.0052790414
$a3$	gravity model constant	0.0000232718
$a4$	gravity model constant	-3.0876910891e-6
$a5$	gravity model constant	4.3977311e-9
$a6$	gravity model constant	7.211e-13

The term g^n is the resulting gravity value. For the INS mechanization at the initial WGS 84 MAV position, g^n is calculated to be 9.8008 $\frac{m}{s^2}$ using Equation (3.28) [35].

$$\begin{aligned} g^n = a1\left[1 + a2 \times sin(P_{wgs}(1))^2 + a3 \times sin(P_{wgs}(1))^4\right] + \\ \left[a4 + a5 \times sin(P_{wgs}(1))^2\right] P_{wgs}(3) + a6 \times P_{wgs}(3)^2 \end{aligned} \tag{3.28}$$

Now that gravity has been calculated, the position and gravity vectors are transformed to the ECEF frame of reference for the remaining calculations. After the initial WGS 84 MAV location is converted to the ECEF frame (P^e), the raw INS data, Δv^b's and $\Delta\theta_{ib}^b$'s, are used to calculate acceleration (a^n), velocity (v^n), position (p^n), attitude (θ_{nb}^n), and attitude rates (ω_{nb}^b). First, the biases (a^b = accelerometer bias; b^b = gyroscope bias), defined in the INS specifications, have to be removed in the following equations, where $\Delta t = 0.02$ seconds:

$$\Delta v^b = \Delta v^b - \Delta t\ a^b \tag{3.29a}$$

$$\Delta\theta_{ib}^b = \Delta\theta_{ib}^b - \Delta t\ b^b \tag{3.29b}$$

If both biases are zero, then the raw INS data will not change. To propagate the attitude at a 50 Hz rate, first the initial MAV location and the current MAV location are combined to create a new vector representing the direction and distance from the center of the Earth to the current location. This vector is updated to account for the rotation of the Earth within 0.02 seconds, and the change in attitude sensed by the gyroscope and given by $\Delta\theta_{ib}^b$. The attitude rates are then calculated to produce the Euler angles in vector form (θ_{nb}^b):

$$\omega_{nb}^b = \frac{\theta_{nb}^b}{\Delta t} \tag{3.30}$$

Next, to propagate the acceleration, the following equation is calculated using q_b^n to convert δv to the navigation frame and the skew symmetric matrix of the Earth's rate in the navigation frame, Ω_{ie}^n, as shown below. Using the propagated acceleration, the velocity and position is easily derived as shown in Equations (3.31a) - (3.31c).

$$a_n = \frac{\Delta v^n}{\Delta t} - g^n - 2\Omega_{ie}^n v_{prior}^n \tag{3.31a}$$

$$v_n = v_{prior}^n + \frac{\Delta t}{2}\left(a_{prior}^n + a_{current}^n\right) \tag{3.31b}$$

$$p_n = p_{prior}^n + \frac{\Delta t}{2}\left(v_{prior}^n + v_{current}^n\right) \tag{3.31c}$$

This mechanization was coded using an S-function in Simulink.

3.4.2 INS Error Propagation. For the final state estimation configuration, instead of the actual state estimation values being propagated, the state error will be propagated in the UKF; therefore, an INS error propagation model is used in place of the El Toro system mathematical model. The error propagation model was previously derived and used for this effort [33]. The state vector, δx used in INS is defined by 15 parameters: the position ($p^n = [x, y, z]^T$), velocity ($v^n = [\dot{x}, \dot{y}, \dot{z}]^T$), attitude ($\theta_b^n = [\phi$, θ, and $\psi]^T$), and INS biases ($a^b = [a_x^b, a_y^b, a_z^b]^T$ and $b^b = [b_x^b, b_y^b, b_z^b]^T$).

$$\delta x = [\delta x,\ \delta y,\ \delta z,\ \delta\dot{x},\ \delta\dot{y},\ \delta\dot{z},\ \delta\phi,\ \delta\theta,\ \psi,\ \delta a_x^b,\ \delta a_y^b,\ \delta a_z^b,\ \delta b_x^b,\ \delta b_y^b,\ \delta b_z^b]^T \tag{3.32}$$

Furthermore, the stochastic inputs into this model, otherwise known as process noise, will change from the vector described in Section 3.1.4 to be as follows, with w_a^b, w_b^b, $w_{a\_bias}^b$, and $w_{b\_bias}^b$ representing accelerometer noise, gyroscope noise, accelerometer bias, and gyroscope bias, respectively [33]:

$$w = [w_a^b, w_b^b, w_{a\_bias}^b, w_{b\_bias}^b]^T \tag{3.33}$$

As a result, the state space INS error model is defined with respect to δx and w [33]:

$$\delta\dot{x}=\begin{bmatrix} 0_3 & I_3 & 0_3 & 0_3 & 0_3 \\ C_e^n G C_n^e & -2C_e^n\Omega_{ie}^e C_n^e & (f^n\times) & C_b^n & 0_3 \\ 0_3 & 0_3 & -(C_e^n\omega_{ie}^e)\times & 0_3 & -C_b^n \\ 0_3 & 0_3 & 0_3 & -\frac{1}{\tau_a}I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & -\frac{1}{\tau_b}I_3 \end{bmatrix}\delta x+\begin{bmatrix} 0_3 & 0_3 & 0_3 & 0_3 \\ C_b^n & 0_3 & 0_3 & 0_3 \\ 0_3 & -C_b^n & 0_3 & 0_3 \\ 0_3 & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & I_3 \end{bmatrix} w \tag{3.34}$$

where I_3 is a 3×3 identity matrix, and 0_3 is a 3×3 zero matrix. The deterministic portion of this model is used to propagate sigma points within the UKF when using INS mechanization to determine position, velocity, attitude and attitude rates, with camera measurement updates.

3.5 System Model and INS Combination

The crux of this effort is to devise a plan to integrate the INS and system model to provide more accurate states for feedback control. Up to this point, a system model, an INS mechanization model, and an INS error model have been defined. Figure 3.3 shows the system block diagram either using the system model or INS model for the mechanization block, in addition to the appropriate models used for Kalman filter propagation. The intent for combining the two designs is to extract information from both models to produce a more accurate solution. Figure 3.10 adds additional detail to Figure 3.3 by providing a graphical depiction of the final design. This setup allows the user/engineer to select a model-only, INS-only, or a combination configuration. The control signal is sent to the UKF and based on its value, directs trajectory information from either the nonlinear system model, INS mechanization, or combination block. The challenge is to produce algorithms to combine the two trajectories and to combine the propagated state error and covariance in the UKF.

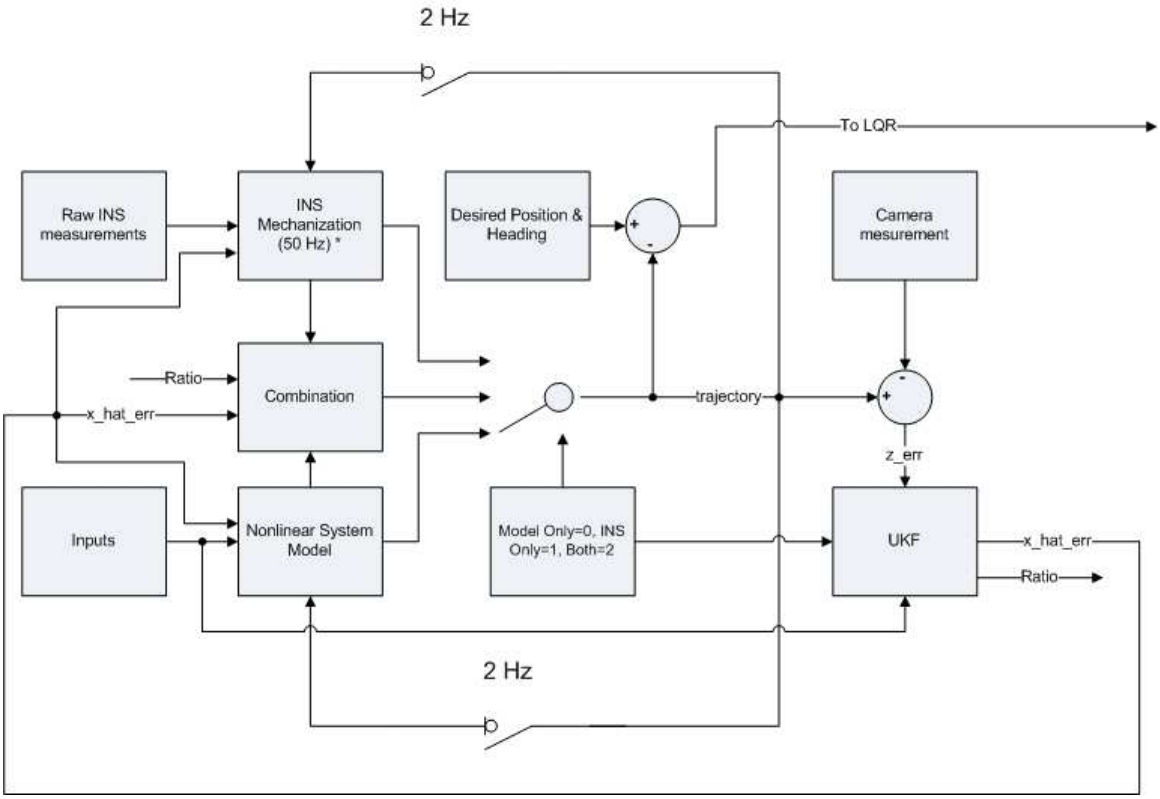


Figure 3.10: Final Design with System Model and INS Combination Selectable. Measurements and inputs are provided in an effort to predict the system state errors for the LQR gain block. The LQR gain block will provide the optimal control to the helicopter.

Combining information is not new to the field of controls and stochastic estimation. For instance, Federated Kalman filters (FKF) provide centralized state estimation using many sensors by combining state estimates from several "local" Kalman filters. Each Kalman filter estimates the system states from a different sensor measurement update, and is weighted by the associated inverse covariance. This method allows the setup with the lowest uncertainty/most information to have the highest weight in the overall state estimate. Furthermore, the covariance from each filter is also combined to produce the corresponding covariance for the combined state estimate. Maybeck also covers the concept of combining state estimates and covariances in a simple "lost at sea" example in the introduction of his first book [17]. This ap-

proach was adapted for this effort. The final equations use information matrices, and are adapted directly from the use of FKFs [5]:

$$P_c^{-1} = P_{ins}^{-1} + P_{sys}^{-1} \tag{3.35a}$$

$$\hat{x}_c = P_c \left(P_{ins}^{-1} \hat{x}_{ins} + P_{sys}^{-1} \hat{x}_{sys} \right) \tag{3.35b}$$

where P^{-1} is the inverse covariance, or otherwise known as the information matrix, and the subscript "c" represents the combination configuration, while "sys" and "ins" represent the values for the model-only and INS-only configurations, respectively. Referring to Figure 3.10, the source of the trajectory is determined by a control signal whose value ranges from 0 to 2. Table 3.5 outlines the control signal value and its corresponding configuration and calculation. Each configuration adjusts its nominal state vector by subtracting the error state estimate ($\delta\hat{x}$). If control signal 2 is selected, the "Ratio" signal from the UKF is used to weight the nominal trajectories coming from the system model and INS mechanization blocks.

Table 3.5: Possible Controller Configurations. The configuration of the controller is determined by the control signal value, selected by the user. The selection determines the nominal trajectory of the system.

Control Signal	**Configuration**	**Equation**
0	Model ONLY	$\bar{x} = \bar{x}_{sys} - \delta\hat{x}$
1	INS ONLY	$\bar{x} = \bar{x}_{ins} - \delta\hat{x}$
2	Combination	$\bar{x} = \bar{x}_{sys}\,(\text{Ratio}) + (1 - \text{Ratio})\,\bar{x}_{ins} - \delta\hat{x}$

The Ratio signal from the UKF is a new output of the filter and is only used when the two methods are integrated. The new output is calculated during propagation, which changes significantly with the combination method. Since there is only one measurement provided, the measurement update section in the code does not change. Several changes to the filter need to be made to accommodate the new method. The first is to combine the state vectors. The system model has 14 states and the INS

error model has 15 states, with only 9 states in common; therefore, the resultant combined vector has 20 states:

$$\delta x = \left[x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}, FB_{\phi}, FB_{\theta}, ab_x, ab_y, ab_z, bb_x, bb_y, bb_z\right] \tag{3.36}$$

This also guarantees an increase in the covariance matrix, P, to a 20×20. When the control is set to 0 or 1, the extra states for that model are defaulted to zero, and the corresponding uncertainty is set exceedingly high. However, when the control is set to 2, all error states are populated and updated. To combine state vectors and covariances, the state vector is propagated using the system model and INS model separately, then combined using the following equations:

$$P_c^- = \left[\left(P_{ins}^-\right)^{-1} + \left(P_{sys}^-\right)^{-1}\right]^{-1} \tag{3.37a}$$

$$\text{ratio} = P_c^- \left(P_{sys}^-\right)^{-1} \tag{3.37b}$$

$$\delta\hat{x} = P_c^- \left(P_{sys}^-\right)^{-1} \delta\hat{x}_{sys} + P_c^- \left(P_{ins}^-\right)^{-1} \delta\hat{x}_{ins} \tag{3.37c}$$

$$\text{Ratio} = \Lambda(\text{ratio}) \tag{3.37d}$$

This process is also depicted graphically in Figure 3.11. The last equation listed is the additional output supplied by the UKF and provided to the control signal block in Figure 3.10. Each step of the design process was tested through simulations, and the controller was hardware tested independently of the stochastic estimation configuration using AFRL's MAV IFF. These steps are discussed in detail in the next chapter.

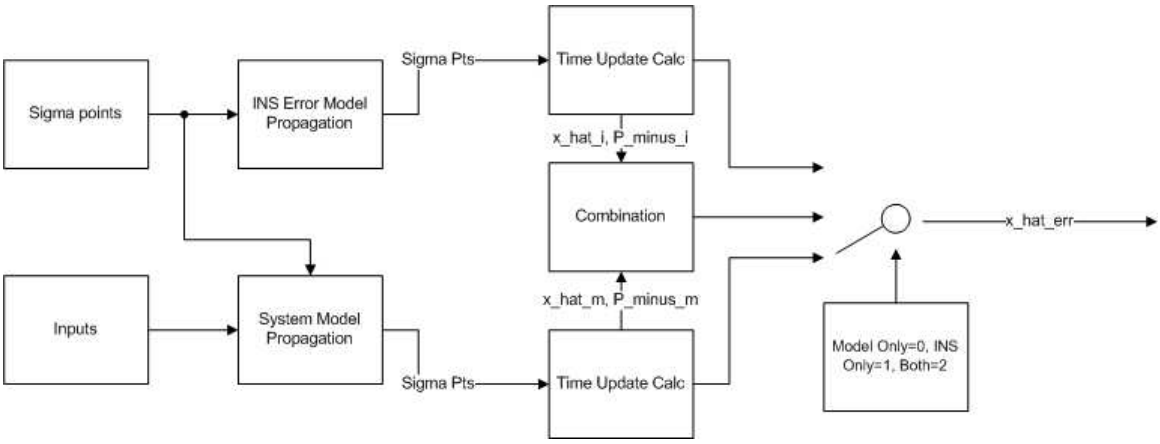


Figure 3.11: UKF Combination Propagation. The inputs and previous error state vector (to produce sigma points) are fed into the propagation section. The sigma points are propagated through the nonlinear system model and INS error model separately, and depending on the control signal, then combined for use in the measurement update section.

IV. Results and Analysis

BEFORE conclusions can be made on the design outlined in Chapter III, it must be tested through simulation and hardware tests. The results of these tests will be analyzed, and through this analysis conclusions will be made. Several aspects of testing will be considered during this chapter:

- **Standardization**: When comparing two methods, the setup will be replicated, along with deterministic and stochastic inputs, for the same amount of time.
- **Repeatability**: This aspect considers performing the same test with varying random inputs. Each test should support the same conclusions established from analysis.
- **Operationally Representative**: The intensity of the random inputs must be within an expected range viewed from testing of the physical plant.

Each step of the design process will be validated through simulation. The LQR controller will also be validated with the hardware (El Toro) through use of the Vicon system located in AFRL's MAV IFF. The testing in this section will follow the design steps in Chapter III: El Toro's system model will be validated and reduced; the process noise matrix, G, will be verified; the EKF and UKF filters will be compared; strapdown INS mechanization will be tested; and, finally, all parts of the controller will be integrated and tested under different configurations using a Monte Carlo analysis.

4.1 El Toro System Model

The LQR is a model-based controller. Since the system model is a requirement for designing an LQR, it is necessary to create and verify the system model before the controller is built. The nonlinear model identified for the helicopter in Chapter III is tested by varying the inputs in simulations while monitoring the system response. The system model should show the helicopter translating as expected given a range of throttle, rudder, pitch, and roll commands. Stochastic inputs were also varied to

obtain the operationally representative process noise intensity levels. The test setup for this open loop testing is shown in Figure 4.1.

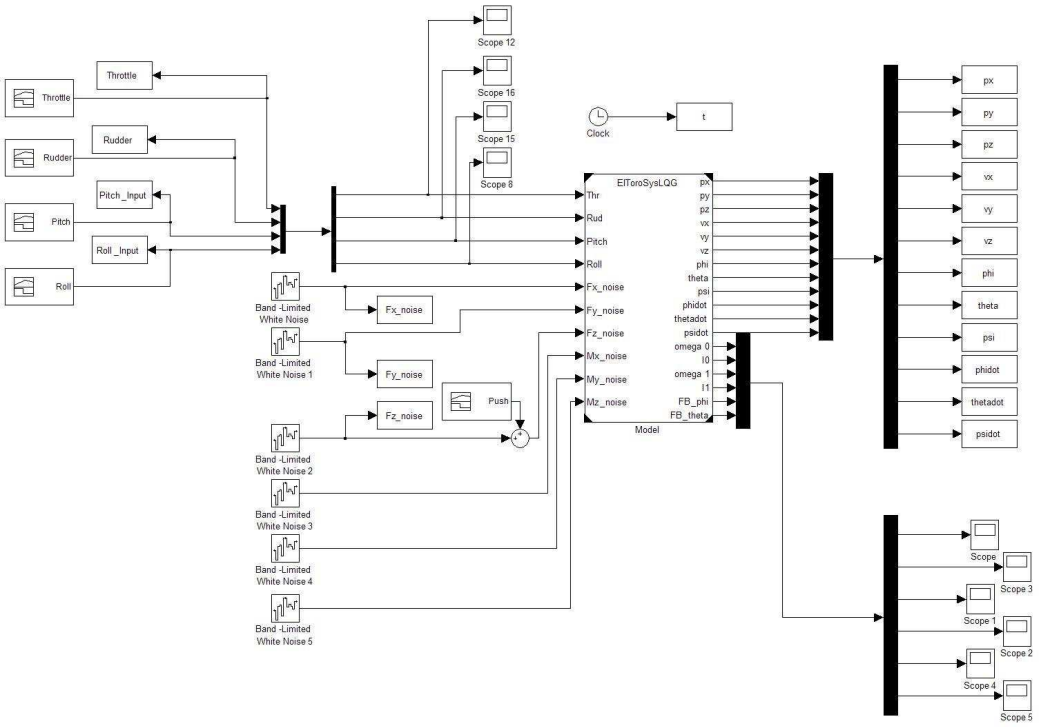
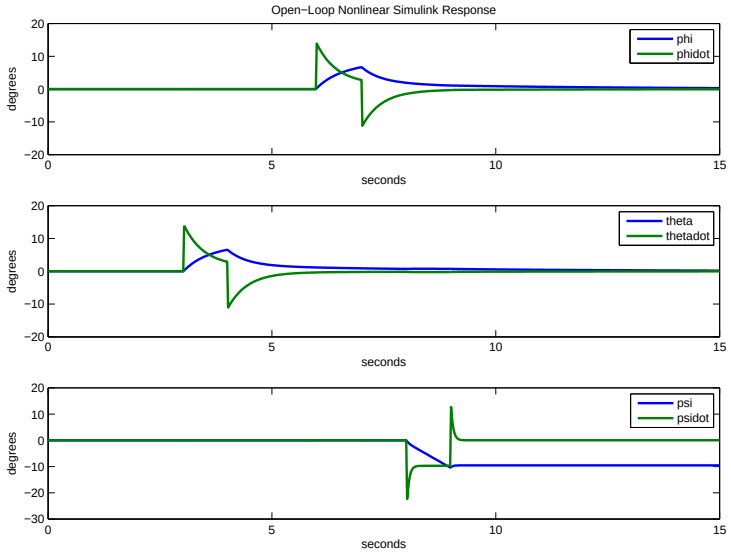
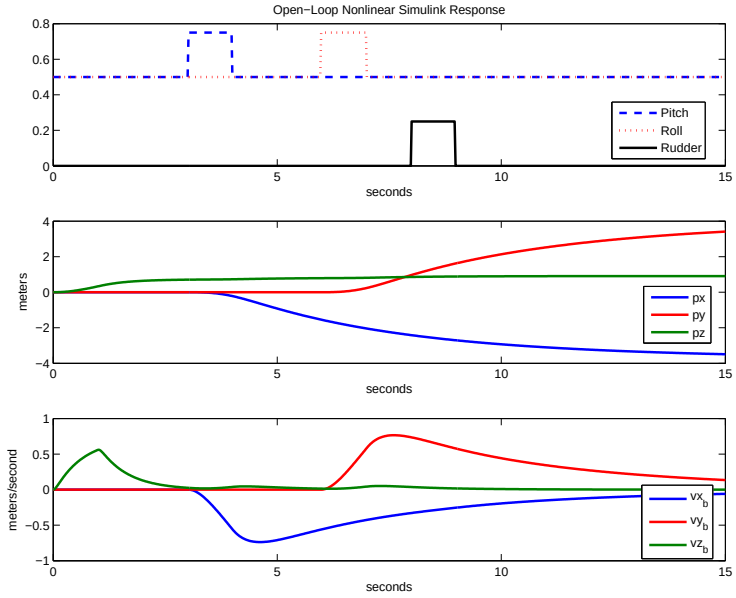


Figure 4.1: Simulink Open Loop Testing. This setup is used to test the response of the Simulink model while varying the deterministic and stochastic inputs.

The results of the simulation are described referencing the results shown in Figure 4.2. To begin, the throttle was increased to a level to simulate hover at approximately one second. Up until one second, the vehicle is falling, but as the throttle is increased, the vehicle starts to level out. Next, a pitch command is provided at approximately three seconds for the duration of one second. In the NED frame of reference, θ increases as expected, and the vehicle begins to translate in the x-direction (px). Next, a roll command is provided at approximately six seconds for a duration of one second. Once again, ϕ increases as expected, and the vehicle begins to move in the positive y-direction (py). Finally, a positive rudder is applied at approximately eight seconds for a one second duration. The angle ψ goes negative as expected from the helicopter dynamics. Note all noises and disturbances were set to zero during this simulation.



(a) Attitude and Attitude Rates



(b) Position And Velocity

Figure 4.2: Open Loop Simulation. A variety of control inputs were supplied to the helicopter system model to analyze the response. The system model responded as expected with changes in pitch, roll, and rudder.

4.1.1 Model Verification. Although the system model roughly responds as expected, how representative the mathematical model is of the system is determined through model verification. AFRL's MAV IFF Vicon system is instrumental in this process by providing a means of recording the inputs to the helicopter and its response in terms of position, velocity, attitude, and attitude rates, at a 50 Hz rate. This data was stored in a comma-separated-variable (CSV) file, and read into MATLAB for analysis. The technique used to verify the system model is the cross-validation test [22] [28]. The cross-validation test verifies, given the same inputs, the physical plant (El Toro) and mathematical model provide like outputs. The cross-validation test is performed in Simulink. The CSV file is first read into MATLAB using the *csvread* function. Each row in the resulting matrix signifies a different measurement (i.e., time, x, $\dot{x}$, ϕ, etc.), and is assigned a variable name. The time vector is recorded in milliseconds, but does not start at zero; therefore, the first time value is subtracted from the entire vector, then the vector is converted from milliseconds to seconds. Likewise, AFRL's Vicon system records position and velocity in millimeters, so a millimeter-to-meter conversion was employed to standardize the units for post-processing activities later performed in MATLAB. Finally, the Vicon coordinate system is a right-hand coordinate system with z pointing up. The following DCM was used to transform position data from the Vicon to the navigation frame of reference. The heading data also encountered a polarity change.

$$C_v^m = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{4.1}$$

These calculations were eventually included in Labview and became transparent to the user, so future recordings did not require manipulation. The Labview Virtual Instrument (VI) hierarchy describing this setup using Vicon is referenced later in Section 4.2.2. The recorded position, velocity, attitude, and attitude rates are now ready for comparison with the nonlinear model in Simulink. During simulation, the

recorded inputs are fed in from the workspace into Simulink at a 50 Hz rate. Since the range of all inputs provided by Vicon to the helicopter remote-control (-1 to +1) is different than the expected inputs of the system model, a conversion block was built to scale the inputs to a typical range and add trim settings. The overall setup is shown in Figure 4.3.

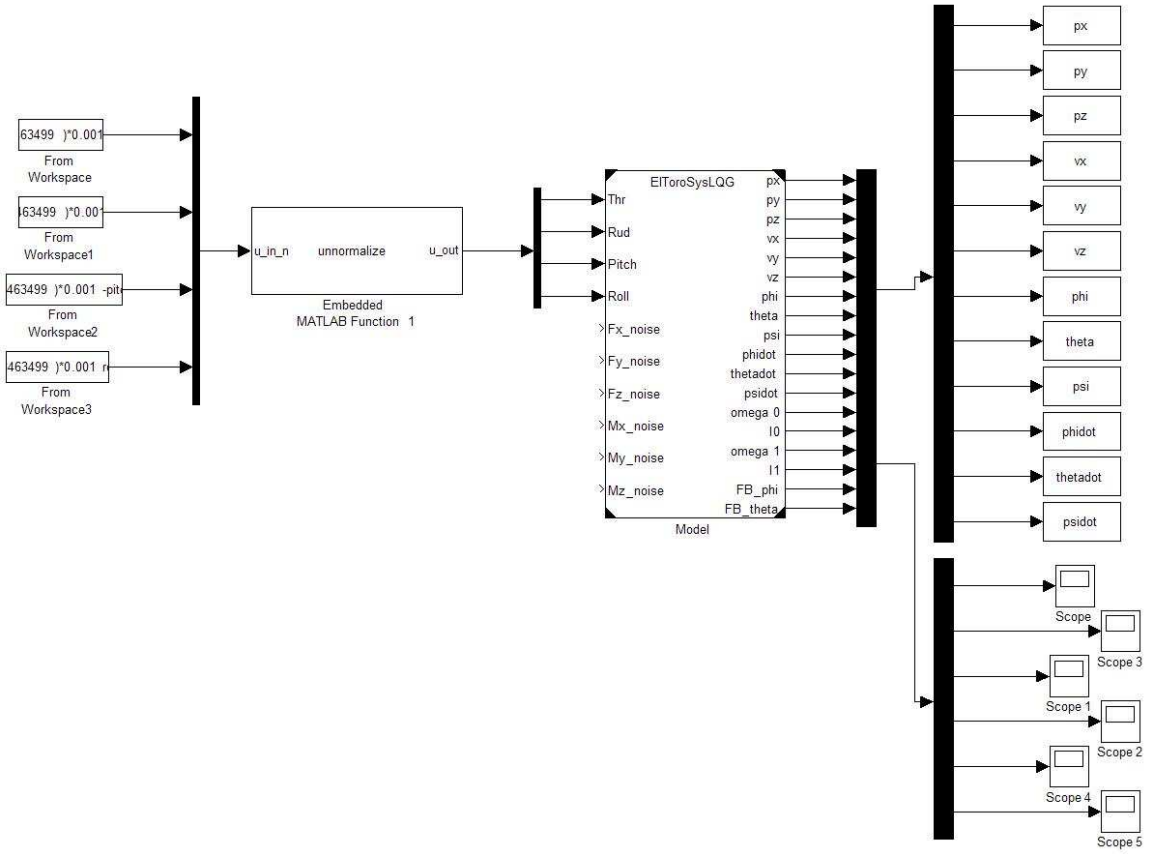


Figure 4.3: Simulink Setup for Model Validation Response Test. The recorded inputs are fed from the MATLAB workspace to the Simulink Model at a 50 Hz rate. The "unnormalized block" adds the trim settings and scales the inputs. The response is then recorded to the workspace for analysis.

After running and comparing the responses, it is noted the rates seem to provide a better indication of the suitability of the system model. The noise on the recorded input commands causes the position response of the nonlinear model to be grossly exaggerated. Furthermore, the nonlinear model doesn't account for the ground during take-off, so the result looks like the vehicle is falling. The rates have shown to give

a good indication of the response of the vehicle without being distracted with these issues. Looking at Figure 4.4, the system model's response resembles the helicopter's response. An anomaly occurred in all directions at approximately 15 seconds in the Vicon recording that did not translate to the mathematical model's response. Furthermore, the system model does not account for the floor (Earth's surface) stopping a fall of the vehicle in the positive z-direction. Based on these observations, along with further tests, the helicopter appears to be more sensitive to input changes, while the system model is more sensitive to biases. The frequency (0.2 Hz) shown in the velocity plots corresponded to the closed-loop system using a PID controller to control the vehicle during recording. This control setup caused El Toro to fly in small circles while trying to hover. A closer examination of the closed-loop system could reveal the cause; however, since the purpose was to record the response given an input, the varying in the controls proved to be a better analysis. Overall, through visual inspection, the plotted responses for both the actual plant and the model are similar enough with velocity and attitude rates to call "good". The ultimate test on whether the model is considered good is whether "the regulator based on this model will give satisfactory control" [13]; this will be tested in the next section. Until then, the biggest issues projected in future work is the slight correlation between throttle and rudder noted but not sufficiently captured in the system model, and the accumulation of battery drain's effect.

4.1.2 Model Reduction. The current identified system model has 18 states. Due to the complexity of the full nonlinear system model equation, four states were investigated for removal in efforts to improve the efficiency of the Kalman filter integration. From the 18 states, the first 12 (position, velocity, attitude, and attitude rates) and the flybar effect were difficult to model and significantly changed the response. Four states, however, could be easily removed with acceptable changes in response. These states are the two motor armature currents (I), and two motor angular velocities (ω). As discussed in Section 3.1.2, these four states were removed and

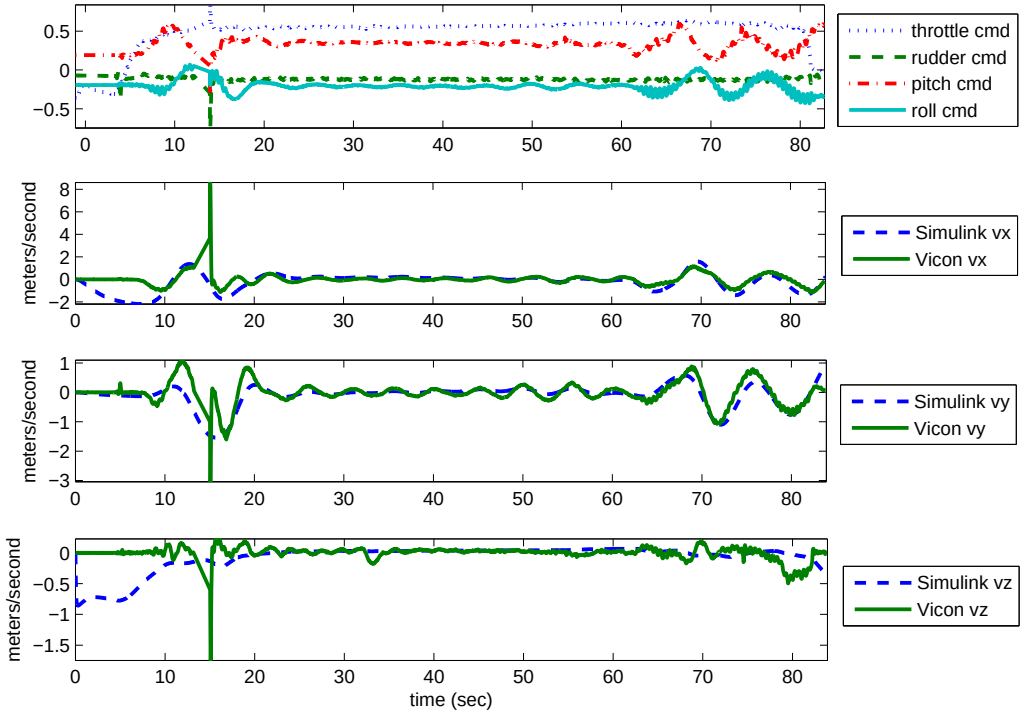


Figure 4.4: Response Test Velocity Results. With inputs varying between +1 and -1, the physical plant and system model's responses show a reasonable degree of commonality.

equations approximating I and ω (Equations (3.10)) were substituted within other state equations. The Simulink model and mathematical model were modified to reflect the changes. The responses of both of these models matched using the same varying inputs, verifying no implementation errors. The new 14-state model response was then compared to the full-state model response through simulations using Simulink. The outputs of both were subtracted to provide the error due to the approximation. The position, velocity, attitude, and attitude rate errors were plotted, as shown in Figures 4.5 and 4.6. Notice translation in the z-direction is notably different between the two; however, this difference was deemed acceptable since hover is the true goal.

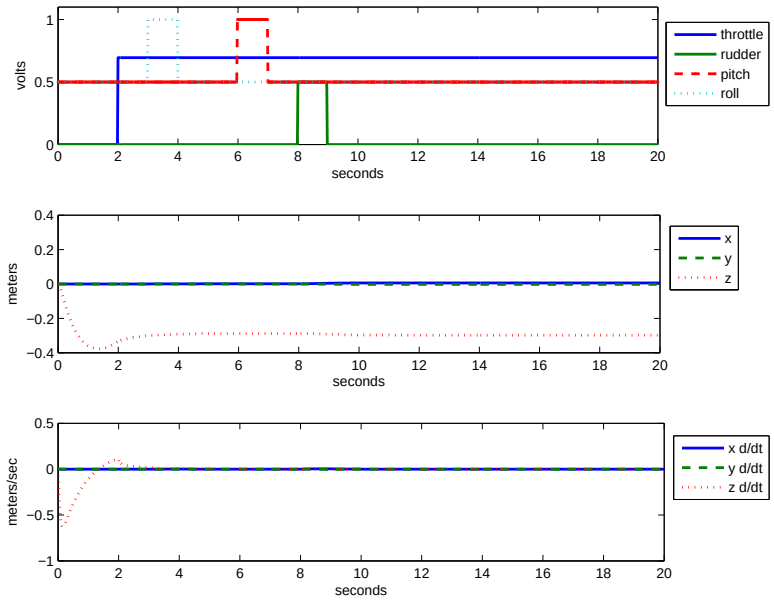


Figure 4.5: Position and Velocity Error Due to Model Reduction. The four inputs provided to the full-state and reduced model over a 20 second period. The difference between the two responses (error) shows acceptable error.

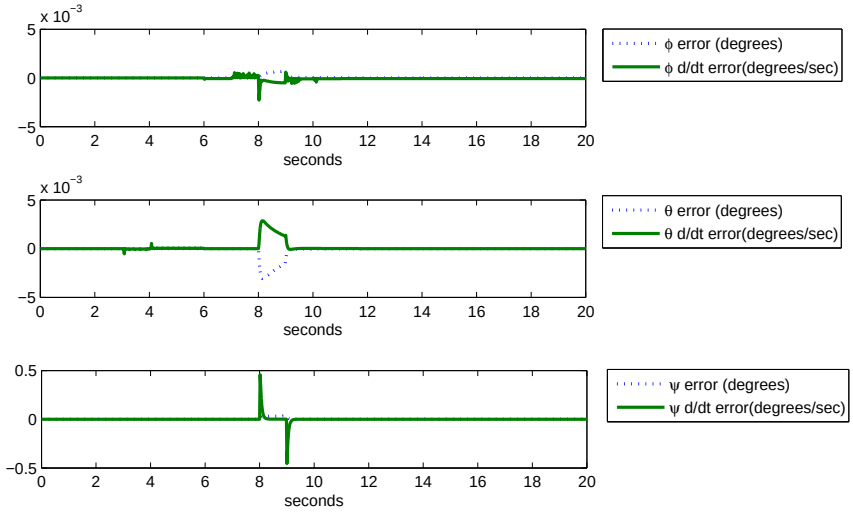


Figure 4.6: Attitude and Attitude Rate Error Due to Model Reduction. The four inputs provided to the full-state and reduced model over a 20 second period. The difference between the two responses (error) is plotted.

4.1.3 Stochastic Noise Insertion. From Section 3.1.4, noise was inserted into the system to account for inaccuracies within the model. An update accounting for these inputs was made to the to the mathematical model, $f(x(t), u(t), w(t))$. The updated math model was verified by comparing the output of the nonlinear mathematical model and the Simulink model when all inputs, deterministic and random, are the same. The responses were identical. Furthermore, the insertion of the process noise at the forces and moments adequately models the real effects, with the real effects being characterized over many runs. Over time, dynamics of the observable states are not defined as ergodic or stationary. In other words, although hover is the ultimate goal, if maneuvers are performed, the ensemble statistics do not resemble the temporal statistics; also, although a steady state error affecting position and heading is realized through hardware testing (later discussed), this error is not random, but more like a drift over many runs. This drift is accounted for through trim settings, further discussed in the next section.

4.2 LQG Controller

The Linear Quadratic Gaussian (LQG) controller uses a Linear Quadratic Regulator (LQR) to provide feedback to the helicopter based on a state error estimate. To verify the LQR design outlined in Section 3.2, simulations are performed by integrating the controller in a feedback loop with the system model using Simulink, and hardware tests are performed by integrating the controller with the Vicon system at AFRL's MAV IFF to control the El Toro helicopter. To help mitigate the risks associated with hardware tests, the simulations are performed first.

4.2.1 Simulations. The LQR controller consists of a gain matrix in a negative feedback loop. The basic setup for simulations is shown in Figure 4.7. This part of the controller test was performed to simulate a hover condition with noise and disturbances in preparation for the hardware test. Unfortunately, since testing with the Vicon system is limited to only controller testing without state estimation, the

gain matrix was tested with a varying number of states with the goal of reducing it to a 4×12 matrix. This size matrix only considers the first 12 states, which are the only directly observable states in the system. The gain was calculated using the 18 and 14-state models, then finally truncated to a 4×12 matrix. Several iterations were performed to test the full-state gain, 14-state gain, and truncated 12-state gain. It was found that the last six states are not required to maintain the vehicle at hover with low process noise and disturbances in all three directions. The simulation results from open-loop and closed-loop configurations were compared to show the influence of the controller using only 12 states, shown in Figure 4.8.

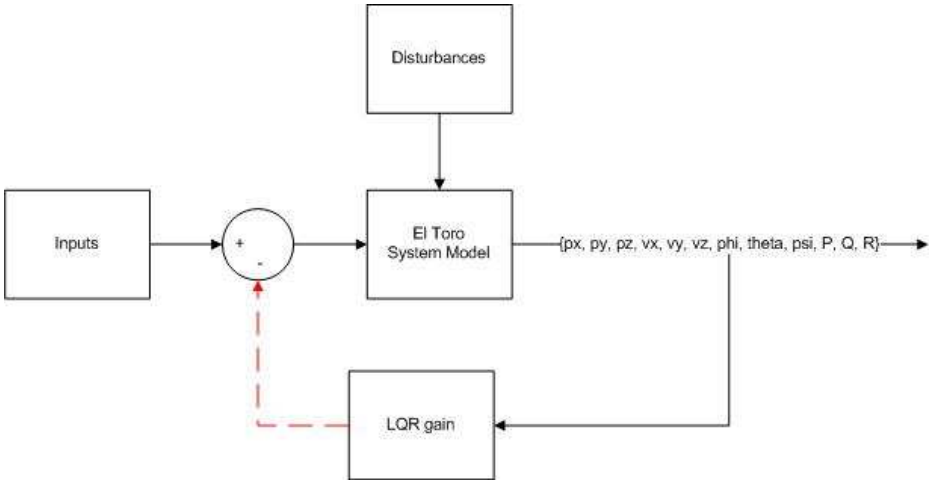
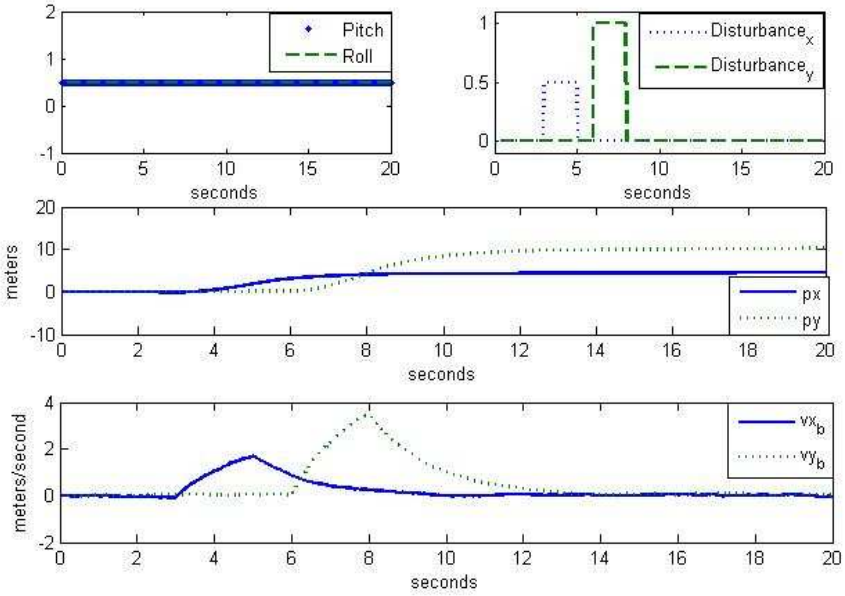
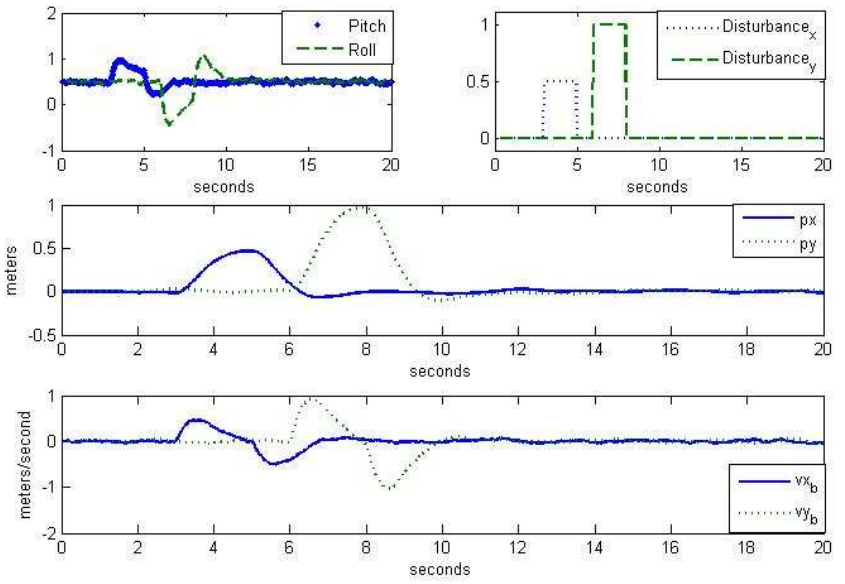


Figure 4.7: Close-Loop Controller Simulation. The first 12 states are fed back and multiplied by the LQR gain to adjust the input in efforts to maintain a hover at position (0,0,0).

In the open-loop configuration, when a external "push" is applied in the x and y-direction, the vehicle does not return to the original position; however, when the loop is closed as shown in Figure 4.7, the vehicle moves slightly during the applied disturbance, but like a spring it moves back to the desired position (0,0,0) when the disturbance goes to zero. The heading was also observed with like results. The outcome justified the use of the truncated matrix during the upcoming hardware test.



(a) Open-Loop Simulation



(b) Closed-Loop Simulation

Figure 4.8: Controller Validation. A series of inputs were supplied to the open loop system and the closed loop setup to determine the effectiveness of the controller design. The closed loop system shows effective disturbance rejection.

4.2.2 Hardware Test. Now that the controller is verified through simulations, the truncated LQR gain matrix is taken to the MAV IFF and integrated in the Vicon system using Labview. Several special efforts were made to make the controller work with the actual system:

- **Apply trim settings**: Duplicate the trim settings set in the remote control used to stabilize drift in position and heading.
- **Scale inputs**: The output of the LQR gain matrix was scaled down due to the differences between the system model and the Labview predefined limits of the inputs going to the PPM generator.
- **Adjust coordinate system**: The position data relative to the IFF's established coordinate system did not use the NED frame of reference; therefore a transformation had to be performed.
- **Filter position and attitude**: Filters were used to filter noisy measurement coming from the Motion Capture Cameras.
- **Mix pitch and roll commands**: The commands controlling the swashplate are actually sent to two servos (S_1 and S_2). Each servo is not purely a pitch or roll servo. These servos act in unison to perform pitch and roll maneuvers; therefore, some mixing is required to provide the correct input to these servos. This mixing can vary depending on the vehicle make and model.
- **Calculate error**: The error was calculated by differencing the recorded position and heading from the desired position and heading.

These compensations were captured in a hierarchy of VIs. Each VI represents a level of software computations to process data. The VI containing the LQR controller is shown in Figure 4.9. The 12 observable states are captured by 36 cameras placed in the IFF's flight test room, subtracted from the desired position and heading to create errors, filtered, then multiplied by the LQR gain matrix to provide negative feedback control.

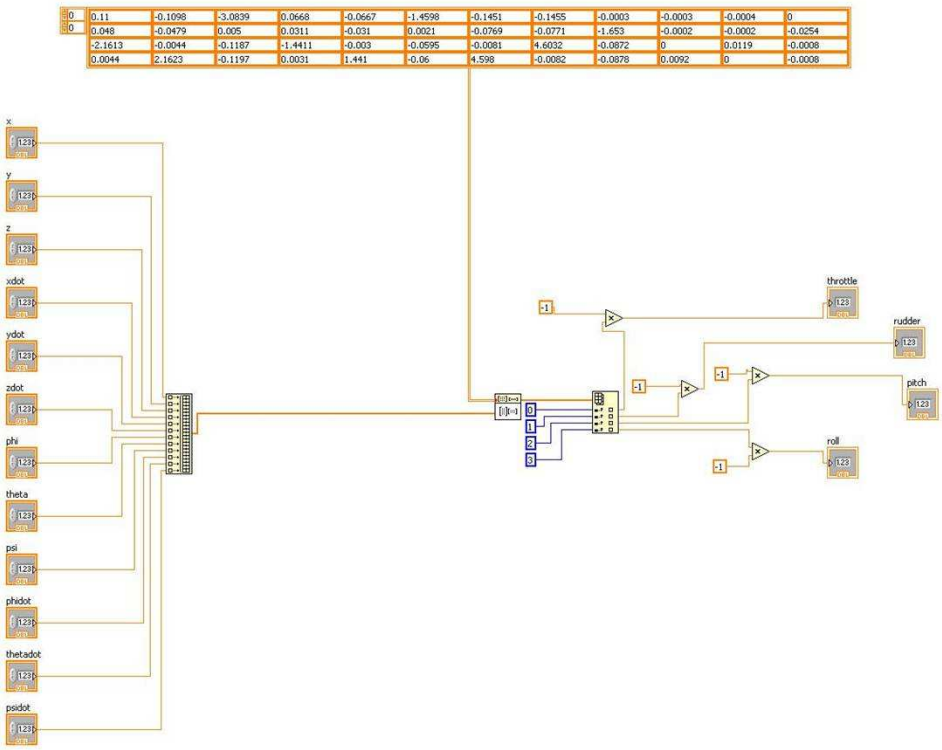


Figure 4.9: Controller VI. The LQR gain matrix is integrated in Labview to control El Toro using the Vicon system at the IFF.

The throttle, rudder, pitch, and roll commands are then sent to the Mixer VI to be mixed and scaled before being routed to the PPM Generator as shown in Figure 2.9. The mixing and scaling is captured in Equations (4.2a) - (4.2d), where S_1 and S_2 are the inputs to the individual servos, and the biases represent the current trim setting.

$$u_1 = 0.085(u_1) + 0.2125 \tag{4.2a}$$

$$u_2 = 0.8(u_2) + 0.1605 \tag{4.2b}$$

$$S_1 = 0.35(u_3 - u_4) - 0.0017 \tag{4.2c}$$

$$S_2 = -0.35(u_3 + u_4) + 0.0083195 \tag{4.2d}$$

This configuration was tested with the same helicopter over several weeks. The results of the controller hardware testing were consistent; the controller provided inputs to El

Toro that concluded in stable flight. However, the trim settings did require periodic tuning to reduce the steady state error in position and heading.

Each hardware test was accomplished in steps. First, personnel assigned to the IFF held the vehicle while the LQR controller was engaged. The desired z-position was set to a level that corresponded to half-throttle. The person holding the vehicle moved around the room to determine if the swashplate was moving in the direction corresponding to the reference point. Next, with the desired position set to (0,0,-1 meter) with a 0 degree heading, the vehicle was allowed to hover just above the person's hand. Once it was established that the vehicle could hover, the person let go of the vehicle. This process eventually progressed to slow take-offs and landings, and slow translations along the x and y axes. Finally, disturbances were injected by physically pushing the vehicle away from its desired position and heading, seen in Figure 4.10. Each time, the vehicle converged back to the reference point, minimizing the error between the desired position and its actual location.

Figure 4.10: Hardware Test at the IFF. The IFF consists of a flight test room and a control room with a window separating the two. The LQR regulator is controlling El Toro while manually injecting disturbances.

The trajectory of one run lasting 264.84 seconds is recorded and plotted (Figure 4.11). Samples of the inputs from the controller, actual position of El Toro, and the desired position (determined by the manipulation of scroll bars in Labview) is captured at a 50 Hz rate. This diagnostic data is used to calculate error (actual minus desired) to determine the accuracy of the controller (Figures 4.13 - 4.16). Upon closer examination, the error signals seem to be characterized by a low frequency corresponding to a 3 to 7 second period. To better understand the frequency content of the error signals, a frequency analysis was performed using MATLAB's *fft* function to generate a single-sided amplitude spectrum (example shown in Figure 4.12). As expected, there is a DC component that appears to correlate with the steady state error in position and attitude. In addition, the error signals are not purely white; it actually resembles a exponential signal with the peak at 0 Hz. The dominant frequency components for each position and heading for one data run are listed in Table 4.1.

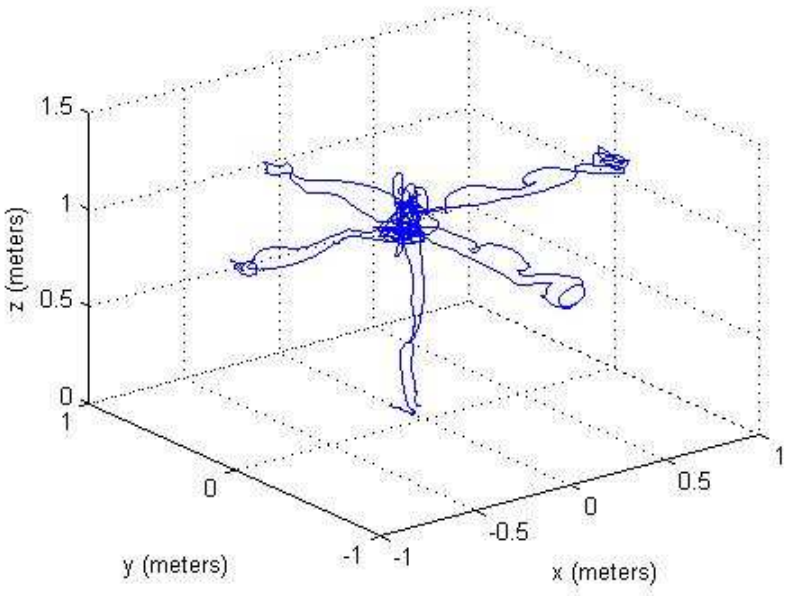


Figure 4.11: Trajectory of El Toro during Hardware Test in the IFF. Although hover is the main focus, the desired position was moved inside the Lab to test the effectiveness of the controller. This trajectory represents the actual trajectory of the vehicle inside 300 seconds.

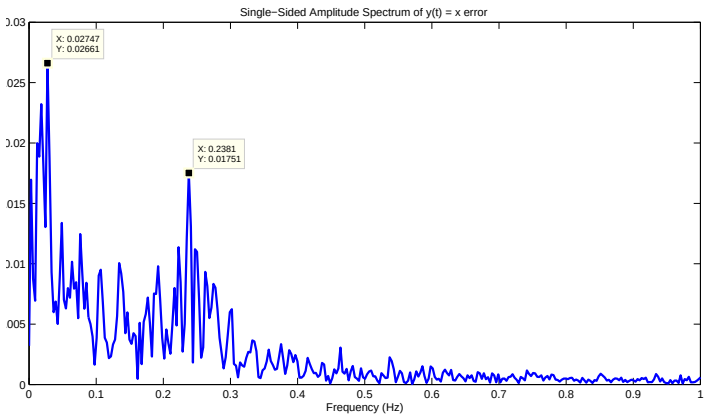


Figure 4.12: Single-Sided Amplitude Spectrum of the x Error. The dominate frequencies are shown to be 0.02747 and 0.2381 in one data run lasting approximately 240 seconds.

Table 4.1: Frequency Analysis of Error Signals. The error represents the difference between the estimate and truth for one hardware testing run.

Error Signal	**Amplitude**	**Frequency**	**Time Period (1/f)**
x	0.0016 m	0 Hz	N/A
	0.02661 m	0.02747 Hz	36.4033 sec
	0.01751 m	0.2381 Hz	4.1999 sec
y	0.0355 m	0 Hz	N/A
	0.03168 m	0.009156 Hz	109.2180 sec
	0.0146 m	0.2228 Hz	4.4883 sec
z	0.0370 m	0 Hz	N/A
	0.04188 m	0.03052 Hz	32.7654 sec
ψ	0.6188 °	0 Hz	N/A
	1.8243 °	0.003052 Hz	327.6540 sec
	1.1442 °	0.009156 Hz	109.2180 sec
	0.8594 °	0.02442 Hz	40.9500 sec
	0.7746 °	0.03052 Hz	32.7654 sec
	0.7288 °	0.05494 Hz	18.2017 sec

To determine which components are not due to noise, many data runs were analyzed. A dominate frequency component that varied between 0.21-0.24 Hz was common in both the x and y-errors over many data runs, therefore deemed repeatable and significant. This component could be attributed to an inaccurate system model; however, there are many processes within the closed loop of the hardware test that

could contribute to its existence. These could include, but are not limited to: noise filters created in Labview, the MX Ultranet processor, Vicon cameras, and the 0.25 second delay just recently found in the system. The LQG controller was eliminated from this list because the same frequency was found when controlling El Toro using a PID controller during the model validation process detailed in Section 4.1.1. A recommendation was sent to the AFRL MAV IFF point of contact to perform the same analysis on another vehicle (e.g., Axe helicopter) to investigate the oscillation source.

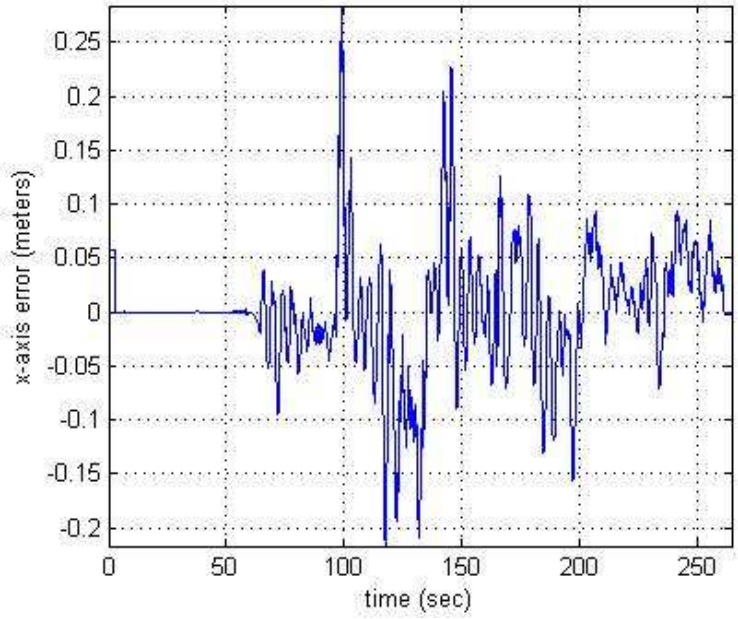


Figure 4.13: Errors in the x-direction. The errors were calculated by subtracting the desired position from actual position of the vehicle. The large jumps in error occur during a vehicle translation in the x-direction.

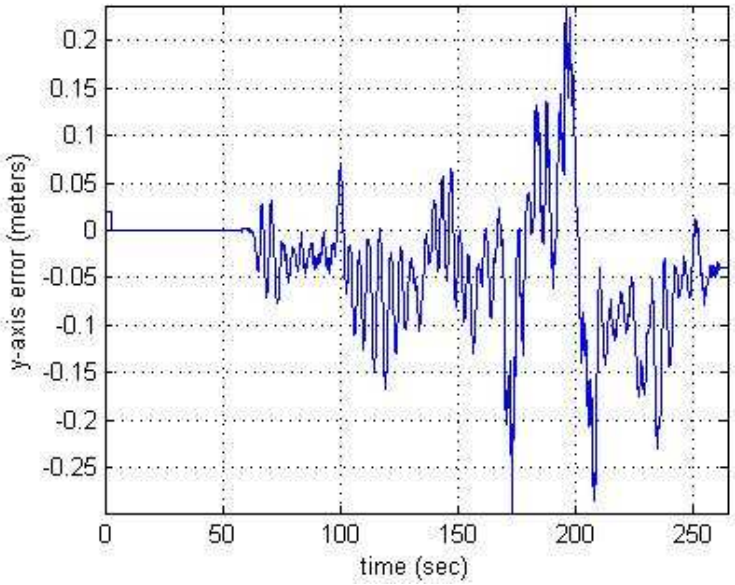


Figure 4.14: Errors in the y-direction. The errors were calculated by subtracting the desired position from actual position of the vehicle. The large jumps in error occur during a vehicle translation in the y-direction.

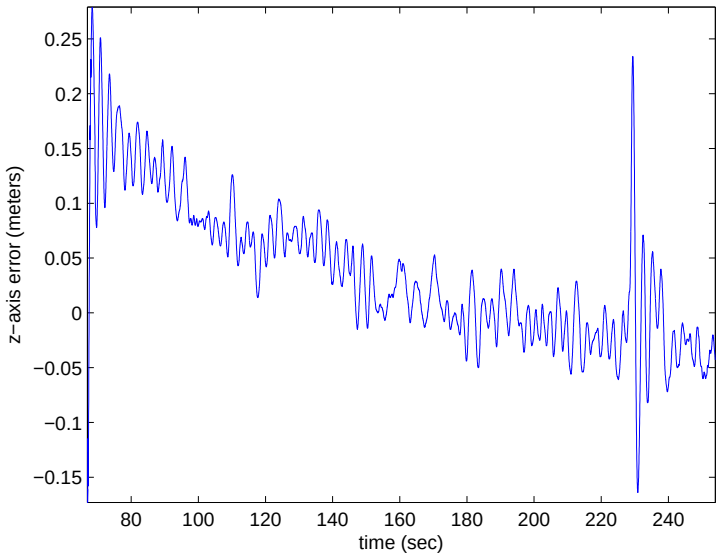


Figure 4.15: Errors in the z-direction. The errors were calculated by subtracting the desired position from actual position of the vehicle. The slight slope in error is due to battery drain.

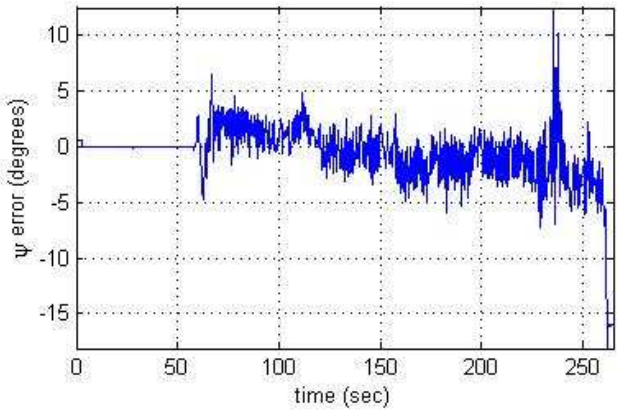


Figure 4.16: Errors in Heading. The errors were calculated by subtracting the desired heading from actual heading of the vehicle. The slope in error is attributed to the slight coupling between throttle and rudder and the on-going battery drain, while the large jump occurs at a heading change.

In addition, the 3D radial error was analyzed. In a separate trajectory where the helicopter maintained hover for 60 seconds, the radial error (err_r) was calculated using recorded data from Vicon:

$$err_r = \sqrt{x_{err}^2 + y_{err}^2 + z_{err}^2} \tag{4.3}$$

A histogram was then created as shown in Figure 4.17. The pdf shows a steady state error of approximately 0.1 meters, which could be easily rectified by adjusting the trim settings in Labview if required. Furthermore, by analyzing the data, the radial error stayed within 13 centimeters, 83% of the time.

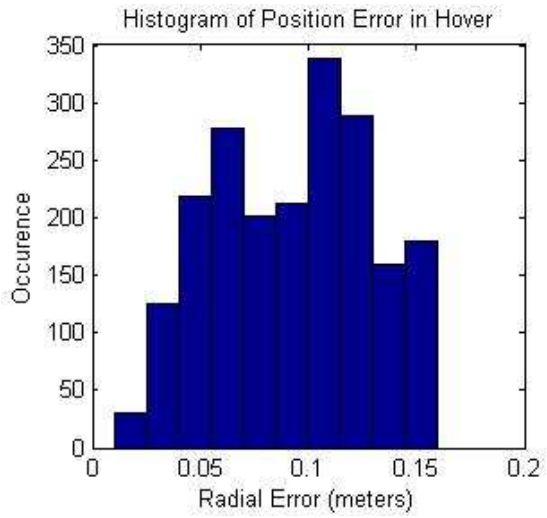


Figure 4.17: Range errors. The desired position in Vicon is a point inside a 3D coordinate system. This graph represents the histogram of the range errors during the hardware test.

Now that the controller and model have been verified, attention is refocused on state estimation.

4.3 Stochastic Estimation

The LQG controller not only consists of the LQR controller, but a Kalman Filter to provide optimal state estimation assuming additive white Gaussian process and measurement noise. Both the Extended and Unscented Kalman Filters were implemented in MATLAB as discussed in Section 3.3. Simulations were performed using both filters with a propagation and measurement update time step of 0.5 seconds.

4.3.1 EKF/UKF Comparison. The EKF equations used are detailed in Section 2.3.1. The UKF tuning parameter that determines the sigma spread, α, was varied to understand its effects on the error and to select the appropriate value. Typically, this value is set to 1e-4 $\leq \alpha \leq 1$ [27]. The simulation would not work with α values lower than 0.25 due to the state covariance violating positive definiteness when performing a Cholesky square root. When varying the value from 0.25 to 1, the position and heading errors change. The value of α that provided the lowest error

statistics on position and heading was 0.8; therefore, when comparing EKF to UKF, the following tuning parameters were used: $\alpha = 0.8$, $\beta = 2$, and $\kappa = 0$. The results of a 16-second, 40-run Monte Carlo simulation, providing a throttle, rudder, pitch, and roll for hover and varying random inputs for process measurement noise for each run are shown in Figures 4.18 - 4.21. Clearly the UKF outperforms the EKF; however, while analyzing the data one problem arose with the UKF. The standard deviation of the ensemble is also estimated from the UKF and EKF by taking the square root of the diagonal terms in the state covariance matrix. The standard deviation estimates were also compared to the ensemble standard deviations of the position and the attitude over the 40 runs; the solid black lines represent the filter's estimate of the standard deviation, while the solid red lines represent the ensemble's standard deviation, as shown in Figures 4.18 - 4.21. The EKF's estimate for these values matched within approximately 20% in position and with more data runs these values are expected to become more closely matched; however, the UKF's estimate for each standard deviation were over 1000% or more larger than the ensemble standard deviation for position and heading. Although at a higher vertical axis scale the filter's standard deviation estimate seems to level off, it never reaches a steady state value (independent of run length). In fact, upon further inspection, the standard deviations for all states except position tend to have a steady state value. This trait is not dependent upon the initial state covariance definition or the tuning parameters. Additionally, the EKF and UKF share three common functions: model propagation, system model linearization, and discrete process noise covariance generation. Although many steps have been taken to isolate the issue, further investigation is required.

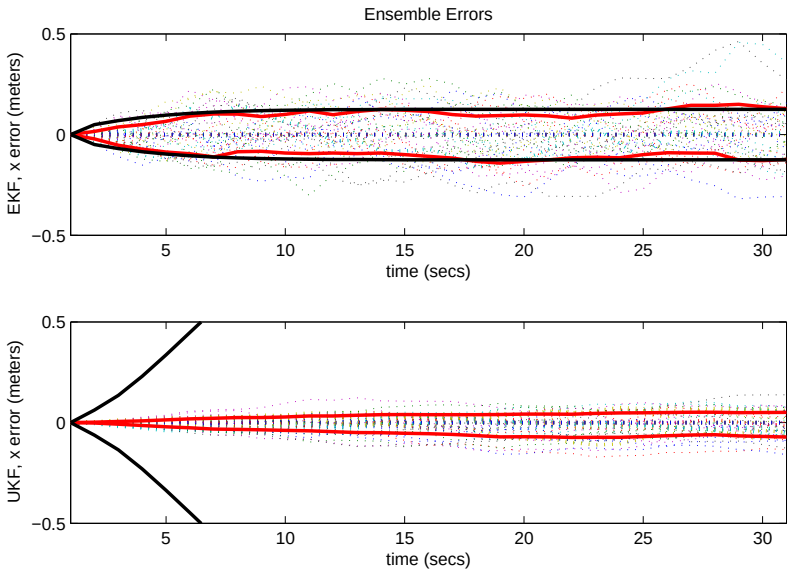


Figure 4.18: Kalman Filter Ensemble Errors for x. The error signifies the difference between the whole-value estimate and the truth. Lines representing ensemble (red) and filter estimate (black) standard deviation are shown. The error variation from the UKF is $\sim$50% less than the EKF.

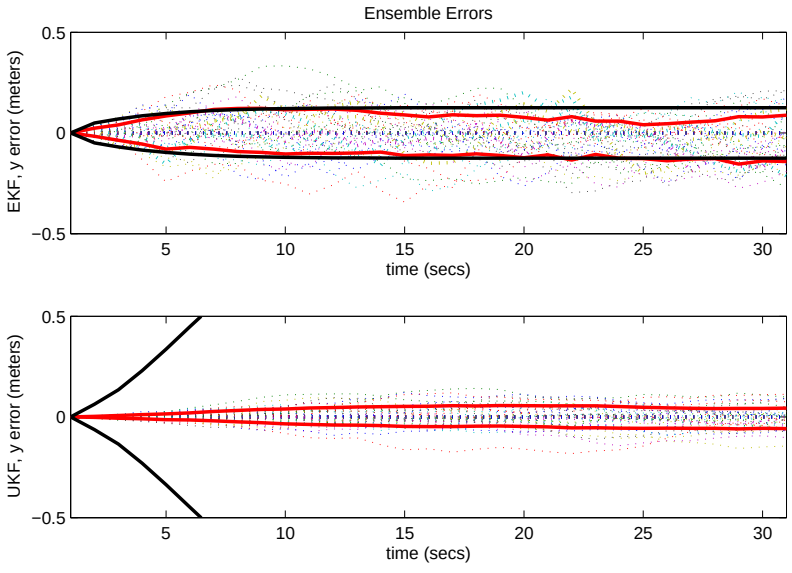


Figure 4.19: Kalman Filter Ensemble Errors for y. The error signifies the difference between the whole-value estimate and the truth. Lines representing ensemble (red) and filter estimate (black) standard deviation are shown. The error variation from the UKF is $\sim$50% less than the EKF.

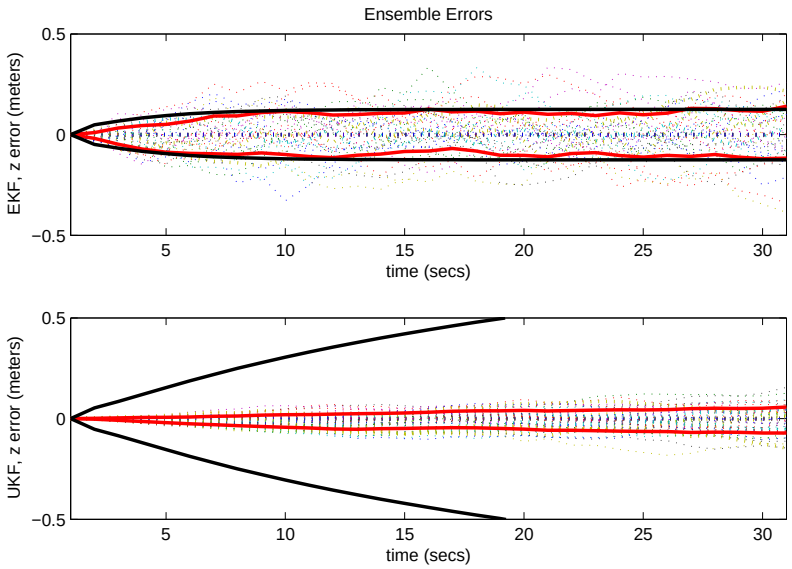


Figure 4.20: Kalman Filter Ensemble Errors for z. The error signifies the difference between the whole-value estimate and the truth. Lines representing ensemble (red) and filter estimate (black) standard deviation are shown. The error variation from the UKF is ∼50% less than the EKF.

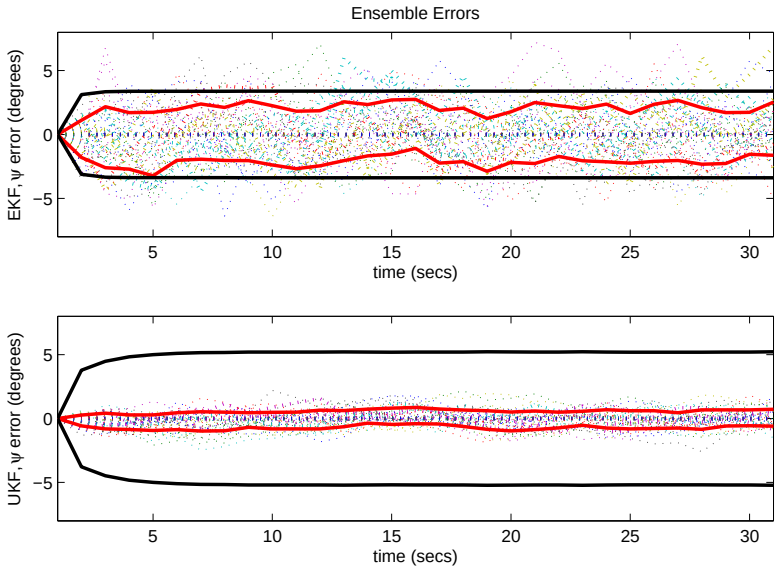
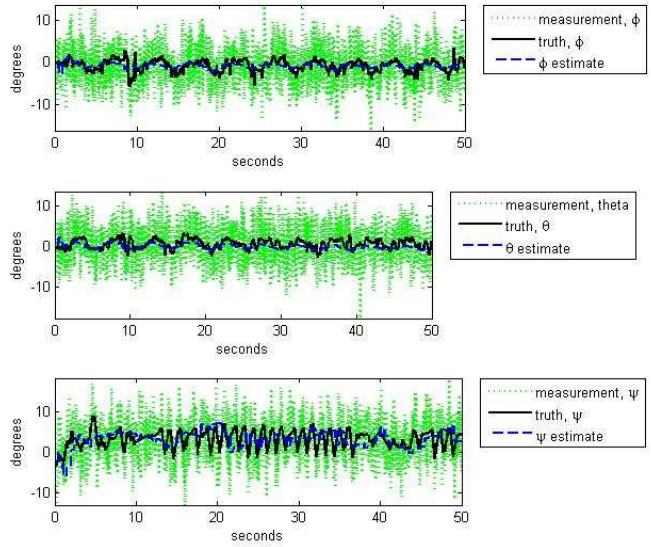


Figure 4.21: Kalman Filter Ensemble Errors for ψ. The error signifies the difference between the whole-value estimate and the truth. Lines representing ensemble (red) and filter estimate (black) standard deviation are shown. The error variation from the UKF is ∼50% less than the EKF.

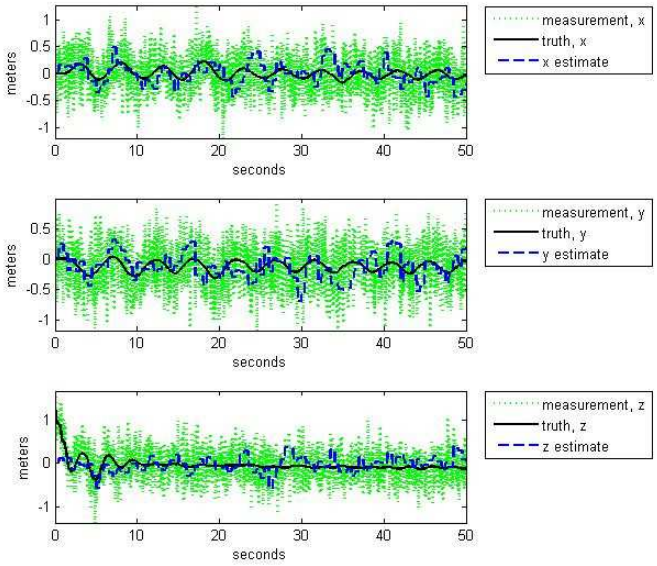
4.3.2 Validation using Vicon Data. Because the MAV IFF is currently unable to support the integration of a UKF in Labview, some additional post-processing efforts were taken to verify the UKF with actual data recorded during hover. The recorded position, velocity, attitude, and attitude rates are considered truth, while random Gaussian noise was added to create measurements. The truth, measurements, reference data depicting desired position and heading, and inputs are made available in the MATLAB workspace and read into Simulink during simulations to test the UKF s-function. The measurements and truth were subtracted from the reference to create errors depicting deviation from the reference, which shows a oscillating trajectory about the desired location. The estimates produced by the filter are based on the measurements and inputs. These estimates were subtracted from the truth to create the error signals. The statistics of these error signals are outlined in Table 4.2, and the plots resulting from the simulation are shown in Figure 4.22. The magnitude of the errors are larger than expected. This seems logical due to the excessive amount of noise recorded on the input signal versus the filtered data depicting the navigation solution and the additional noise placed on the truth to create measurements. Due to this logic, the UKF design is deemed sufficient for integration into the final controller setup. The final testing will commence after after confirming the INS mechanization.

Table 4.2: UKF Temporal Error Statistics using Actual Data. The error defined for these statistics is the difference between the truth and UKF estimate.

State	**Mean**	**Standard Deviation**
x	0.0509 m	0.1451 m
y	-0.0154 m	0.1505 m
z	0.1199 m	0.1188 m
ϕ	0.2994 °	1.1857 °
θ	-0.1621 °	1.0972 °
ψ	-3.2473 °	1.9259 °



(a) Attitude



(b) Position

Figure 4.22: Open Loop Simulation with Actual Data. The recorded inputs and measurements from the Vicon system were used as inputs and truth. Additional random white Gaussian noise was added to the truth and used as measurements to test the UKF in an open-loop environment. All signals characterize the deviation from the desired position/attitude.

4.4 Inertial Navigation

The inertial navigation design solution is the last major piece of the controller setup to be tested before testing the integrated, final configuration. The inertial navigation mechanization and error model used for propagation during state estimation is readily available through AFIT's ANT Center. The m-files were only slightly modified for use in this effort, as discussed in Section 3.4. The difficulty in this test laid in the creation of the Δv^b's and $\Delta\theta_{ib}^b$'s (raw INS data) which represent the change in velocity and attitude over each sample period, respectfully, from the accelerometers and gyros. There were three methods for deriving this information:

- Using data recorded by Vicon
- Using smoothed Vicon data
- Creating data using MATLAB

All three methods used a function (m-file) available in the ANT Center to extract the raw INS data from position and attitude. This function will be referred to as *reverse_ins_integrate*. The code used to call this function is located in the Appendix C. Once the raw INS data became available, they were fed into the INS mechanization, as shown in Figure 4.23. The result is then compared to the original trajectory.

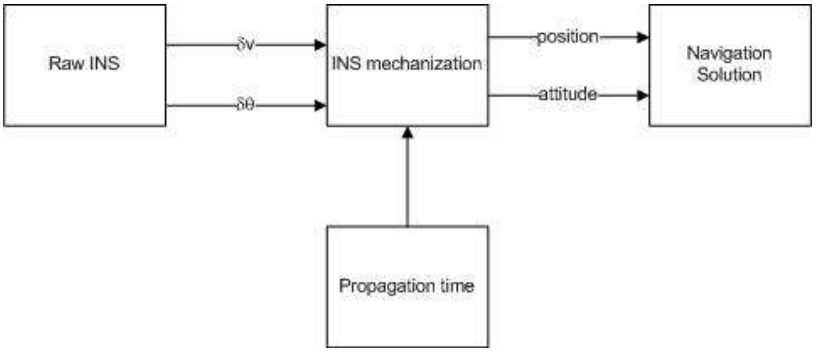


Figure 4.23: INS Mechanization Setup. The generated raw INS data, Δv^b and $\Delta\theta_{ib}^b$, are fed into the INS Mechanization to analyze the resulting trajectory.

First, the actual position data (x, y, and z) recorded by the Vicon system is used to test the mechanization. The results were found to be grossly erroneous. To help troubleshoot this problem, the Vicon data was smoothed using a Butterworth filter available in Simulink, with a filter order of 30 and passband edge frequency of 0.9 $\frac{rad}{sec}$. The smoothed Vicon data is then processed using *reverse_ins_integrate* to gain the raw INS data. Next, the raw INS data is fed into the INS mechanization using the same process used for the raw Vicon position data. The resulting trajectory when compared to the original trajectory, only showed minor differences. The mechanization was further tested with a trajectory created in MATLAB. The outcome of this analysis showed the mechanization did work, but only if the raw INS data did not have large, high-frequency variations (noise). To better control this effect, the trajectory and inputs to the system model/filter will be created in MATLAB in future simulations. Now that the INS mechanization has been verified, the next goal is to test the integration of all parts previously covered in this section.

4.5 System Model and INS Combination

Thus far, each block within the final design outlined in Figure 3.10 has been verified through simulations. In addition, the LQR controller has not only been verified through simulations, but also through hardware tests using the actual helicopter, El Toro. This section tests and analyzes the results of the final design. The point of which is to find a way to combine information contained in INS and system model mechanization, and models contained within the Kalman filter to more accurately predict the actual position of the helicopter to maintain a hover condition. This final analysis is divided into two sections: controller integration/test and Monte Carlo analysis.

4.5.1 Controller Integration/Test. The integration of all parts of the final design is accomplished in three stages: Model-Only implementation, INS-only implementation, and combination implementation. The model-only implementation

setup is constructed in Simulink with s-functions built for the UKF and mechanization blocks based on Figure 3.10, indicating the system model mechanization and the UKF system model propagation will be used. The INS-only implementation follows the same design; the INS mechanization and the UKF INS error model will be used instead. The setup concept for these first two parts is more generally depicted in Figure 3.3. The final part, combination implementation, combines the two mechanizations and estimates used in parallel model propagations to provide a more accurate navigation solution as described in Section 3.5. The first two stages were tuned to provide the best navigation solution based on the available model. The final stage was then analyzed and compared by performing a Monte Carlo analysis. The Simulink diagrams used in the simulations are shown in the Appendix D.

4.5.2 Monte Carlo Analysis. A Monte Carlo analysis was used to calculate the statistical properties of the controllable states being routed to the LQR controller [17]. This type of analysis is performed to determine the most probable range of outputs, given the same deterministic inputs and different independent, Gaussian, random inputs over many samples. The mean and standard deviation of the errors will be the primary focus of this analysis. Furthermore, root-sum-square (RSS) calculations are performed to provide the overall radial and heading error over 50 runs. The root-mean-square (RMS) is derived from the RSS, given the symbol Θ and calculated as shown in the following equations to determine the typical radial and heading error:

$$\Theta = \sqrt{\frac{\sum_{i=1}^{50} err(i)^2}{50 \text{ runs}}} \tag{4.4}$$

Fifty samples was determined to be a sufficient number to calculate these statistics. To begin, the Simulink model shown in Figure D.1 was used to perform the analysis. A loop was constructed in an m-file to simulate this model three times for each run; once in each of the three configurations. This process was repeated 50 times to

generate 50 sample runs in each configuration. At the beginning of each run, random noise vectors were generated by using the *randn* function. These vectors were used for measurement noise (v), noise on the input signal (intensity S), and noise added to the raw INS inputs (intensity I), and did not change until the next run. These noise vectors were added to the truth $\left[x, y, z, \phi, \theta, \psi\right]^T$, the inputs $\left[u_1, u_2, u_3, u_4\right]^T$, and the raw INS inputs $\left[\delta v^b(1), \delta v^b(2), \delta v^b(3), \delta\theta^b_{ib}(1), \delta\theta^b_{ib}(2), \delta\theta^b_{ib}(3)\right]^T$, respectively. The covariance matrices for these signals are defined as follows:

$$R = \begin{bmatrix} 0.01\ \text{m}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01\ \text{m}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01\ \text{m}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.005\ \text{rad}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.005\ \text{rad}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.005\ \text{rad}^2 \end{bmatrix} \tag{4.5}$$

$$S = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.02 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0.2 \end{bmatrix} \tag{4.6}$$

$$I = \begin{bmatrix} 5e\text{-}5\frac{m^2}{s^2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 5e\text{-}5\frac{m^2}{s^2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 8e\text{-}3\frac{m^2}{s^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 5e\text{-}6\ \text{rad}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5e\text{-}6\ \text{rad}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5e\text{-}5\ \text{rad}^2 \end{bmatrix} \tag{4.7}$$

Additionally, the system and INS error models within the filter were tuned previously by tweaking the values in the process noise matrix, Q. The resultant matrices for the INS error model and system error model are as follows:

$$\begin{aligned} Q_{INS} = \Lambda\Bigg(& 0.01\frac{m^2}{s},\ \ 0.01\frac{m^2}{s},\ \ 0.25\frac{m^2}{s},\ \ 0.0011\frac{m^2}{s^3}, \\ & 0.0011\frac{m^2}{s^3},\ \ 0.0011\frac{m^2}{s^3},\ \ 0.15e\text{-}5\frac{rad^2}{s},\ \ 0.15e\text{-}5\frac{rad^2}{s}, \\ & 0.82e\text{-}3\frac{rad^2}{s},\ \ 0.82e\text{-}4\frac{m^2}{s^5},\ \ 0.82e\text{-}4\frac{m^2}{s^5},\ \ 0.82e\text{-}4\frac{m^2}{s^5}, \\ & 0.11e\text{-}9\frac{rad^2}{s^3},\ \ 0.11e\text{-}9\frac{rad^2}{s^3},\ \ 0.11e\text{-}9\frac{rad^2}{s^3}\Bigg) \end{aligned} \tag{4.8}$$

$$Q_{sys} = \begin{bmatrix} 1\frac{N^2}{s} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1\frac{N^2}{s} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1\frac{N^2}{s} & 0 & 0 & 0 \\ 0 & 0 & 0 & 5e\text{-}15\frac{N^2-m^2}{s} & 0 & 0 \\ 0 & 0 & 0 & 0 & 5e\text{-}15\frac{N^2-m^2}{s} & 0 \\ 0 & 0 & 0 & 0 & 0 & 5e\text{-}3\frac{N^2-m^2}{s} \end{bmatrix} \tag{4.9}$$

Each run performed a pitch, roll, and yaw maneuver and had a duration 100 seconds. Samples for measurement and truth were collected for each run at a 50 Hz rate. The measurements (trajectory) are then subtracted from the truth to create the error signals. These error signals are saved in a 50×5001 matrix with each row representing the run and each column representing the samples collected over the duration of each run. The error resulting from each of the 50 runs (i.e. δx, δy, δz, $\delta\phi$, ...) is plotted with respect to time to gain a better perspective of the pdf (see Figures 4.24 - 4.29).

Furthermore, temporal statistics are calculated using the RMS results over 50 runs. The mean and standard deviation were calculated for each configuration for

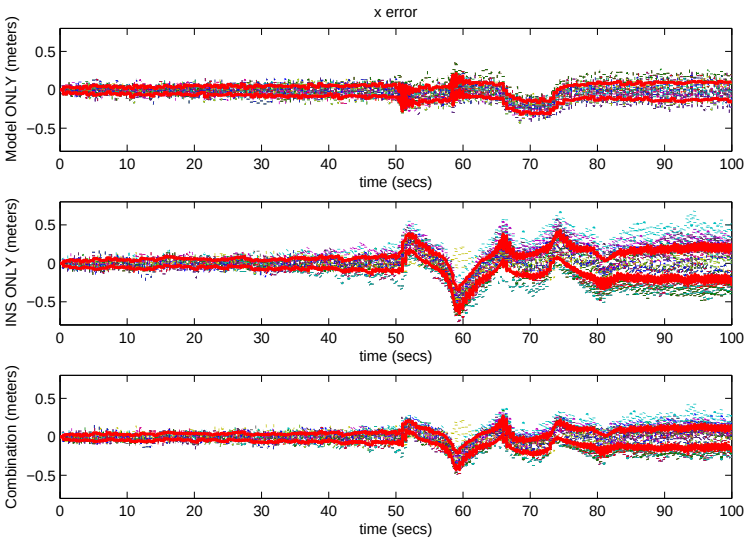


Figure 4.24: Ensemble Statistics for Position Error in the x-direction over 50 Runs. The x error was calculated over 50 runs in the Model ONLY, INS ONLY, and Combination (Both) configurations. The red lines indicate the standard deviation over the 50 runs at each point in time.

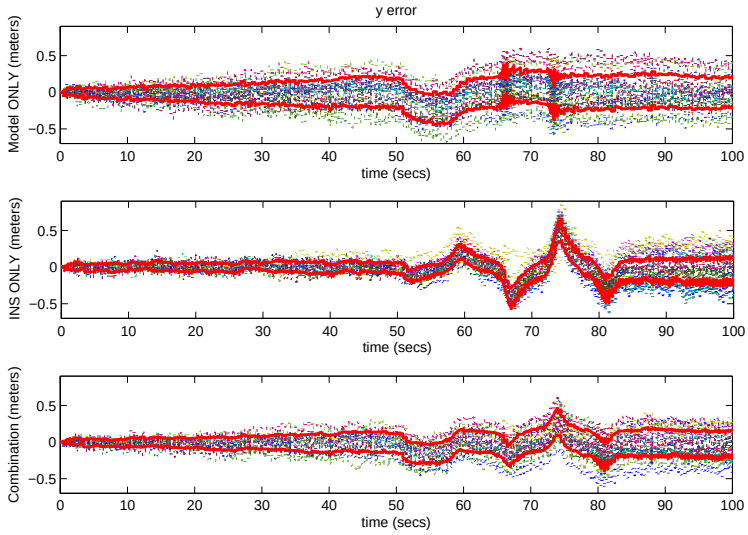


Figure 4.25: Ensemble Statistics for Position Error in the y-direction over 50 Runs. The y error was calculated over 50 runs in the Model ONLY, INS ONLY, and Combination (Both) configurations. The red lines indicate the standard deviation over the 50 runs at each point in time.

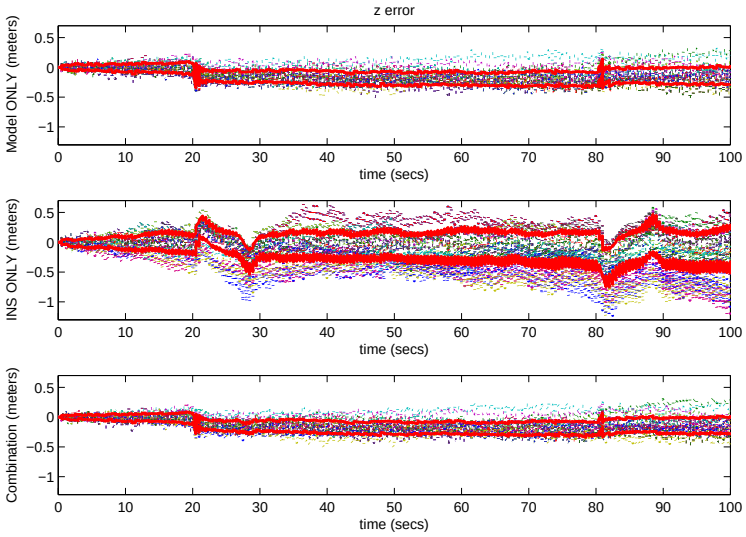


Figure 4.26: Ensemble Statistics for Position Error in the z-direction over 50 Runs. The z error was calculated over 50 runs in the Model ONLY, INS ONLY, and Combination (Both) configurations. The red lines indicate the standard deviation over the 50 runs at each point in time.

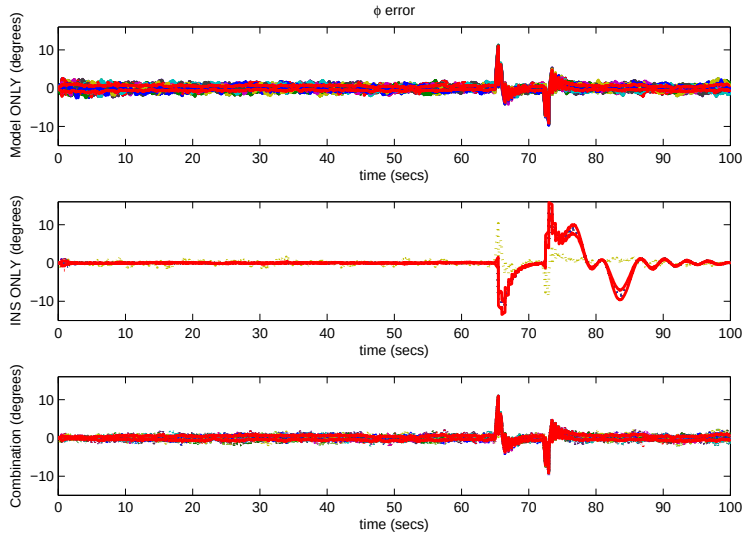


Figure 4.27: Ensemble Statistics for Position Error in the ϕ-direction over 50 Runs. The ϕ error was calculated over 50 runs in the Model ONLY, INS ONLY, and Combination (Both) configurations. The red lines indicate the standard deviation over the 50 runs at each point in time.

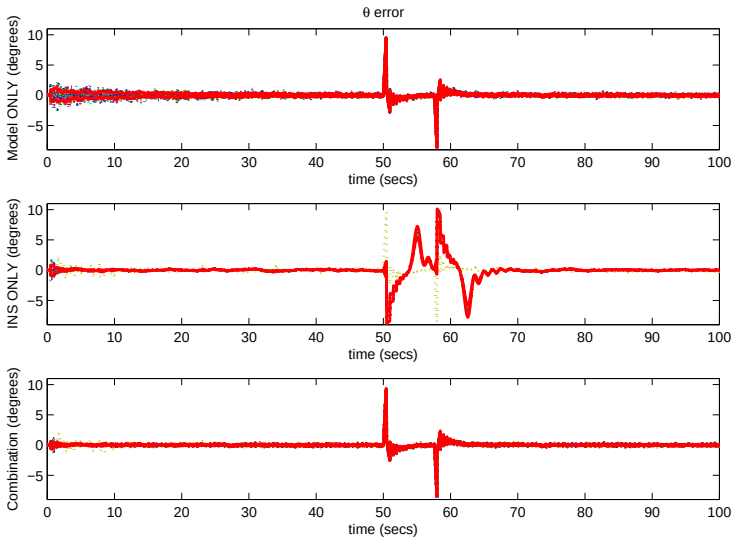


Figure 4.28: Ensemble Statistics for Position Error in the θ-direction over 50 Runs. The θ error was calculated over 50 runs in the Model ONLY, INS ONLY, and Combination (Both) configurations. The red lines indicate the standard deviation over the 50 runs at each point in time.

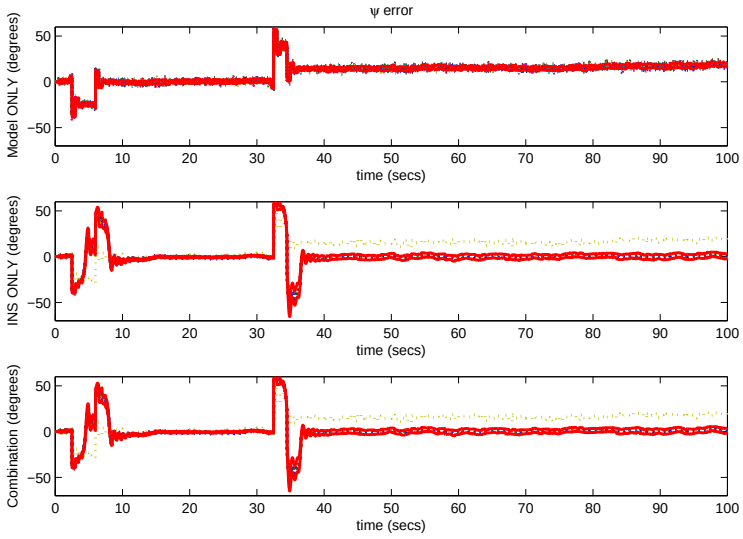


Figure 4.29: Ensemble Statistics for Position Error in the ψ-direction over 50 Runs. The ψ error was calculated over 50 runs in the Model ONLY, INS ONLY, and Combination (Both) configurations. The red lines indicate the standard deviation over the 50 runs at each point in time.

position and attitude. The results when the combination configuration is compared to the other two configurations are summarized in Table 4.3. Notably, the combination configuration demonstrated superior performance with respect to all variables except for the y error. In hindsight, the models could have been tuned further to place a larger noise intensity on the corresponding system process noise for y and a smaller noise intensity on the INS process noise for y. The reason this was not changed is to provide an example of how tuning the process noise matrix to compensate for model defects is crucial in the success of the combination implementation.

Table 4.3: Combination vs. Model Only and INS Only Configuration Results. The combination configuration was compared to the model-only and INS-only by calculating the mean of each RMS run and comparing the results for each state between configurations. A plus (+) indicates an improvement over the over method, while a minus (-) indicates a degradation.

State	**Improvement over Model ONLY**	**Improvement over INS ONLY**
x	+5.72%	+31.44%
y	+26.56%	-15.34%
z	+4.50%	+25.59%
radial	+14.00%	+19.57%
ϕ	+8.00%	+37.50%
θ	+36.36%	+40.23%
ψ	+433.70%	+0.21%

To keep the helicopter in a hover condition, the most important errors in this effort are the radial position and heading (ψ) errors. The radial RMS position errors were calculated for each of three configurations, using Equation 4.3. These errors are plotted over time for each configuration, as shown in Figures 4.30 and 4.31. Notice the INS does a better job of predicting ψ. This fact was realized early (before the Monte Carlo runs), thus the system model's process noise intensity value in the UKF for ψ was increased dramatically so the combination configuration would rely heavily on the INS result. The heading error results shown in the plot convey the combination tracking the INS steadily throughout each 100-second run. In contrast, the radial position results for the combination implementation outperform the model-only and INS-only configurations by 14% and 20%, respectfully. Looking at Figure 4.30, the INS-only

results have a large variation; after further investigation, this is due to the noise added to the raw INS inputs combined with the sensitivity of the INS mechanization. Due to the previous issues with the UKF's state covariance, this exercise was repeated with the EKF as the state estimator. Once again, new truth was generated and the noise levels applied to the raw INS data, inputs and measurements were modified to reflect realistic levels, and the process noise covariance matrix for each model was tuned to reflect the model's uncertainty. A Monte Carlo analysis was performed over 30 runs, which resulted in the same conclusions with varying improvements equal to or greater than the UKF analysis. The results of the methodology are concluded. The combination configuration method does show a moderate improvement over the original methods. The conclusions gained from this effort are detailed in Section V.

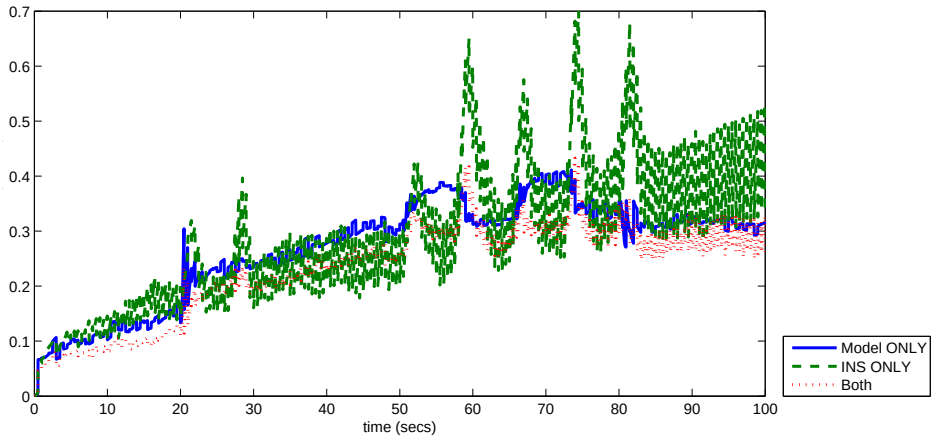


Figure 4.30: Ensemble Statistics for Radial Position RMS Error over 50 Runs. The radial position RMS error was calculated over 50 runs in the model-only, INS-only, and combination (both) configurations. The combination configuration performed better than the model-only and INS-only configurations by 14% and 20%, respectively.

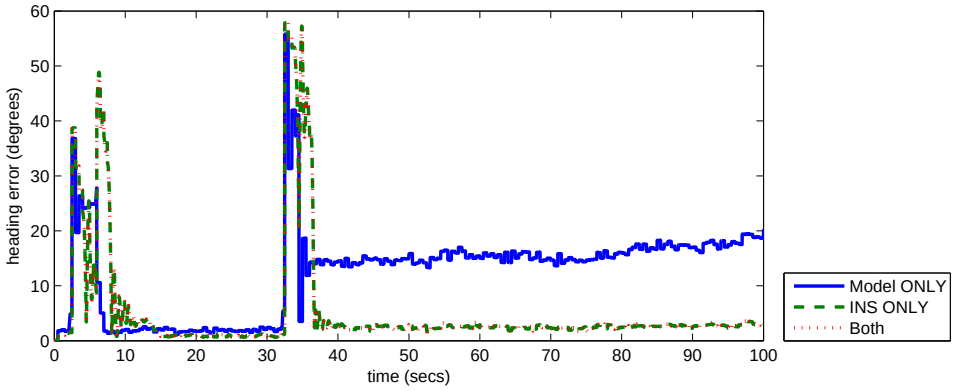


Figure 4.31: Ensemble Statistics for Radial Position RMS Error over 50 Runs. The radial position RMS error was calculated over 50 runs in the model-only, INS-only, and combination (both) configurations.

V. Conclusions

Unmanned aerial vehicles have become a vital part of today's military arsenal. The successful demonstration of the RQ-1 Predator's unique capabilities has led to the exploration of UAVs further employment into increasingly challenging combat environments. With today's war on terror, the battle is most commonly fought in urban areas; this realization became the impetus for research into micro-air vehicles. Due to degradation or denial of GPS inside buildings or underground, alternative methods for navigating are being pursued to provide the most accurate solution. One method relies on vision to extrapolate position and attitude. The purpose of this thesis is to build a linear quadratic Gaussian controller for a micro-sized helicopter with inputs provided from a system model combination INS mechanization using vision updates from a Kalman filter, then to test its effectiveness against more traditional setups.

5.1 Research Summary

The LQG technique is a type of model-based control. The nonlinear system model was built partially from equations derived from experimental data and partially from standard motor and 6DOF formulas. The model was reduced, linearized, and verified in an open-loop configuration using software-created inputs, then tested with actual data. For the hover condition, the weighting on position and heading was emphasized in the creation of the LQR gain matrix. With the system's states as inputs, the gain matrix provided throttle, rudder, pitch, and roll commands to the vehicle that kept the vehicle in a hover. The gain matrix was then validated through software and hardware tests, thus further verifying the legitimacy of the model. When testing with the actual helicopter, the radial position error was within 13 cm for 83% of a 60 second run, while the heading error remained under 5 degrees.

Next, an EKF and UKF were constructed and compared to determine the best candidate for state estimation of the vehicle's nonlinear system model. Consistent with previous studies, the UKF outperformed the EKF at a steady state hover by

approximately 50%. The final effort before integration involved constructing the INS mechanization and error model. The algorithms were previously developed and provided by AFIT's Advanced Navigation Technology Center. Each component within the overall architecture, shown in Figure 3.10, was integrated and simulated in three configurations: model-only mechanization, INS-only mechanization, and combination mechanization. The first two configurations use a different mechanization and model propagation in the Kalman filter. The final configuration combines the system and INS mechanizations and models to produce a more accurate navigation solution. This fusion technique is derived from combining the mean and covariance using the following equations:

$$P_c^{-1} = P_{ins}^{-1} + P_{sys}^{-1} \tag{5.1a}$$

$$\hat{x}_c = P_c \left(P_{ins}^{-1} \hat{x}_{ins} + P_{sys}^{-1} \hat{x}_{sys} \right) \tag{5.1b}$$

The process noise covariance was tuned for each model; if this tuning is not accomplished, it is possible that the combination configuration will not provide the most accurate solution. A Monte Carlo simulation was performed on each simulation and statistics calculated. With the covariances appropriately tuned, the combination configuration provided a radial position improvement over the model-only and INS-only configurations by 14% and 20%, respectively.

5.2 Issues

During this research several issues arose that leave cause for further investigation.

5.2.1 AFRL's MAV Indoor Flight Facility. This research served as the first use of the MAV IFF from an outside agency. Labview was used as the interface software for autonomous control. Much progress was made while integrating the LQR

gain matrix for closed-loop helicopter control; however, much work still needs to be accomplished. The hardware testing stopped when trying to install the Kalman filter inline with the controller. The following methods were attempted without success: building a C code wrapper for an m-file and building the Kalman filter in Simulink to be installed in Labview using advertised techniques. Translating the MATLAB code to C code was another option that was not pursued due to risk in schedule. The Kalman filter, as an alternative, was tested using actual data in software simulations. AFRL/RB is currently pursuing techniques to integrate state estimators in future testing for customers.

5.2.2 Trim Settings. Although the state estimation was not verified at the MAV IFF, the LQR gain matrix was verified through closed loop operation. This hardware test with El Toro was repeated over several weeks. The LQR provided stable control of the vehicle; however, a steady state error was revealed during each flight test. After some investigation, this steady state error corresponded to the helicopter's trim settings. Adjustment of the trim setting values became a part of the setup procedures before each flight test, like it would for manual operation. Further investigation into appropriate techniques is required to eliminate this procedure.

5.2.3 State Estimation. Unlike discovering the steady state error during flight tests was associated with trim settings, the state covariance figures produced by the UKF was unexplainable. In the state estimation section, the EKF and UKF were compared using a Monte Carlo Analysis. The state standard deviation values produced by the EKF matched the ensemble standard deviations, as expected; however, the UKF did not. The values produced by the UKF were up to 400% larger than the ensemble standard deviation. Many avenues were investigated to isolate the cause to no avail. The UKF was used due to its superior performance in estimating the states during hover; however, the EKF was used to help verify the UKF's suitability due to this unresolved issue.

These issues, in addition to improvements and hardware implementation outlined in the next section, should be addressed.

5.3 Future Work

The research performed in this thesis is only a stepping stone toward producing the most accurate navigation solution for a MAV in an urban setting when GPS is denied. The next step is to integrate the design onboard the airframe and perform a hardware test. There are several considerations in performing this feat. First, the payload capacity of the vehicle must be considered. The helicopter must be able to easily carry a processor, INS, cameras, and an additional battery, at a minimum. Furthermore, the controller could be improved from the LQG technique to provide an adaptive or learning capability to automatically adjust to fluctuating trim settings. Finally, the visual navigation algorithm should be integrated to provide measurement updates to the Kalman Filter.

5.3.1 Other Vehicles. El Toro was used in this effort because it was readily available through the ANT Center and had already been tested to derive the motor parameters and lift/torque equations. Although El Toro was suitable for the work performed in this thesis, it was limited in the amount of payload it could carry (approximately 0.22 pounds). Other considerations for vehicle selection are size (ability to maneuver through a doorway), battery requirement (duration of flight), and noise (only if the need to be stealthy arises). A larger tail-rotor helicopter or quadrotor type would be more appropriate to carry a larger payload (i.e., processor, extra battery, cameras, etc.). A new quadrotor, built in the ANT Center, has been selected as the interim solution over El Toro. It has a payload capacity of two pounds, which could ensure stable flight with all necessary equipment without having to max out at full throttle to maintain a hover. The process for modeling the ANT Center's quadrotor was completed; however, a controller was not built due to the limitations in schedule.

The nonlinear dynamics model derivation for the quadrotor is located in Appendix E for future reference.

5.3.2 Other Types of Controllers. Although LQG control techniques were employed during this research effort, other types of controls would be more suitable for integration. To reduce the position and heading steady state error, additional or alternative control techniques could be utilized. For instance, an integrating LQG technique could be employed to reduce steady state error; another name for this type of control is a *proportional-plus-integral* (PI) control. This technique "will provide a nonzero steady state control when its own input is zero" [16]. Also, an adaptive learning parameter estimation technique could be utilized to eliminate the need to readjust trim settings in pre-flight and adjust for dynamics differences between helicopters. Neural dynamic programming and direct adaptive control techniques are both realistic implementations for performing these tasks [4] [11] [8].

In conclusion, the simulation and hardware implementation of the LQR controller provided stable control of the Walkera 53-1, commercial micro-helicopter. Furthermore, simulations of the state estimation and model/INS combination mechanization and propagation approach provided moderate improvement over using the individual configurations discussed in this thesis. Hopefully, the work here will pave the avenue to flight control hardware implementation and testing for El Toro and/or the ANT Center quadrotor in the near future.

Appendix A. El Toro System Model

A.1 El Toro System Model Linearized about Hover

$$A=\begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -2.5 & 0 & 0 & 0 & -9.8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -5 & 0 & 9.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -398 & 0 & 0 & -398 & 0 & 0 & -398 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -323 & 0 & 0 & -323 & 0 & 0 & -323 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -166 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -5
\end{bmatrix}$$

$$B=\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & -2.073 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -112 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 323 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 398 & 0 & 0 & 0 & 0
\end{bmatrix}^T$$

$$C=\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}$$

$$D=\begin{bmatrix} 0 \end{bmatrix}$$

$$G=\begin{bmatrix}
3.125 & 0 & 0 & 0 & 0 & 0 \\
0 & 3.125 & 0 & 0 & 0 & 0 \\
0 & 0 & 3.125 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 5769 & 0 & 0 \\
0 & 0 & 0 & 0 & 4687 & 0 \\
0 & 0 & 0 & 0 & 0 & 3319 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

A.2 El Toro Linearized System Model Transfer Functions

Note: If an input-output transfer function is not listed, assume it's function is zero.

$$\frac{z}{throttle} = \frac{-2.073}{s(s+1.25)}$$

$$\frac{\dot{z}}{throttle} = \frac{-2.073}{s+1.25}$$

$$\frac{\psi}{rudder} = \frac{-112}{s(s+165.9)}$$

$$\frac{\dot{\psi}}{rudder} = \frac{-112}{s + 165.9}$$

$$\frac{x}{pitch} = \frac{-3169(s + 5.0016)}{s(s + 321.3708)(s + 6.2212)(s + 2.4990)(s + 0.809)}$$

$$\frac{\dot{x}}{pitch} = \frac{-3169(s + 5.0016)}{(s + 321.3708)(s + 6.2212)(s + 2.4990)(s + 0.809)}$$

$$\frac{\theta}{pitch} = \frac{323.4(s + 5)}{(s + 321.3708)(s + 6.2203)(s + 0.8089)}$$

$$\frac{\dot{\theta}}{pitch} = \frac{323.4s(s + 5)}{(s + 321.3708)(s + 6.2203)(s + 0.8089)}$$

$$\frac{y}{roll} = \frac{3901(s + 4.9987)}{s(s + 395.9769)(s + 6.2148)(s + 4.9995)(s + 0.8088)}$$

$$\frac{\dot{y}}{roll} = \frac{3901(s + 4.9987)}{(s + 395.9769)(s + 6.2148)(s + 4.9995)(s + 0.8088)}$$

$$\frac{\phi}{roll} = \frac{398(s + 5)}{(s + 395.9769)(s + 6.2144)(s + 0.8087)}$$

$$\frac{\dot{\phi}}{roll} = \frac{398s(s + 5)}{(s + 395.9769)(s + 6.2144)(s + 0.8087)}$$

$$G_c = \begin{bmatrix} 0.1100 & 0.0480 & -2.1613 & 0.0044 \\ -.1098 & -0.0479 & -0.0044 & 2.1623 \\ -3.0839 & 0.0050 & -0.1187 & -0.1197 \\ 0.0668 & 0.0311 & -1.4411 & 0.0031 \\ -0.0667 & -0.0310 & -0.0030 & 1.4410 \\ -1.4591 & 0.0021 & -0.0595 & -0.0600 \\ -0.1451 & -0.0769 & -0.0081 & 4.5980 \\ -0.1455 & -0.0771 & 4.6032 & -0.0082 \\ -0.0003 & -1.6530 & -0.0872 & -0.0878 \\ -0.0003 & -0.0002 & 0.0000 & 0.0092 \\ -0.0004 & -0.0002 & 0.0119 & 0.0000 \\ 0.0000 & -0.0254 & -0.0008 & -0.0008 \end{bmatrix}^T$$

Listing C.1:

```
%Create Δ_v and Δ_theta from Pwgs(t)
%
%Created by:  Capt Constance Hendrix

initialize


%% Initialize Cnb0 and v_ned0

C2b = [1 0 0;0 cos(Phi(1)) -sin(Phi(1));0 sin(Phi(1)) cos(Phi...
    (1))];
C12 = [cos(Theta(1)) 0 sin(Theta(1)); 0 1 0;-sin(Theta(1)) 0 ...
    cos(Theta(1))];
Cn1 = [cos(Psi(1)) -sin(Psi(1)) 0;sin(Psi(1)) cos(Psi(1)) 0;0 0...
     1];
Cnb = Cn1*C12*C2b;
Cnb0 = Cnb;

v_ned0 = [x_dot(1) y_dot(1) z_dot(1)];


%% sampling rate
dt=0.02;



%% obtain Δ_v and Δ_theta
for ii=1:(length(x)-1)
    C2b = [1 0 0;0 cos(Phi(ii+1)) -sin(Phi(ii+1));0 sin(Phi(ii...
        +1)) cos(Phi(ii+1))];
```

```
        C12 = [cos(Theta(ii+1)) 0 sin(Theta(ii+1)); 0 1 0;-sin(...
            Theta(ii+1)) 0 cos(Theta(ii+1))];
        Cn1 = [cos(Psi(ii+1)) -sin(Psi(ii+1)) 0;sin(Psi(ii+1)) cos(...
            Psi(ii+1)) 0;0 0 1];
        Cnb = Cn1*C12*C2b;
31      Cnb1 = Cnb;
        [△_v(ii,:), △_theta(ii,:), v_ned1(ii,:)] = ...
            reverse_ins_integrate(dt,Pwgs(ii+1,1),Pwgs(ii+1,2),Pwgs(...
            ii+1,3),Cnb1,Pwgs(ii,1),Pwgs(ii,2),Pwgs(ii,3),Cnb0,...
            v_ned0);
        Cnb0=Cnb1;
        v_ned0=v_ned1(ii,:);
    end
36

    %% Check

    C2b = [1 0 0;0 cos(Phi(1)) -sin(Phi(1));0 sin(Phi(1)) cos(Phi...
        (1))];
41  C12 = [cos(Theta(1)) 0 sin(Theta(1)); 0 1 0;-sin(Theta(1)) 0 ...
        cos(Theta(1))];
    Cn1 = [cos(Psi(1)) -sin(Psi(1)) 0;sin(Psi(1)) cos(Psi(1)) 0;0 0...
         1];
    Cnb = Cn1*C12*C2b;
    Cnb0 = Cnb;

46  lat0 = Pwgs(1,1);
    lon0 = Pwgs(1,2);
    alt0 = Pwgs(1,3);

    v_ned0 = [x_dot(1) y_dot(1) z_dot(1)];
51
    for ii=1:(length(x)-1)
```

```
   [lat1(ii),lon1(ii),alt1(ii),v_ned1(ii,:),Cnb1,f_NED]=...
      ins_integrate(△_v(ii,:),△_theta(ii,:),dt,lat0,lon0,alt0,...
      v_ned0,Cnb0,0);
   Cnb0 = Cnb1;
   v_ned0 = v_ned1(ii,:);
56 lat0 = lat1(ii);
   lon0 = lon1(ii);
   alt0 = alt1(ii);
   end
```

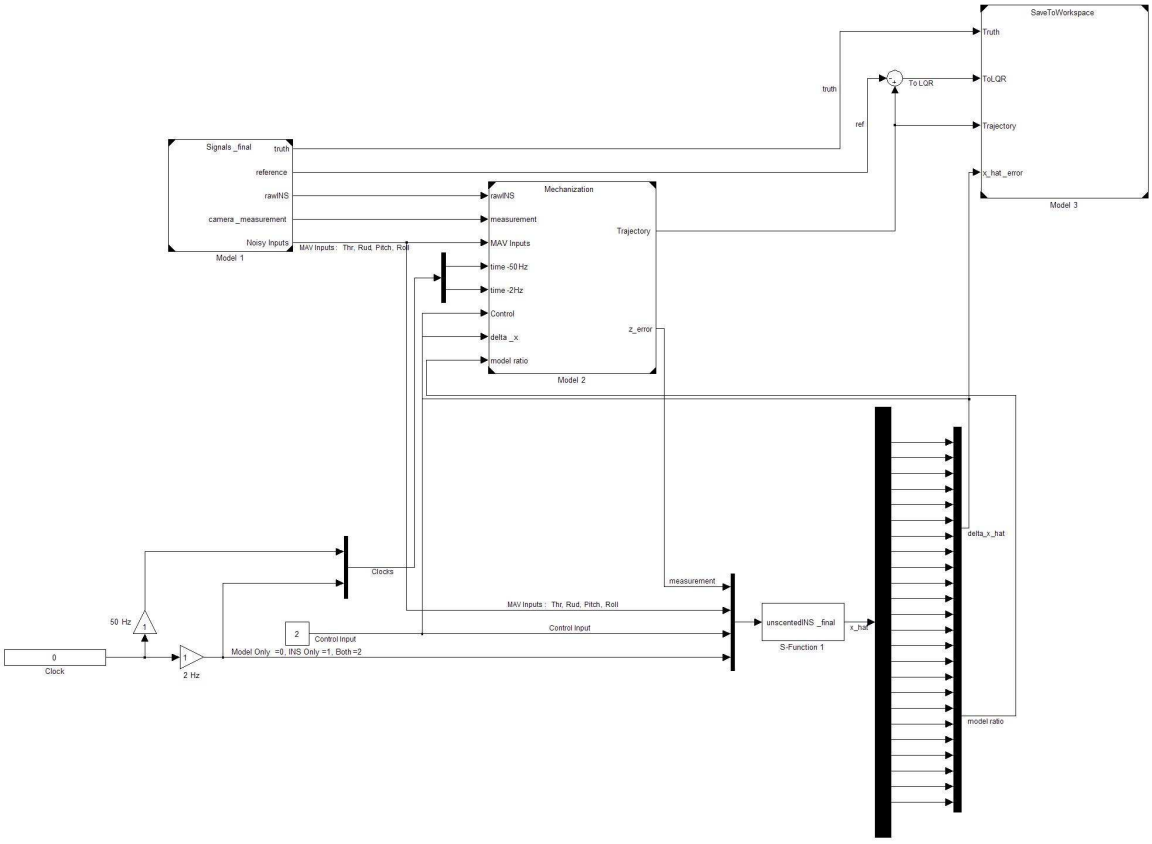


Figure D.1: Overall Process with System Model and INS Combination Selectable. This Simulink model is used for simulations to test the state estimation configuration.

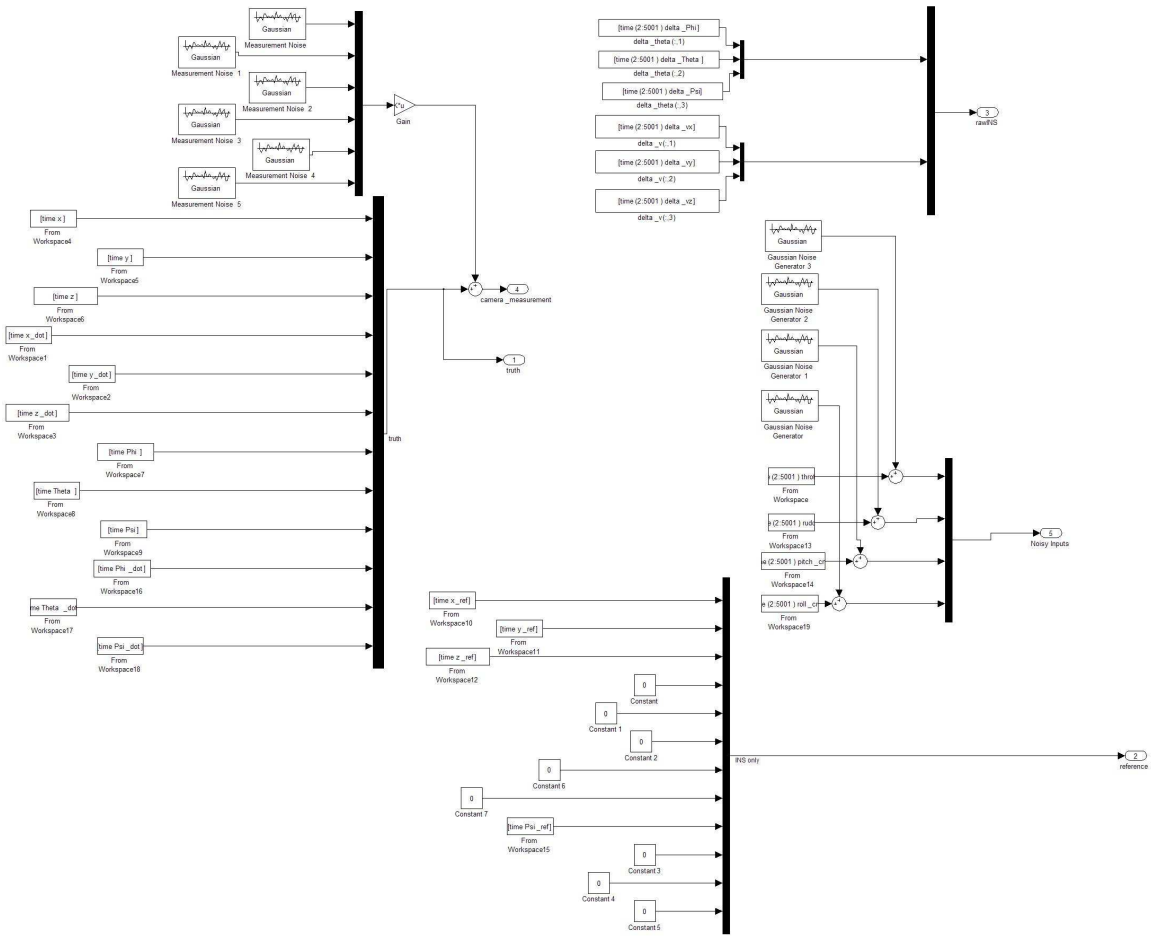


Figure D.2: Deterministic and Stochastic inputs to the Simulink Model. These inputs are fed to the Mechanization block, as well as the UKF, for simulation.

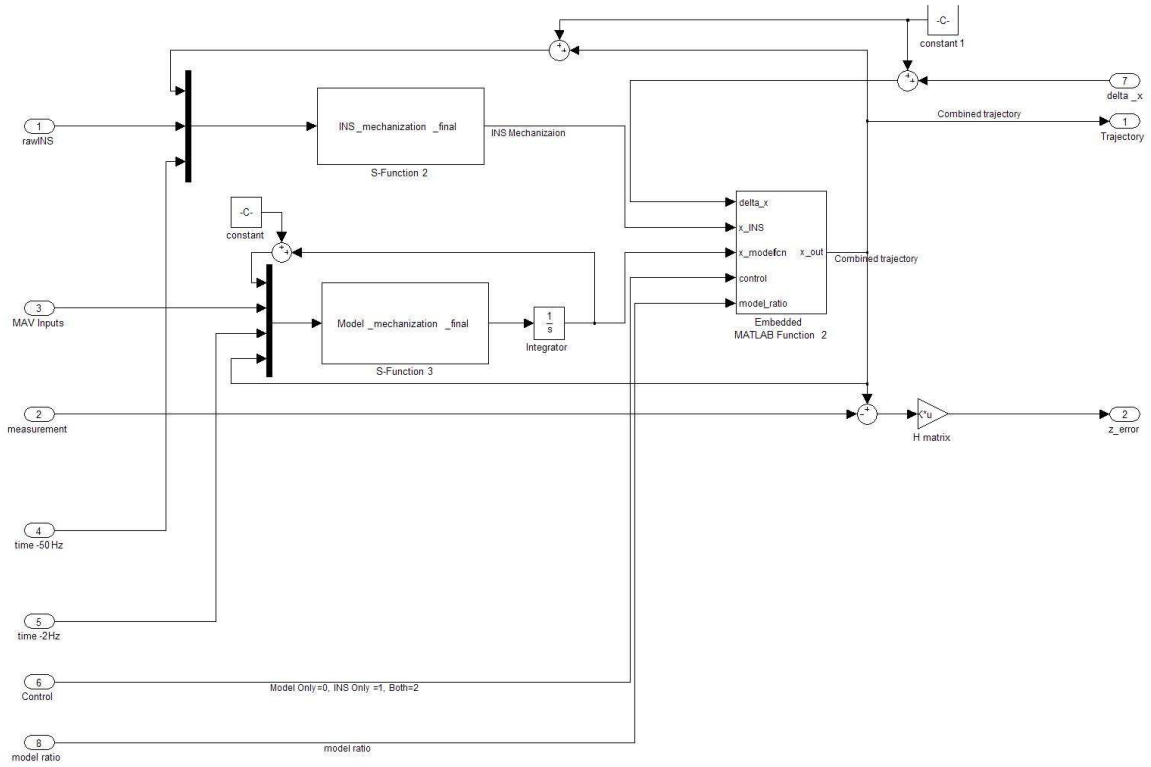
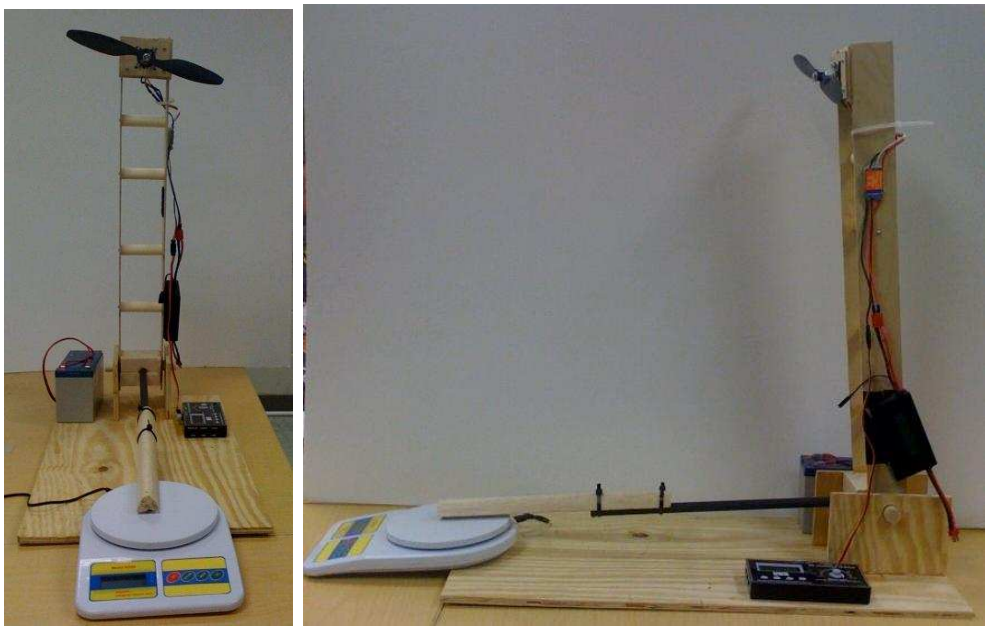


Figure D.3: Mechanization Block. The system model, INS, or combination mechanization is updated by the error state estimate, producing a nominal trajectory. The nominal state vector is then subtracted by the camera measurement to produce the measurement error, which is then used to update the error state vector in the UKF.

Appendix E. Quadrotor Modeling

Modeling of the ANT Center's quadrotor was accomplished as a first step to future work. The modeling began by generating lift and torque curves using experimental methods. One of the four motor/blade assemblies was dismantled from the vehicle and configured in the setup shown in Figure E.1. The motor was controlled using a servo tester that allowed the pulse width of the pulse-width modulation (PWM) signal to be varied. Varying of this pulse width from 1000 μsec to 2000 μsec causes the motor to go from zero to full throttle. In 50 μsec steps, the force due to lift was recorded using a digital scale, the angular velocity was measured using a tachometer, and the battery power and current was measured using an inline wattmeter. Additionally, no-load tests were performed to estimate the rotational power loss [29]. All other losses, to save time, were neglected.



(a) Front View (b) Side View

Figure E.1: Quadrotor Modeling Test Setup. Lift and torque equations were derived through testing of motor/blade assembly for the quadrotor using the following setup. The moment arms from the pivot point to the blade and scale were measured to be exact.

The resulting equations for lift (F_{lift}) and torque (τ) for one motor blade assembly in terms of the PWM pulse width input (p) are shown below with the curves

plotted in Figure E.2.

$$\tau = -2.7407e-011(p)+1.8141e-007(p)-3.9919e-004(p)+0.3694(p)-124.4460(p)$$

$$F_{lift} = 6.7816e-006(p) - 0.0099(p) + 2.9035(p)$$

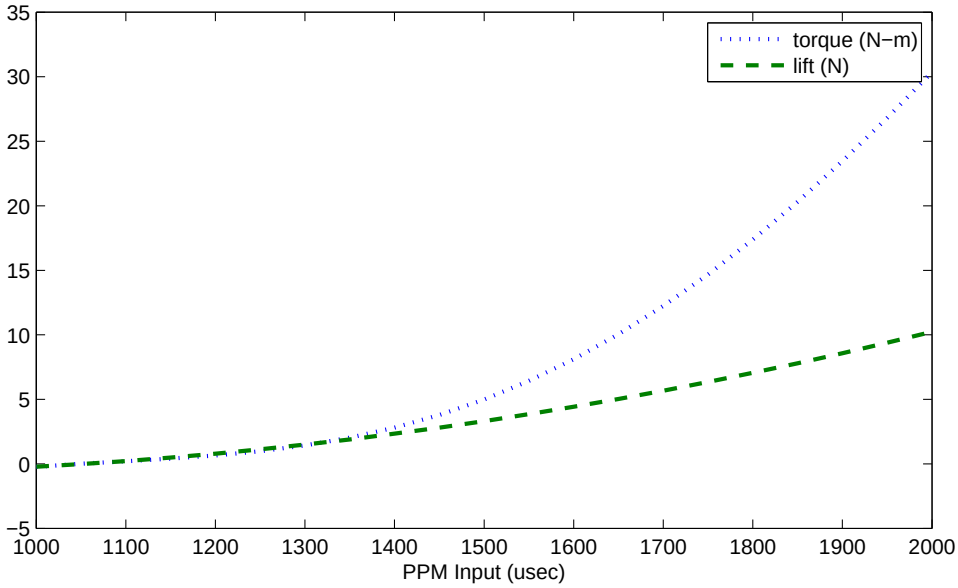


Figure E.2: Quadrotor Lift and Torque Curves. These plots were derived from experimental data in terms of the PWM input, then approximated using a polynomial trendline in Excel and verified in MATLAB using the *polyfit* function.

These equations are used for each of the four motor blade assemblies, with the subscript number for each equation corresponds to the motor location layout in Figure E.3. The force (F) and moment (M) equations used as input to the 6DOF model,

$$F = C_b^n \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -F_{lift1} - F_{lift2} - F_{lift3} - F_{lift4} \end{bmatrix}$$

$$M = \begin{bmatrix} r_b & 0 & 0 \\ 0 & r_b & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} F_{lift1} - F_{lift2} + F_{lift3} - F_{lift4} \\ F_{lift1} + F_{lift2} - F_{lift3} - F_{lift4} \\ \tau_1 - \tau_2 - \tau_3 + \tau_4 \end{bmatrix}$$

where r_b is the radius from the center of mass of the vehicle to the center of the blade. These equations were provided as inputs to the 6DOF model.

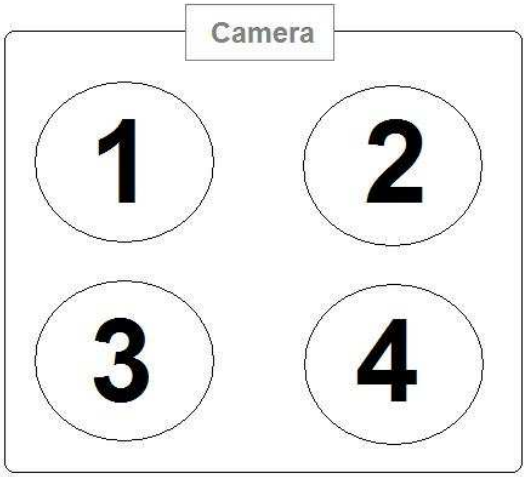


Figure E.3: Quadrotor Motor Layout. The camera location indicates the front of the vehicle, while the numbers indicate motor blade assembly location.

Next, the moment of inertial tensor in the body frame is garnered. Like El Toro, the quadrotor uses Equation (3.6) with Ixx, Iyy, and Izz provided by 2Lt. Don J. Yates from the ANT Center through previous experimentation. All parameters defined are summarized in Table E.1.

Table E.1: Quadrotor Parameters. The following parameters were previously defined using experimentation and various measurement devices.

Parameter	**Value**
m	0.9 kg
r_b	0.3 m
Ixx	0.0547 kg-m^2
Iyy	0.0547 kg-m^2
Izz	0.0547 kg-m^2

The final ingredient to produce the nonlinear system model is to define how the commands, throttle (u_1), rudder (u_2), pitch (u_3), and roll (u_4), relate to the PWM inputs.

$$p_1 = u_1 + u_2 + u_3 + u_4 + 1385.5$$
$$p_2 = u_1 - u_2 + u_3 - u_4 + 1385.5$$
$$p_3 = u_1 - u_2 - u_3 + u_4 + 1385.5$$
$$p_4 = u_1 + u_2 - u_3 - u_4 + 1385.5$$

These equations can be also be expressed as the commands in terms of the inputs; The 1385.5 μsec value refers to the pulse-width for all four motors in hover.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_3 \end{bmatrix} - \begin{bmatrix} 1385.5 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The resulting nonlinear mathematical model is used in an embedded MATLAB function in Simulink (Figure E.4), and provided inputs ranging from -1 to +1. The system responded as expected.

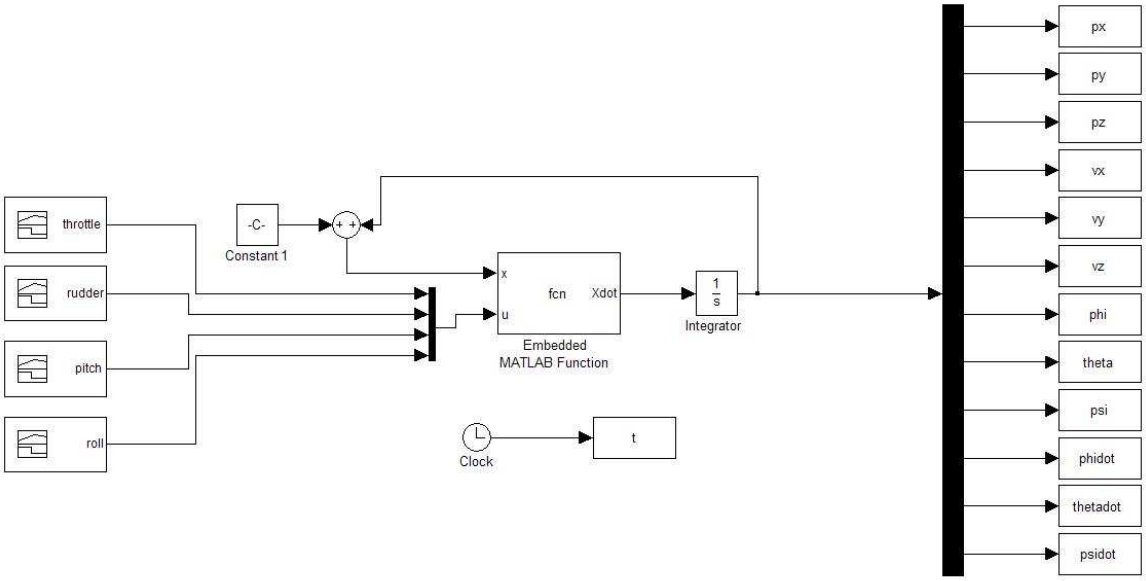


Figure E.4: Open-Loop Test Setup for the Quadrotor. Throttle, rudder, pitch, and roll commands were varied from -1 to +1 to observe the response of the quadrotor dynamics model. The model behaved as expected.

Bibliography

1. Allen D. Wu, Eric N. Johnson and Alison A. Proctor. "Visual-aided inertial navigation for flight control". *AIAA Journal of Aerospace Computing, Information, and Communication*, 2:348–360, 2005.

2. Dorato, Peter et al. *Linear Quadratic Control, An Introduction*. Krieger Publishing Company, Malabar, Florida, 2000.

3. Ebcin, S. and M. Veth. "Tightly-Coupled Image-Aided Inertial Navigation Using the Unscented Kalman Filter". *AFIT internal publication*, 2007.

4. Enns, Russell and Jennie Si. "Helicopter Trimming and Tracking Control Using Direct Neural Dynamic Programming". *IEEE Transactions on Neural Networks*, 14(4), July 2003.

5. Hajiyev, C. and M.A. Tutucu. "Development of GPS aided INS via Federated Kalman filter". *Proceedings of the Recent Advances in Space Technologies (RAST)*. Nov 2003.

6. Hendershot, J.R. and Tje Miller. *Design of Brushless Permanent-Magnet Motors*. Magna Physics Publishing, Hillsboro, OH, 1994.

7. Houwu Bai, Xubo Song, Eric Wan and Andriy Myronenko. "Vision-only Navigation and Control of Unmanned Aerial Vehicles Using the Sigma-Point Kalman Filter". 2007 ION NTM Conference, 2007.

8. J.E. Corban, A.J. Calise and J.V.R. Prasad. "Helicopter flight control design using a learning control approach". *Proceedings of Decision and Control, 37th IEEE Conference*. December 1998.

9. Jiang, Han J. Wang Y., Z. and Q. Song. "Enhanced LQR Control for Unmanned Helicopter in Hover". *Proceedings of Systems and Control in Aerospace and Astronautics, 2006, ISSCAA 2006*. January 2006.

10. John C. Morris, Michiel van Nieuwstadt and Pascale Bendotti. "Identification and Control of a Model Helicopter in Hover". *Proceedings of the American Control Conference*. June 1994.

11. Johnson, Eric N. and Suresh K. Kannan. "Adaptive Flight Control for an Autonomous Unmanned Helicopter". *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*. August 2002.

12. Julier, Simon J. and Jeffrey K. Uhlmann. "A New Extension of the Kalman Filter to Nonlinear Systems". *Proc. of AeroSense: The 11th Symp. on Aerospace/Defence Sensingm Simulation, and Controls*. 1997.

13. Ljung, Lennart. *System Identification, Theory for the User*. Prentice Hall, Inc., Englewood Cliffs, New Jersey 07632, 1987.

14. Lowe, David G. "Distinctive Image Features from Scale-Invariant Keypoints". *International Journal of Computer Vision*, 60(2):91–110, 2004.

15. Maybeck, Peter S. *Stochastic Models Estimation and Control, Vol II.* Academic Press, Inc., Orlando, Florida 32887, 1982.

16. Maybeck, Peter S. *Stochastic Models Estimation and Control, Vol III.* Academic Press, Inc., Orlando, Florida 32887, 1982.

17. Maybeck, Peter S. *Stochastic Models Estimation and Control, Vol I.* Navtech Book and Software Store, Orlando, Florida 32887, 1994.

18. McMurrough, Chris and Kyle French. *Real-Time Motion Capture Flight Control System User's Manual.* Air Force Research Laboratory, AFRL/RB, Wright Patterson Air Force Base, OH, 1.0 edition, July 2008.

19. Mejias, Saripalli Srikanth Cervera Pascual, Luis O. and Gaurav S. Sukhatme. "Visual Servoing for Tracking Features in Urban Areas Using an Autonomous Helicopter". *IEEE International Conference on Robotics and Automation.* 2006.

20. van der Merwe, R. and E.A. Wan. "Sigma-Point Kalman Filters for Integrated Navigation". *Proceedings of the 60th Annual Meeting of The Institute of Navigation (ION).* June 2004.

21. Misra, Pratap and Per Enge. *Global Positioning System, Signals, Measurements, and Performance, 2nd Edition.* Ganga-Jamuna Press, Lincoln, Massachusetts 01773, 2001.

22. Nelles, Oliver. *Nonlinear System Identification.* Springer-Verlag Berlin Heidelberg, Germany, 2001.

23. Office of the Secretary of Defense. "Unmanned Systems Roadmap 2007-2032", December 2007.

24. Ogata, Katsuhiko. *Modern Control Engineering.* Prentice Hall, Upper Saddle River, New Jersey 07458, 2002.

25. Pachter, M. "EENG 534, Inertial Navigation Systems, Class Handout", 2007. Graduate-level course taken at Air Force Institute of Technology.

26. Pachter, Meir and Alec Porter. "INS Aiding by Tracking an Unknown Ground Object - Theory". *Proceedings of the American Control Conference*, volume 2, 1260–1265. 2003.

27. Qi, Song and Han Jian-Da. "An Adaptive UKF Algorithm for the State Parameter Estimations of a Mobile Robot". *Acta Automatica Sinica*, 34(1), January 2008.

28. R. Sutton, T. Xu, W. Naeem and A. Tiano. "The design of a navigation, guidance, and control system for an unmanned surface vehicle for environmental monitoring." *Journal of Engineering for the maritime Environment*, 222:67–79, 2008.

29. Stephanik, Michael J. *A Quadrotor Sensor Platform*. Ph.D. thesis, Russ College of Engineering and Technology, Ohio University, November 2008.

30. Titterton, D.H. and J.L. Weston. *Strapdown Inertial Navigation Technology*. Institution of Electrical Engineers, Stevenage, UK, 2nd edition, 2004.

31. Veth, M. "EENG 562, Feedback Control, Handout", 2007. Graduate-level course taken at Air Force Institute of Technology.

32. Veth, M. "EENG 765, Stochastic Estimation and Control Coursework", 2008. Graduate-level course taken at Air Force Institute of Technology.

33. Veth, Michael J. *Fusion of Imaging and Inertial Sensors for Navigation*. Ph.D. thesis, Graduate School of Engineering, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, September 2006. AFIT/DS/ENG/06-09.

34. Wan, E.A. and R. van der Merwe. "The Unscented Kalman Filter for Nonlinear Estimation". Adaptive Systems for Signal Processing, Communications, and Control Symposium, 2000.

35. Wei, M. and K. P. Schwarz. "A Strapdown Inertial Algorithm Using an Earth-Fixed Cartesian Frame". *Journal of the Institute of Navigation*, 37(2):153–167, Summer 1990.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
26-03-2009	Master's Thesis	June 2008-March 2009

4. TITLE AND SUBTITLE

Model-Based Control using Model and Mechanization Fusion Techniques for Image-Aided Navigation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Hendrix, Constance D., Capt, USAF

5d. PROJECT NUMBER
JON# 09-224

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765 DSN: 785-3636

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GE/ENG/09-19

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory, Advanced Guidance Division (AFMC)
Attn: Dr. Mikel Miller
101 West Eglin Blvd, Bldg 13
Eglin AFB, FL 32542
mikel.miller@eglin.af.mil COMM (850) 882-4033

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RWG

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Unmanned aerial vehicles are no longer used for just reconnaissance. Current requirements call for smaller autonomous vehicles that replace the human in high-risk activities. Many times these activities are performed in GPS-degraded environments. Without GPS providing today's most accurate navigation solution, autonomous navigation in tight areas is more difficult. Today, image-aided navigation is used and other methods are explored to more accurately navigate in such areas (e.g., indoors). This thesis explores the use of inertial measurements and navigation solution updates using cameras with a model-based Linear Quadratic Gaussian controller. To demonstrate the methods behind this research, the controller will provide inputs to a micro-sized helicopter that allows the vehicle to maintain hover. A new method for obtaining a more accurate navigation solution was devised, originating from the following basic setup. To begin, a nonlinear system model was identified for a micro-sized, commercial, off-the-shelf helicopter. This model was verified, then linearized about the hover condition to construct an Linear Quadratic Regulator (LQR). The state error estimates, provided by an Unscented Kalman Filter using simulated image measurement updates, are used to update the navigation solution provided by inertial measurement sensors using strapdown mechanization equations. The navigation solution is used with a reference signal to determine the position and heading error. This error, along with other states, is fed to the LQR, which controls the helicopter. Research revealed that by combining the navigation solution from the INS mechanization block with a model-based navigation solution, and combining the INS error model and system model during the time propagation in the UKF, the navigation solution error decreases by 20%. The equations used for this modification stem from state and covariance combination methods utilized in the Federated Kalman Filter.

15. SUBJECT TERMS

Model-based, Linear Quadratic Gaussian, Unscented Kalman Filter, micro air vehicle, unmanned, model fusion, LQR, LQG, mechanization fusion, image-aided navigation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
REPORT	**ABSTRACT**	**c. THIS PAGE**			Lt Col Michael Veth (ENG)
U	**U**	**U**	**UU**	141	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-6565x4541; email: michael.veth@afit.edu