VERIFICATION OF A DUAL-STATE EXTENDED KALMAN FILTER WITH LIDAR-ENABLED AUTONOMOUS HAZARD-DETECTION FOR PLANETARY LANDERS

by

Peter J. Jorgensen, B.S.

A Thesis Submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

Milwaukee, Wisconsin

May 2015

ABSTRACT

VERIFICATION OF A DUAL-STATE EXTENDED KALMAN FILTER WITH LIDAR-ENABLED AUTONOMOUS HAZARD-DETECTION FOR PLANETARY LANDERS

Peter J. Jorgensen, B.S.

Marquette University, 2015

This thesis presents a mathematical model for a LIDAR-enabled Terrain- and Hazard-Relative Navigation sensor and the design and implementation of a dual-state extended Kalman filter. The extended Kalman filter equations are presented in summary. Mathematical models for an altimeter, a velocimeter, a star tracker, and a lidar-enabled mapping/tracking sensor are presented in depth. An explanation of the software designed for computer simulation is included.

It is proved through this analysis that, when implemented as part of a well-tuned extended Kalman Filter and in combination with other sensors, the proposed model for a lidar-enabled mapping/tracking sensor significantly reduces estimation error. This enables accurate navigation capable of precision descent-to-landing scenarios.

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to Dr. Robert H. Bishop for supporting me, both academically and economically, during the course of this work and my graduate studies. Special thanks to my thesis committee for their flexibility and feedback. I would also like to thank Dr. Timothy P. Crain for his guidance and sharing his expertise.

Finally, eternal thanks to my friends and family for their support and encouragement. I would not have made it without you.

TABLE OF CONTENTS

TABLE OF CONTENTS — *Continued*

TABLE OF CONTENTS — *Continued*

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES — *Continued*

CHAPTER 1
Introduction

1.1 Problem Statement and Motivation

Autonomous precision navigation is a capability that is indispensable for space exploration[1]. Whether it is landing a rover on Mars or docking with the International Space Station, guidance, navigation, and control (GN&C) require accuracy and precision. The obstacles to overcome in descent-to-landing scenarios are particularly complex, as delivering science or supplies to a specific point on the surface requires knowledge of the state of both the spacecraft and the landing point. This ideal of anytime, anywhere landing is achieved through the use of a dual-state Extended Kalman Filter (EKF).

As exploration of space continues, placing vehicles and assets near points of interest will be critical to efficient use of scientific resources. Current autonomous landing capabilities yield asset placement with distances from the targeted landing position measured in kilometers[2]. Any improvements in landing accuracy enable better use of multi-million dollar assets that are sent to other planets, comets, or asteroids, as the assets will require less time spent traveling to reach the ideal target areas for their designated missions. The challenging constraint to achieving anytime, anywhere landings is that the system must be able to operate in any lighting conditions.

The problem of precision anytime, anywhere landings can be effectively solved through use of a combination of navigation sensors including a vehicle-ground relative altimeter, doppler-lidar velocimeter, star-tracker attitude sensor, and a lidar-enabled mapping/tracking sensor to be known as the Terrain/Hazard Relative Navigation (THRN) sensor. Fusing these sensors' measurements into the vehicle's estimated state is done by the EKF system. To prove that this combination of sensors is effective in producing an accurate navigation solution, the vehicle, sensor, and environment dynamics must be modeled and the dual-state EKF implemented in simulation.

1.2 State of the Art

The current sophistication of autonomous landing systems was proven in August of 2012, when the Mars Science Laboratory (MSL) landed on the Martian surface just 2385 m from the intended target inside the Gale Crater. Previous landings on Mars were characterized by large footprints, on the order of a 100 x 15 km ellipse. The predicted footprint for MSL was 21 x 7 km. This increase in accuracy was made through extensive use of computer simulation and new landing technologies, such as the "Sky Crane."[2] Previously, the Apollo Lunar landings were characterized by landing accuracies of 1 km with a pilot in-the-loop.

Hazard detection systems have been employed in aerospace systems for decades, including but not limited to landing-assistance systems for airplanes [3] and vision-based hazard detection[4][5]. However, these systems rely on pre-deployed navigation assets or the requirement to land in well-lit conditions. Therefore, they cannot be used for anytime, anywhere landing scenarios.

The NASA Autonomous Landing Hazard Avoidance Technology (ALHAT) project is the main project underway seeking to develop transformational navigation techniques for use aboard spacecraft during rendezvous and entry, descent, and landing scenarios. The objectives are to operate autonomously, perform precision landing, detect hazards and avoid hazards. The ALHAT development project was chartered by NASA Headquarters in October 2005. The ALHAT sensor consists of a lidar system that creates a digital elevation map of a targeted landing surface. Additional sensors include a Doppler-lidar instrument capable of providing precision velocity vectors, vehicle ground relative altitude, and attitude.[6] As of June 2014, the sensor was tested unsuccessfully in closed-loop flight tests aboard the Morpheus vehicle at Kennedy Space Center, Florida. Reasons for the unsuccessful test include image clarity issues, image correlation issues, and that the sensor drove the navigation solution away from a GPS-based solution[7].

1.3 Objectives

To enable autonomous landing for future missions, the proposed lidar-enabled mapping/tracking navigation system must be able to:

- land autonomously anywhere, anytime within tens of meters if there are navigation assets in place and/or precise maps

- land autonomously anywhere, anytime within 1 kilometer in the absence of navigation assets or precise maps
- detect 0.5 m tall surface features, 10-degree slopes, and pre-deployed assets.

Feature recognition must be done at sufficient altitude to enable trajectory diversion to avoid undesirable landing areas and to efficiently maneuver to the re-designated target. It is assumed that this sensor exists and can provide the required measurement to the EKF system, based on published information on the ALHAT project sensor.

This thesis provides a summary of the EKF equations, a derivation of system dynamics, the mathematical formulation of sensor models, and an overview of the developed simulation for testing the EKF. Filter verification testing is done through simulation analysis including *Monte Carlo* covariance analysis, the development of an error budget, and sensitivity analysis.

1.4 Thesis Outline

Chapter 2 discusses the claimed contributions this thesis makes to the state of the art. Chapter 3 covers the motivation behind using an estimator and the estimation framework of the EKF, including both the propagation and update phases of the filter. This is directly followed by Chapter 4, the mathematical derivation of the spacecraft dynamics, including translational, rotational, and error dynamics. Chapter 5 describes the mathematical derivations of the sensor models. Chapters 6 and 7 respectively give an overview of the simulation software framework and simulation results. The conclusion in Chapter 8 summarizes the results and outlines possible extensions to this thesis.

1.5 Mathematical Notation

The mathematical notation used throughout this thesis is as follows:

- Vector quantities are in lowercase boldface, i.e. $\mathbf{r}$
- Scalar quantities are in lowercase normal weight, i.e. a_0
- Matrix quantities are in uppercase boldface, i.e. $\mathbf{T}$
- Quaternions are indicated by a vector quantity with an overbar, i.e. $\bar{\mathbf{q}}$.

Also, the vector part of a quaternion is simply the same boldface variable without the overbar i.e. $\mathbf{q}$, and the scalar part is the same variable, but normal weight i.e. q. Unless indicated, all quaternions adhere to the unity norm constraint.

Quaternion multiplication is defined:

$$\bar{\mathbf{p}} \otimes \bar{\mathbf{q}} = \begin{bmatrix} \mathbf{p} \\ p \end{bmatrix} \otimes \begin{bmatrix} \mathbf{q} \\ q \end{bmatrix} = \begin{bmatrix} p\mathbf{q} + q\mathbf{p} - \mathbf{p} \times \mathbf{q} \\ pq - \mathbf{p} \cdot \mathbf{q} \end{bmatrix}. \tag{1.1}$$

This definition of quaternion multiplication differs from Hamilton's original definition with the negative cross-product quantity. This is done because it allows the composition of a sequence of quaternion rotations to be written in the same order as a sequence of transformation-matrix rotations[8].

Several symbols are used to denote specific attributes of a variable. The hat $\{\hat{\cdot}\}$ indicates an estimated quantity, while the same variable without the hat is the true quantity. The dot $\{\dot{\cdot}\}$ indicates the temporal derivative. The double vertical bars $\|\cdot\|$ indicate the vector or matrix 2-norm. The transpose is indicated by a superscript "T" $\{\cdot\}^T$, and the inverse is indicated by the superscript "-1" $\{\cdot\}^{-1}$.

The cross-product matrix $[\cdot\times]$ is defined for two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$ such that $\mathbf{a} \times \mathbf{b} = [\mathbf{a}\times]\,\mathbf{b}$, where

$$[\mathbf{a}\times] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \tag{1.2}$$

Note that this follows the anticommutative property of the cross product where $[\mathbf{a}\times]\,\mathbf{b} = -\,[\mathbf{b}\times]\,\mathbf{a}$.

The diagonal matrix of a vector is defined as:

$$[\mathbf{a}\diagdown] = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{bmatrix}. \tag{1.3}$$

Note that $[\mathbf{a}\diagdown]\,\mathbf{b} = [\mathbf{b}\diagdown]\,\mathbf{a}$.

The off-diagonal symmetric matrix is defined as:

$$[\mathbf{a} \mid \times \mid] = \begin{bmatrix} 0 & a_3 & a_2 \\ a_3 & 0 & a_1 \\ a_2 & a_1 & 0 \end{bmatrix}. \tag{1.4}$$

Note that $[\mathbf{a} \mid \times \mid] \, \mathbf{b} = [\mathbf{b} \mid \times \mid] \, \mathbf{a}$.

1.6 Reference Frames

Reference frames are denoted by a series of letters. Table 1.1 indicates the superscript-frame pairings.

Table 1.1: Reference frame definitions

Superscript	Reference frame
i	Earth-centered inertial
f	Earth-centered Earth-fixed
UVW	Takeoff-position-fixed
ilp	Initial-landing-position-fixed frame
b	Spacecraft body-fixed
sr	Star-map fixed
$thrn$	Lidar-enabled mapping/tracking sensor fixed
vel	Velocimeter sensor fixed
alt	Altimeter sensor fixed
sc	Star-tracker camera fixed

The transformation between two reference frames is denoted by a capital "T" with a subscript to specify the from-frame and a superscript to specify the to-frame. For example, the transformation from the inertial frame to the spacecraft body frame is $\mathbf{T}_i^b$.

1.7 Tools and Software

All simulation software is written and developed in MATLAB version 2014b[9]. The simulation encompasses the environmental modeling of Earth's rotation and gravity, nominal flight trajectory generation, EKF implementation, and post-flight analysis including generation of figures.

CHAPTER 2
Contributions to State of the Art

There are two main contributions this thesis makes to the state of the art: first, a mathematical model for a Terrain/Hazard-Relative Navigation (THRN) sensor is proposed and its derivation shown; second, software to implement an Extended Kalman Filter (EKF) is written to test the proposed sensor model in a complex, dual-state estimation system. The software implements the standard EKF propagation and update equations as well as the sensor models as derived for this thesis. The sensor models are based on basic principles of the functionality for each sensor, and were derived specifically for this thesis. Using estimation filter analysis techniques the simulation is tested, and the resulting performance of the EKF implementation and the proposed THRN sensor model are analyzed for effectiveness.

2.1 A Terrain/Hazard-Relative Navigation Sensor Model

A sensor capable of mapping surface features and tracking them for navigation purposes exists and is being tested by the NASA Autonomous Landing Hazard Avoidance Technology (ALHAT) project. However, as of June 2014, the sensor was unsuccessful in flight tests, plagued by image correlation issues and image clarity issues[7]. This thesis derives a mathematical model for a sensor similar to the ALHAT sensor in functional capabilities of mapping, feature detection, and feature tracking.

The proposed model differs from the model used during the flight tests of the ALHAT system on board the Morpheus vehicle. The ALHAT sensor depends on a current state estimate in order to produce a new measurement because the sensor is on a gimbal and must be pointed to the estimated position of the tracked feature. Thus, in order to operate properly, the ALHAT sensor requires a very accurate state estimate, otherwise the image it produces for each measurement may not correlate to the previous image, resulting in a failed measurement.

It is believed that the model presented in this thesis implements the idea of Terrain/Hazard-Relative Navigation for inertial navigation better than the ALHAT system; the proposed THRN sensor model relies solely on output from the sensor, in the form of a vector to the tracked feature, and assumes that the sensor operates independently from the navigation state estimate. The resulting mathematical model is presented in full and in-depth and is one part of the contributions this thesis makes to the state of the art.

2.2 Dual-State Extended Kalman Filter

This thesis covers the design and implementation of a dual-state Extended Kalman Filter for state estimation, including the derivation of spacecraft system dynamics, estimation error dynamics, and sensor models. Since the EKF is a model-dependent filter, the modeling of the system and sensors is described in-depth. It is not claimed that the system dynamics or sensor models for the accelerometer, gyroscope, altimeter, velocimeter, and star camera are novel or unique. However, the models were derived from a basic understanding of the measurements they produce, and therefore the models and their derivations are included in full. Where the models were adapted from other sources, proper citations are made.

The dual-state nature of the proposed EKF is a new concept in inertial navigation. Historically, guidance and navigation for rendezvous requires knowledge of the spacecraft state relative to the target's state; this effectively assumes that the target's state is known[10][11]. Extending the rendezvous scenario to inertial navigation requires assuming the inertial state of the target is perfectly known, which is unrealistic in real-life scenarios. The navigation system and EKF proposed by this thesis estimate the inertial position of the intended landing position, which is corrupted by map-tie errors, in addition to the inertial spacecraft state. This enables rendezvous in an inertial navigation system. The software implementation of this dual-state EKF is the second major contribution of this thesis and is used to prove the THRN sensor model in simulation.

Figure 2.1 shows the portions of a control scheme that the software covers, including the related system inputs and outputs. Note that in a typical control scheme, there is a modeled system or plant and a controller included; they are greyed-out in the diagram to indicate that they are not included in this thesis. The simulation inputs include the true state vector $\mathbf{x}_k$ (fed through the sensors to create measurements) and measurement noise $\boldsymbol{\eta}_k$. These two quantities are added together

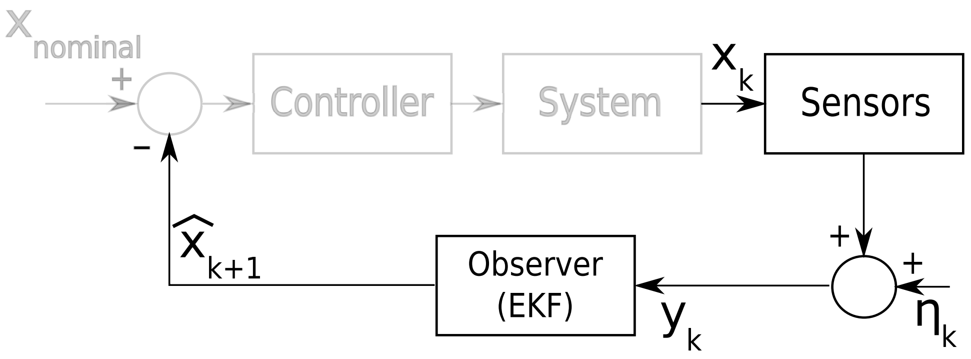

Figure 2.1: Control scheme diagram and focus of simulation software

and the noisy measurement is processed by the EKF, which has access to the prior estimated state $\hat{\mathbf{x}}_k$, to create the new state estimate $\hat{\mathbf{x}}_{k+1}$.

The simulation software was developed with filter verification and tuning in mind. Therefore, the ability to selectively activate sensors, noise sources, and inclusion of initial estimation error is built-in. The simulation does not include modeling of the system's environment, removing the need to include a controller and system model as part of the simulation loop. This simplified the software and focused analysis efforts on filter verification and tuning.

CHAPTER 3
Estimation Framework

This chapter covers the basics behind estimation theory, including why an estimator is used and a summary of the Kalman filter equations. Further, filter design is explained in the context of this problem. The chapter ends with a section on filter tuning and explanation of why this is an "art" rather than a science.

3.1 Motivation: What Is Estimation?

In control theory, an observer yields an estimate of the internal state of the system based on measurements of the input to and output from the system. Many forms of observers are available, and the Extended Kalman Filter (EKF) is one of these forms. The EKF is an extension to the optimal estimator for linear systems with additive white noise, known as the Kalman Filter.

There are multiple methods for obtaining a state quantity such as altitude above the ground for a given system, including model-based integration. For example, a rocket's altitude above a planet after takeoff is governed by equations of motion relating thrust, the vehicle's mass, and environmental effects such as gravity and atmospheric drag. A model of all of these effects can be produced, and, given an initial vehicle state, the vehicle's position at a future time can be computed. Similarly, the same rocket system's altitude above the ground can be directly measured by various sensors such as a laser altimeter, GPS receivers, and barometric sensors.

Sensors and models can both be used to get quantitative information about the rocket's state; however, sensors and models are of varying accuracy. For example, the model may be corrupted by numerical integration errors or effects that were not modeled; likewise, the sensors may be biased and include measurement noise. In both cases, the end-result solution for the rocket's altitude will include error. For accurate guidance and controls, it is desired to keep this error as small as possible.

The best way to reduce state estimation error is to combine the model solution with the measurement from sensors. In practice, sensor noise parameters can be known with relative certainty through pre-flight testing. Incorporating the sensor measurements of relatively-known noise characteristics into a current state estimate requires knowledge of how accurate the current state estimate is. If the measurement noise is known, and the current estimate accuracy is known, then the two can be combined optimally based on the relative statistics. For example, if the measurement noise on the rocket's altimeter is $\pm 5m$, and the current estimated altitude is thought to be accurate to $\pm 2m$, the estimated state, after being updated, will be based more on the previous state estimate than the new measurement.

The Extended Kalman Filter is capable of processing measurements like the example rocket's altimeter; further, as long as there exists a mathematical model for a sensor, the EKF is capable of processing its information. Statistical weighting of the state updates is done based on known sensor measurement noise characteristics and the current filter solution for state-estimate accuracy. The EKF depends on accurate models of the system dynamics and sensors; given both, the EKF estimator fuses model and sensor information to provide the best possible state estimate.

3.2 Motivation: Why Estimate?

For some systems, it is simple to directly measure quantities of interest. Many robots are used in conjunction with position and velocity sensors to provide direct, accurate state information. For other systems, measurements might indirectly provide the information that is desired or the proper sensors may be too expensive to incorporate into the system. Sensors may not provide measurements that are accurate enough to control the system, or measurements may be obtained too infrequently. Finally, where accurate state information is required, it may be necessary to model sensors with biases and other errors, which are parameters that cannot be known in advance. In these cases, it is imperative to use a state estimator. State estimation fuses prior knowledge of the system states with incoming measurement information. Various measures of error exist, and therefore there are different possible methods for minimizing this error. However, all estimators attempt to minimize the chosen definition of state error.

In the specific case of autonomous precision spacecraft descent and landing, two quantities are of interest: the spacecraft position and the position of the landing site. Sensors on the spacecraft measure quantities such as altitude, attitude, and velocity relative to the planetary surface. The focus of this thesis, a Terrain/Hazard Relative Navigation (THRN) sensor, measures the position of the landing site and selected features relative to the spacecraft. However, there is no sensor that exists that can definitively measure the inertial position of the spacecraft and the position of the landing site. Therefore, state estimation is necessary for fusing the available measurements together to produce the desired control information.

In short, estimation is not always a requirement for accurate control of a system. However, inertial guidance, navigation, and control is one area where estimation is fundamental to success, because directly measuring system state values is impossible.

3.3 The Extended Kalman Filter

Following the presentation of what estimation is and why it is required for inertial navigation, the next sections explain the Extended Kalman Filter propagation and state update equations. The equations as presented are not novel; their expression in detail in this thesis, however, builds a fundamental understanding of how the simulation software was developed to test the THRN sensor and this EKF estimator. As such, the equations are presented in-depth enough to provide the basis of understanding to be able to translate them into software.

In 1960, R. E. Kalman published his work on linear filtering and prediction[12], leading to what has become known as the Kalman filter, an algorithm that optimally fuses measurements into a non-stationary system's state based on the statistical measures of the system mean and covariance. The derivation of the Kalman filter is readily available, so it is not repeated here. For in-depth analysis, see Kalman's original paper among other sources[13][14][15]. Many variations on the original Kalman filter exist; the proposed navigation system uses the extended Kalman filter (EKF) which enables modeling of system dynamics and measurements as nonlinear equations while still using the optimality of the Kalman filter equations to update the system state and covariance through state-localized linearization using Taylor-series approximations[16].

Navigation is based on the combination of measurements from sensors and knowledge of the physical system and environment, such that the system states can be propagated forward in time while minimizing the accumulation of errors. The EKF is a recursive data processing algorithm that fuses asynchronous nonlinear measurements from sensors with potentially time-varying noise characteristics to update state variables[17][18].

The EKF uses prior knowledge of the system states and the associated error statistics. Although the EKF can fuse asynchronous measurements with prior state information, it is heavily dependent on accurate modeling of the sensors and environment. Application of the EKF is done in two stages: propagation, where state information and inertial measurement unit (IMU) data is used to predict future states, and update, where sensor measurements are used to alter state information and reduce accumulated errors. The propagation stage is most often done using the nonlinear models of the environment and the linear models of the error dynamics. The update stage uses the nonlinear measurement models and the linearized measurement sensitivities.

The EKF equations are very similar to those of the standard Kalman filter, although the dynamics are allowed to be nonlinear. Two Jacobian matrices are used to linearize the dynamics and measurement equations about the current state vector, used to generate the Kalman gain and propagate the system covariance. The nonlinear system model is assumed to be of the form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) + \mathbf{M}(t)\mathbf{w}(t) \tag{3.1}$$

where $\mathbf{x}(t)$ is the state of the system, $\mathbf{f}(\mathbf{x}(\mathrm{t}),\mathrm{t})$ is the nonlinear system model, and $\mathbf{w}(\mathrm{t})$ is the zero-mean, white-noise sequence process noise such that:

$$E\{\mathbf{w}(t)\} = \mathbf{0}, \quad E\{\mathbf{w}(t)\mathbf{w}^T(\tau)\} = \mathbf{Q}_{spec}\delta(t-\tau) \tag{3.2}$$

where $E\{\cdot\}$ is the expected value operator, $\delta(t-\tau)$ is the Dirac delta function, and $\mathbf{Q}_{spec}$ is a constant-power spectral density which is an input to the Kalman filter. Finally, $\mathbf{M}(t)$ is the process noise mapping matrix.

Similarly, the nonlinear measurement model is assumed to be of the form:

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k \tag{3.3}$$

where the subscript k denotes a discrete moment in time $t = t_k$ and the nonlinear measurement model evaluated at the state $\mathbf{x}_k = \mathbf{x}(t_k)$ is defined as $\mathbf{h}_k(\mathbf{x}_k)$.

Furthermore, $\mathbf{v}_k$ is the measurement noise which is assumed to be a zero-mean white noise sequence:

$$E\{\mathbf{v}_k\} = \mathbf{0}, \quad E\{\mathbf{v}_k \mathbf{v}_{k'}^T\} = \mathbf{R}_k \delta_{kk'} \tag{3.4}$$

where $\mathbf{R}_k$ is the noise covariance and is an input to the filter, and $\delta_{kk'}$ is the Kronecker delta. Additionally, it is assumed that the process noise and measurement noise are not correlated in time, or:

$$E\{\mathbf{w}(t)\mathbf{v}_k^T\} = \mathbf{0} \quad \forall \quad t, t_k \tag{3.5}$$

3.3.1 EKF Propagation Stage

The first step of the EKF cycle is propagation, where the state estimate after the previous update is integrated over time to create a new estimate to which the next update is applied. Propagation is done at a faster rate than state updates to emulate the continuous dynamics of the system. This generally enables more accurate solutions of the differential equations of motion governing the spacecraft.

Given a time interval such that $t \in \{t_{k-1}, t_k\}$, the state estimate dynamics of the form:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), t) \tag{3.6}$$

are numerically integrated from $t = t_{k-1}$ to $t = t_k$ with initial condition $\hat{\mathbf{x}}(t_{k-1})$.

Similarly, in order to propagate the state estimation error covariance, the equation:

$$\dot{\mathbf{P}}(t) = \mathbf{F}(\hat{\mathbf{x}}(t), t)\mathbf{P}(t) + \mathbf{P}(t)\mathbf{F}^T(\hat{\mathbf{x}}(t), t) + \mathbf{Q}_{spec}(t) \tag{3.7}$$

must be numerically integrated from $t = t_{k-1}$ to $t = t_k$, where:

$$\mathbf{F}(\hat{\mathbf{x}}(t), t) = \left[\left. \frac{\partial \mathbf{f}(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)} \right|_{\mathbf{x}(t) = \hat{\mathbf{x}}(t)} \right] \tag{3.8}$$

3.3.2 EKF Update Stage

When a discrete measurement comes into the filter at time $t = t_k$, the most up to date state estimate (from the propagation stage) is used along with the measurement to create an updated estimate of the state and state error covariance. The values immediately preceding an update are called *a priori* and denoted with a superscript $(-)$, and those immediately proceeding are *a posteriori* and denoted with a superscript $(+)$. The state estimate and state estimation error covariance update are calculated using the Joseph formula[17]:

$$\begin{aligned} \hat{\mathbf{x}}_k^{(+)} &= \hat{\mathbf{x}}_k^{(-)} + \mathbf{K}_k \left[\mathbf{y}_k - \mathbf{h}(\hat{\mathbf{x}}_k^{(-)}) \right] \\ \mathbf{P}_k^{(+)} &= \left[\mathbf{I}_{n \times n} - \mathbf{K}_k \mathbf{H}_k(\hat{\mathbf{x}}_k^{(-)}) \right] \mathbf{P}_k^{(-)} \left[\mathbf{I}_{n \times n} - \mathbf{K}_k \mathbf{H}_k(\hat{\mathbf{x}}_k^{(-)}) \right]^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \end{aligned} \tag{3.9}$$

and the Kalman gain is calculated:

$$\mathbf{K}_k = \mathbf{P}_k^{(-)} \mathbf{H}_k^T(\hat{\mathbf{x}}_k^{(-)}) \left[\mathbf{H}_k(\hat{\mathbf{x}}_k^{(-)}) \mathbf{P}_k^{(-)} \mathbf{H}_k^T(\hat{\mathbf{x}}_k^{(-)}) + \mathbf{R}_k \right]^{-1} \tag{3.10}$$

and $\mathbf{H}_k$ is the measurement sensitivity matrix defined as:

$$\mathbf{H}_k(\hat{\mathbf{x}}_k^{(-)}) = \left[\left. \frac{\partial \mathbf{h}_k(\mathbf{x}_k)}{\partial \mathbf{x}_k} \right|_{\mathbf{x}_k = \hat{\mathbf{x}}_k^{(-)}} \right] \tag{3.11}$$

3.4 Filter Design

The Kalman filter equations are well-known, but each implementation of the filter must be independent. This is because the filter's performance relies heavily on how well the system is modeled and how the filter is tuned.

3.4.1 Filter Modeling

Modeling of the system for descent-to-landing navigation requires knowledge of the spacecraft and of the environment. Translational dynamics are formed based on the planet's gravitational characteristics and non-gravitational accelerations measured by the Inertial Measurement Unit (IMU), and rotational dynamics are formed based on angular velocity measurements by the IMU. Sensors are modeled using system state variables, and for simulation purposes the sensor models include both truth models and models based on estimated quantities.

It is important to have models that accurately reflect the physical system, but some quantities cannot be measured, including sensor biases, scale-factor errors, and other sources of systematic error. Other parameters of the system may be variable and hard to measure, such as sensor noise characteristics. Thus, the state vector is augmented with systematic error variables to capture these effects.

Chapters 3 and 4 cover the modeling of the system dynamics and sensor measurements.

3.4.2 Filter Tuning

If the system were perfectly known, the models would lead to a perfect implementation of the EKF. However, due to uncertainties in the sensor models and the process of linearization to compute the Kalman gain, unknown errors are embedded in the filter. Two primary characteristics are used to compensate for these unknown errors: process noise in the form of a spectral density matrix $\mathbf{Q}_{spec}$ and sensor noise covariance matrix $\mathbf{R}_k$. Both matrices are assumed to be constant for this analysis.

The spectral density matrix $\mathbf{Q}_{spec}$ serves to increase the system covariance in combination with the system dynamics. This helps to prevent the system covariance from becoming too small, which would lead to a false sense of security in the system state estimate, ultimately rejecting new measurement data. In theory, if the system covariance converges to zero, the filter assumes it knows the system states perfectly. Thus, introducing process noise requires at least a minimum level of measurement processing, which means that the filter will be forced to continually update the state estimate.

The measurement noise covariance matrix $\mathbf{R}_k$ accounts for unknown errors introduced in the sensor models as well as sensor measurement noise. As can be seen in Eq. 3.10, $\mathbf{R}_k$ is in the denominator of the Kalman gain calculation. Thus, for increased values of measurement noise covariance, the Kalman gain is reduced, leading the filter to use less of the measurement residual to update the state during each filter cycle. The corollary is that for smaller measurement noise covariance values, the sensors are trusted more and the measurement residuals are more directly included in the state update[13].

The matrices $\mathbf{Q}_{spec}$ and $\mathbf{R}_k$ can be manipulated to ensure that, for designed trajectories, the filter state covariance always encapsulates the state error; this is the primary measure of a filter that performs well.

CHAPTER 4
System Dynamics

For effective navigation, an accurate model of the system must be known. This includes physical and propulsive characteristics of the spacecraft as well as knowledge of the environment including gravity and atmospheric interaction. Additionally, without accurate measurement of the non-gravitational acceleration and angular velocity of the spacecraft by an accelerometer/gyroscope package, precision landing would be impossible.

In addition to having a model for the translational and rotational system dynamics, a model for the estimation error dynamics is required to be able to predict new system states. The estimation errors describe the difference between the true spacecraft states and what is estimated, and knowledge of how these errors dynamically evolve is important to estimating new state values.

This chapter presents the translational, rotational, and estimation error dynamics for a general spacecraft in terms of the estimation errors and systematic errors such as sensor alignment biases.

4.1 Translational Dynamics

The translational dynamics of a spacecraft are effectively described with the two quantities of position and velocity. It is these two that are most important for precise control of a spacecraft, especially in a precision descent-to-landing scenario.

This section yields the translational-state differential equations, or the part of $\dot{\mathbf{x}} = f(\mathbf{x}, t)$ relating to position and velocity.

In dynamics, it is often easiest to derive equations of motion about the body's center of gravity (CG). This is useful, especially because gravitational forces are a function of the position of the CG. However, inertial navigation is more often done from the vantage point of the inertial measurement unit (IMU). This is because the IMU does not move relative to the spacecraft body, while the CG can due to burning fuel, moving payloads, etc. (although the relative position of the CG and the IMU was assumed to be static for this analysis). However, the position of the CG must be known to calculate the gravitational acceleration, so it is included in the derivation:

$$\mathbf{r}_{imu} = \mathbf{r}_{cg} + \mathbf{r}_{imu/cg}. \tag{4.1}$$

Taking the first temporal derivative of the IMU position yields:

$$\dot{\mathbf{r}}_{imu} = \dot{\mathbf{r}}_{cg} + \mathbf{v}_{imu/cg} + \boldsymbol{\omega}_b \times \mathbf{r}_{imu/cg} \tag{4.2}$$

where $\dot{\mathbf{r}}_{cg}$ is the velocity of the CG, $\mathbf{v}_{imu/cg}$ is the velocity of the IMU relative to the (potentially moving) CG, and $\boldsymbol{\omega}_b$ is the angular velocity of the spacecraft relative to the inertial frame, resulting in another velocity term for the IMU about the CG.

Differentiating again with respect to time yields:

$$\ddot{\mathbf{r}}_{imu} = \ddot{\mathbf{r}}_{cg} + \mathbf{a}_{imu/cg} + 2\boldsymbol{\omega}_b \times \mathbf{v}_{imu/cg} + \dot{\boldsymbol{\omega}}_b \times \mathbf{r}_{imu/cg} + \boldsymbol{\omega}_b \times \left(\boldsymbol{\omega}_b \times \mathbf{r}_{imu/cg}\right). \tag{4.3}$$

Note that $\ddot{\mathbf{r}}_{cg}$ is the same as $\mathbf{a}_{cg}$. This can be expanded into gravitational and non-gravitational acceleration:

$$\mathbf{a}_{cg} = \mathbf{a}_g + \mathbf{a}_{cg,nc} \tag{4.4}$$

where the "nc" subscript is used for any non-gravitational acceleration acting on the CG.

Substituting terms into Eq. 4.3 means that it can be written:

$$\ddot{\mathbf{r}}_{imu} = \mathbf{a}_g + \mathbf{a}_{ng} \tag{4.5}$$

where

$$\mathbf{a}_g = f\left(\mathbf{r}_{cg}\right) = f\left(\mathbf{r}_{imu} - \mathbf{r}_{imu/cg}\right) \tag{4.6}$$

and

$$\mathbf{a}_{ng} = \mathbf{a}_{cg,nc} + \mathbf{a}_{imu/cg} + 2\boldsymbol{\omega}_b \times \mathbf{v}_{imu/cg} + \dot{\boldsymbol{\omega}}_b \times \mathbf{r}_{imu/cg} + \boldsymbol{\omega}_b \times \left(\boldsymbol{\omega}_b \times \mathbf{r}_{imu/cg}\right). \tag{4.7}$$

It is this term, $\mathbf{a}_{ng}$, that is sensed by the IMU.

4.2 Rotational Dynamics

The rotational dynamics of spacecraft are relatively simple in that the angular velocity of the body is the same about the CG or the IMU. Thus, for simplicity, the IMU is used as the point of reference. The rotational quantity of most significance for inertial navigation is the rotation of the spacecraft relative to the inertial frame. The two methods used to deal with attitude are the rotation transformation matrix and the attitude quaternion. See [19] or [20] for in-depth information and derivation of the rotation transformation matrix and attitude quaternion with relation to dynamics.

This section yields the rotational-state differential equations, or the part of $\dot{\mathbf{x}} = f\left(\mathbf{x}, t\right)$ relating to attitude.

The rotation transformation matrix is a 3 by 3 matrix and is used to rotate vector quantities between frames of reference. Euler's theorem states that an angle of rotation and axis of rotation are all that is needed to completely describe an arbitrary attitude. This leads to Euler's formula for representing the rotation matrix as[21]:

$$\mathbf{T} = \mathbf{I}_{3\times3} + \sin\theta\left[\mathbf{e}\times\right] + \left(1-\cos\theta\right)\left[\mathbf{e}\times\right]^2 \tag{4.8}$$

where θ is the angle of rotation, $\mathbf{e}$ is the axis of rotation, and $[\cdot\times]$ represents the cross-product matrix. The rotation transformation matrix is subject to the constraints of a proper orthogonal transformation.

The attitude quaternion is used to describe the spacecraft's angular position. Euler angles are less effective due to singularities that can occur in certain spacecraft orientations. The inertial-to-body quaternion is made up of two parts; the vector quantity describing the axis of rotation, $\mathbf{q}$, and the scalar quantity describing the angle of rotation, q. These are written in column form:

$$\bar{\mathbf{q}} = \begin{bmatrix} \mathbf{q} \\ q \end{bmatrix} \tag{4.9}$$

where the over-bar is used to denote a quaternion,

$$\mathbf{q} = \sin\left(\frac{\theta}{2}\right)\mathbf{e}, \tag{4.10}$$

and

$$q = \cos\left(\frac{\theta}{2}\right). \tag{4.11}$$

Euler's formula may be written in terms of the quaternion quantities $\mathbf{q}$ and q:

$$\mathbf{T} = \left(q^2 + \mathbf{q}^T\mathbf{q}\right)\mathbf{I}_{3\times3} - 2q\left[\mathbf{q}\times\right] + 2\left[\mathbf{q}\times\right]^2. \tag{4.12}$$

For small angles of rotation, the quaternion can be approximated by a Taylor series expansion, to first order, as:

$$\delta\bar{\mathbf{q}} = \begin{bmatrix} \delta\mathbf{q} \\ \delta q \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\alpha} \\ 1 \end{bmatrix} \tag{4.13}$$

where $\delta\boldsymbol{\alpha}$ is a vector of small angle deviations, commonly ordered as roll-pitch-yaw angles.

Combining Eq. 4.13 and Eq. 4.12, the small-angle approximate rotation transformation matrix is:

$$\delta \mathbf{T} = \mathbf{I}_{3\times 3} - [\delta \boldsymbol{\alpha} \times] \,. \tag{4.14}$$

Returning to Eq. 4.9, differentiating with respect to time yields:

$$\dot{\bar{\mathbf{q}}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\cos\left(\frac{\theta}{2}\right)\mathbf{e}\dot{\theta} + \sin\left(\frac{\theta}{2}\right)\dot{\mathbf{e}} \\ -\frac{1}{2}\sin\left(\frac{\theta}{2}\right)\dot{\theta} \end{bmatrix} \tag{4.15}$$

where

$$\dot{\mathbf{e}} = \frac{1}{2}\left[[\mathbf{e}\times] - \cot\left(\frac{\theta}{2}\right)[\mathbf{e}\times]\right]\boldsymbol{\omega}_b \tag{4.16}$$

with $\boldsymbol{\omega}_b$ as the spacecraft angular velocity, and

$$\dot{\theta} = \mathbf{e}^T \boldsymbol{\omega}_b. \tag{4.17}$$

Making some final substitutions using Eqs. 4.10, 4.11, 4.16, and 4.17, into Eq. 4.15 yields:

$$\dot{\bar{\mathbf{q}}} = \frac{1}{2}\begin{bmatrix} q\boldsymbol{\omega}_b - \boldsymbol{\omega}_b \times \mathbf{q} \\ -\boldsymbol{\omega}_b^T \mathbf{q} \end{bmatrix} \tag{4.18}$$

which, when examined, is the definition of quaternion multiplication:

$$\dot{\bar{\mathbf{q}}}_i^b = \frac{1}{2}\bar{\mathbf{w}}_{b/i}^b \otimes \; \bar{\mathbf{q}}_i^b, \tag{4.19}$$

where $\bar{\boldsymbol{\omega}}$ is the pure quaternion created from the angular velocity of the spacecraft body relative to the inertial frame, as defined in the body-fixed frame:

$$\bar{\boldsymbol{\omega}} = \begin{bmatrix} \boldsymbol{\omega}_b \\ 0 \end{bmatrix} \tag{4.20}$$

Thus, for any attitude quaternion $\bar{\mathbf{q}}$ and angular velocity $\boldsymbol{\omega}_b$, the quaternion dynamics are known and can be used to propagate the attitude quaternion.

Finally, it is important to note that the attitude quaternion is constrained to be a unit quaternion, and therefore its vector norm must remain 1. Under time propagation, it is possible that the future attitude quaternion is non-unitary, and thus must be unitized by brute-force methods to maintain validity.

4.3 Estimation Error Dynamics

Now that the translational and rotational dynamics have been detailed, the spacecraft's inertial state can be propagated forward in time. For inertial navigation, however, the estimated errors for each state must also be propagated. This enables the filter to properly scale new measurement processing based on how accurate the known state is, especially if measurements are not processed regularly. In other words, if no new measurements come in for a given time, the state error estimate can grow accordingly, which factors into how the current state and new measurements are weighted in the state update phase.

Because this derivation is for an inertial navigation system, the superscript ^{i} will be used to indicate a vector in the inertial frame. Rotation transformation matrices are used to convert non-inertial vectors into the inertial frame for consistency.

The results of this section yield the makings of the Jacobian matrix **F**, or the matrix of partial derivatives of the state dynamics equations with respect to the state vector. This Jacobian matrix is used in the EKF to propagate the estimation error covariance.

4.3.1 Position Error Dynamics

The position error is defined as the true position minus the estimated position, and symbolized by $\delta\mathbf{r}$. Differentiating yields:

$$\delta\dot{\mathbf{r}}^i_{imu} = \dot{\mathbf{r}}^i_{imu} - \dot{\hat{\mathbf{r}}}^i_{imu} = \mathbf{v}^i_{imu} - \hat{\mathbf{v}}^i_{imu}, \tag{4.21}$$

which finally simplifies to

$$\delta\dot{\mathbf{r}}^i_{imu} = \delta\mathbf{v}^i_{imu} \tag{4.22}$$

where, in this case, $\delta\dot{\mathbf{r}}^i_{imu}$ is the rate of change of the position error, and $\delta\mathbf{v}^i_{imu}$ is the velocity error.

4.3.2 Velocity Error Dynamics

Following a similar formula, the velocity error is defined as the true velocity minus the estimated velocity. Differentiating yields:

$$\delta\dot{\mathbf{v}}^i_{imu} = \dot{\mathbf{v}}^i_{imu} - \dot{\hat{\mathbf{v}}}^i_{imu}. \tag{4.23}$$

Recalling that $\dot{\mathbf{v}}^i_{imu}$ is the summation of gravitational and non-gravitational acceleration, this equation can be written:

$$\delta\dot{\mathbf{v}}^i_{imu} = \mathbf{a}^i_g + \mathbf{T}^i_b\mathbf{a}^b_{ng} - \hat{\mathbf{a}}^i_g - \hat{\mathbf{T}}^i_b\hat{\mathbf{a}}^b_{ng}. \tag{4.24}$$

Expanding $\mathbf{a}^i_g$ into a first-order Taylor series approximation gives:

$$\mathbf{a}^i_g = \hat{\mathbf{a}}^i_g + \left[\left.\frac{\partial\mathbf{a}^i_g}{\partial\mathbf{r}^i_{cg}}\right|_{\mathbf{r}^i_{cg}=\hat{\mathbf{r}}^i_{cg}} \right] \delta\mathbf{r}^i_{cg}, \tag{4.25}$$

where

$$\mathbf{G}^i = \left.\frac{\partial\mathbf{a}^i_g}{\partial\mathbf{r}^i_{cg}}\right|_{\mathbf{r}^i_{cg}=\hat{\mathbf{r}}^i_{cg}}. \tag{4.26}$$

The gravity gradient matrix $\mathbf{G}^i$ is readily calculable granted the gravity environment and spacecraft position are known. Thus, substituting Eq. 4.26 and Eq. 4.25 into Eq. 4.24 simplifies to:

$$\delta\dot{\mathbf{v}}^i_{imu} = \mathbf{G}^i\delta\mathbf{r}^i_{cg} + \mathbf{T}^i_b\mathbf{a}^b_{ng} - \hat{\mathbf{T}}^i_b\hat{\mathbf{a}}^b_{ng}. \tag{4.27}$$

Since $\mathbf{r}^i_{cg}$ is not a part of the state vector, it must be rewritten in terms of state vector quantities. Recall that

$$\mathbf{r}^i_{cg} = \mathbf{r}^i_{imu} + \mathbf{T}^i_b \mathbf{r}^b_{cg/imu} \tag{4.28}$$

and, evaluating at the estimated state,

$$\hat{\mathbf{r}}^i_{cg} = \hat{\mathbf{r}}^i_{imu} + \hat{\mathbf{T}}^i_b \mathbf{r}^b_{cg/imu}. \tag{4.29}$$

It is assumed that $\mathbf{r}^b_{cg/imu}$ is well-known. Further, the first-order Taylor series approximation for the error rotation transformation matrix is:

$$\mathbf{T}^i_b - \hat{\mathbf{T}}^i_b = \delta \mathbf{T}^i_b = \hat{\mathbf{T}}^i_b \left[\delta\alpha^b \times\right] \tag{4.30}$$

where $\delta\alpha^b$ is the small-angle deviation of the spacecraft attitude in the body-frame. Recalling that the error is defined as truth minus estimate and substituting Eqs. 4.29 and 4.30 into Eq. 4.28 yields:

$$\delta\mathbf{r}^i_{cg} = \delta\mathbf{r}^i_{imu} - \hat{\mathbf{T}}^i_b \left[\mathbf{r}^b_{cg/imu} \times\right] \delta\alpha^b. \tag{4.31}$$

Substituting Eq. 4.31 into Eq. 4.27 yields:

$$\delta\dot{\mathbf{v}}^i_{imu} = \mathbf{G}^i \delta\mathbf{r}^i_{imu} - \mathbf{G}^i \hat{\mathbf{T}}^i_b \left[\mathbf{r}^b_{cg/imu} \times\right] \delta\alpha^b + \mathbf{T}^i_b \mathbf{a}^b_{ng} - \hat{\mathbf{T}}^i_b \hat{\mathbf{a}}^b_{ng}. \tag{4.32}$$

Recalling Eq. 4.30 and applying it to the last two terms in Eq. 4.32 yields:

$$\mathbf{T}^i_b \mathbf{a}^b_{ng} - \hat{\mathbf{T}}^i_b \hat{\mathbf{a}}^b_{ng} = \hat{\mathbf{T}}^i_b \mathbf{a}^b_{ng} + \hat{\mathbf{T}}^i_b \left[\delta\alpha^b \times\right] \mathbf{a}^b_{ng} - \hat{\mathbf{T}}^i_b \hat{\mathbf{a}}^b_{ng} \tag{4.33}$$

which, to first order, expands and simplifies to:

$$\mathbf{T}_b^i \mathbf{a}_{ng}^b - \hat{\mathbf{T}}_b^i \hat{\mathbf{a}}_{ng}^b = \hat{\mathbf{T}}_b^i \delta \mathbf{a}_{ng}^b + \hat{\mathbf{T}}_b^i \left[\delta\alpha^b \times\right] \hat{\mathbf{a}}_{ng}^b. \tag{4.34}$$

Applying Eq. 4.34 to Eq. 4.32 yields:

$$\delta \dot{\mathbf{v}}_{imu}^i = \mathbf{G}^i \delta \mathbf{r}_{imu}^i - \mathbf{G}^i \hat{\mathbf{T}}_b^i \left[\mathbf{r}_{cg/imu}^b \times\right] \delta\alpha^b + \hat{\mathbf{T}}_b^i \delta \mathbf{a}_{ng}^b + \hat{\mathbf{T}}_b^i \left[\delta\alpha^b \times\right] \hat{\mathbf{a}}_{ng}^b. \tag{4.35}$$

Considering that $\mathbf{a}_{ng}^b$ is not part of the state vector, $\delta \mathbf{a}_{ng}^b$ must be rewritten in terms of state variables. The true measured acceleration is:

$$\mathbf{a}_{ng}^b = \mathbf{a}_{ng,m}^b - \left[\mathbf{a}_{ng,m}^b \times\right] \mathbf{m}_a^b - \left[\mathbf{a}_{ng,m}^b \mid \times \mid\right] \mathbf{n}_a^b - \left[\mathbf{a}_{ng,m}^b \diagdown\right] \mathbf{s}_a^b - \mathbf{b}_a^b - \boldsymbol{\eta}_a^b \tag{4.36}$$

and the estimated measured acceleration is:

$$\hat{\mathbf{a}}_{ng}^b = \mathbf{a}_{ng,m}^b - \left[\mathbf{a}_{ng,m}^b \times\right] \hat{\mathbf{m}}_a^b - \left[\mathbf{a}_{ng,m}^b \mid \times \mid\right] \hat{\mathbf{n}}_a^b - \left[\mathbf{a}_{ng,m}^b \diagdown\right] \hat{\mathbf{s}}_a^b - \hat{\mathbf{b}}_a^b \tag{4.37}$$

where the subscript $_{meas}$ signifies a measured quantity, $\mathbf{b}_a$ is the constant accelerometer bias term, and $\boldsymbol{\eta}_a$ is the accelerometer measurement noise. Differencing Eqs. 4.36 and 4.37 yields:

$$\delta \mathbf{a}_{ng}^b = -\left[\mathbf{a}_{ng,m}^b \times\right] \delta \mathbf{m}_a^b - \left[\mathbf{a}_{ng,m}^b \mid \times \mid\right] \delta \mathbf{n}_a^b - \left[\mathbf{a}_{ng,m}^b \diagdown\right] \delta \mathbf{s}_a^b - \delta \mathbf{b}_a^b - \boldsymbol{\eta}_a^b \tag{4.38}$$

Substituting Eq. 4.38 into Eq. 4.35 yields the velocity error dynamics in terms of state variables and known quantities:

$$\begin{aligned}\delta\dot{\mathbf{v}}^i_{imu} =& \mathbf{G}^i\delta\mathbf{r}^i_{imu} - \mathbf{G}^i\hat{\mathbf{T}}^i_b\left[\mathbf{r}^b_{cg/imu}\times\right]\delta\alpha^b - \hat{\mathbf{T}}^i_b\left[\hat{\mathbf{a}}^b_{ng}\times\right]\delta\alpha^b - \hat{\mathbf{T}}^i_b\left[\mathbf{a}^b_{ng,m}\times\right]\delta\mathbf{m}^b_a \\ &- \hat{\mathbf{T}}^i_b\left[\mathbf{a}^b_{ng,m}\mid\times\mid\right]\delta\mathbf{n}^b_a - \hat{\mathbf{T}}^i_b\left[\mathbf{a}^b_{ng,m}\diagdown\right]\delta\mathbf{s}^b_a - \hat{\mathbf{T}}^i_b\delta\mathbf{b}^b_a - \hat{\mathbf{T}}^i_b\boldsymbol{\eta}^b_a.\end{aligned} \tag{4.39}$$

4.3.3 Attitude Error Dynamics

Unlike vector quantities, quaternions cannot be added or subtracted. The difference between two quaternions is found by the multiplicative error quaternion, defined as:

$$\delta\bar{\mathbf{q}}^b_i = \bar{\mathbf{q}}^b_i \otimes \left[\hat{\bar{\mathbf{q}}}^b_i\right]^{-1}. \tag{4.40}$$

Differentiating the multiplicative error quaternion yields:

$$\delta\dot{\bar{\mathbf{q}}}^b_i = \dot{\bar{\mathbf{q}}}^b_i \otimes \left[\hat{\bar{\mathbf{q}}}^b_i\right]^{-1} + \bar{\mathbf{q}}^b_i \otimes \left[\dot{\hat{\bar{\mathbf{q}}}}^b_i\right]^{-1} \tag{4.41}$$

recalling from Eq. 4.18 that $\dot{\bar{\mathbf{q}}}^b_i = \frac{1}{2}\bar{\boldsymbol{\omega}}^b_{b/i} \otimes \bar{\mathbf{q}}^b_i$. This leads to Eq. 4.41 being:

$$\delta\dot{\bar{\mathbf{q}}}^b_i = \frac{1}{2}\bar{\boldsymbol{\omega}}^b_{b/i} \otimes \bar{\mathbf{q}}^b_i \otimes \left[\hat{\bar{\mathbf{q}}}^b_i\right]^{-1} + \frac{1}{2}\bar{\mathbf{q}}^b_i \otimes \left[\hat{\bar{\mathbf{q}}}^b_i\right]^{-1} \otimes \left[\hat{\bar{\boldsymbol{\omega}}}^b_{b/i}\right]^{-1}, \tag{4.42}$$

which can be rewritten again using the definition of the multiplicative error quaternion as:

$$\delta\dot{\bar{\mathbf{q}}}^b_i = \frac{1}{2}\bar{\boldsymbol{\omega}}^b_{b/i} \otimes \delta\bar{\mathbf{q}}^b_i + \frac{1}{2}\delta\bar{\mathbf{q}}^b_i \otimes \left[\hat{\bar{\boldsymbol{\omega}}}^b_{b/i}\right]^{-1}. \tag{4.43}$$

Because the quaternion can be broken into a vector part and a scalar part, the error quaternion can similarly be broken up:

$$\delta\bar{\mathbf{q}}_i^b = \begin{bmatrix} \delta\mathbf{q} \\ \delta q \end{bmatrix} \tag{4.44}$$

which can then be substituted into Eq. 4.43 as part of the definition of quaternion multiplication:

$$\delta\dot{\bar{\mathbf{q}}}_i^b = \frac{1}{2}\begin{bmatrix} \delta q_i^b \delta\boldsymbol{\omega}_{b/i}^b - 2\hat{\boldsymbol{\omega}}_{b/i}^b \times \delta\mathbf{q}_i^b - \delta\boldsymbol{\omega}_{b/i}^b \times \delta q_i^b \\ -\left[\delta\boldsymbol{\omega}_{b/i}^b\right]^T \delta\mathbf{q}_i^b \end{bmatrix}. \tag{4.45}$$

The angular velocity error is defined as the true angular velocity minus the estimated quantity, where:

$$\boldsymbol{\omega}_{b/i}^b = \boldsymbol{\omega}_{b/i,m}^b - \left[\boldsymbol{\omega}_{b/i,m}^b \times\right]\mathbf{m}_g^b - \left[\boldsymbol{\omega}_{b/i,m}^b \mid \times \mid\right]\mathbf{n}_g^b - \left[\boldsymbol{\omega}_{b/i,m}^b \diagdown\right]\mathbf{s}_g^b - \mathbf{b}_g^b - \boldsymbol{\eta}_g^b \tag{4.46}$$

and

$$\hat{\boldsymbol{\omega}}_{b/i}^b = \boldsymbol{\omega}_{b/i,m}^b - \left[\boldsymbol{\omega}_{b/i,m}^b \times\right]\hat{\mathbf{m}}_g^b - \left[\boldsymbol{\omega}_{b/i,m}^b \mid \times \mid\right]\hat{\mathbf{n}}_g^b - \left[\boldsymbol{\omega}_{b/i,m}^b \diagdown\right]\hat{\mathbf{s}}_g^b - \hat{\mathbf{b}}_g^b, \tag{4.47}$$

and therefore the angular velocity error is:

$$\delta\boldsymbol{\omega}_{b/i}^b = -\left[\boldsymbol{\omega}_{b/i,m}^b \times\right]\delta\mathbf{m}_g^b - \left[\boldsymbol{\omega}_{b/i,m}^b \mid \times \mid\right]\delta\mathbf{n}_g^b - \left[\boldsymbol{\omega}_{b/i,m}^b \diagdown\right]\delta\mathbf{s}_g^b - \delta\mathbf{b}_g^b - \boldsymbol{\eta}_g^b. \tag{4.48}$$

Assuming small angles for each propagation step (recall Eq. 4.13) and neglecting higher-order terms, the attitude error quaternion dynamics simplify to:

$$\delta\dot{\mathbf{q}}_i^b = \begin{bmatrix} \frac{1}{2}\delta\dot{\boldsymbol{\alpha}} \\ 0 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} -\left[\boldsymbol{\omega}_{b/i,m}^b \times\right]\delta\mathbf{m}_g^b - \left[\boldsymbol{\omega}_{b/i,m}^b \mid\times\mid\right]\delta\mathbf{n}_g^b - \left[\boldsymbol{\omega}_{b/i,m}^b \diagdown\right]\delta\mathbf{s}_g^b - \delta\mathbf{b}_g^b - \boldsymbol{\eta}_g^b - \hat{\boldsymbol{\omega}}_{b/i}^b \times \left[\delta\boldsymbol{\alpha}^b\right] \\ 0 \end{bmatrix} \tag{4.49}$$

Extracting the terms that are directly related to the state vector yields the final attitude error dynamics equation:

$$\delta\dot{\boldsymbol{\alpha}}^b = \left[\hat{\boldsymbol{\omega}}_{b/i}^b \times\right]\delta\boldsymbol{\alpha}^b - \left[\boldsymbol{\omega}_{b/i,m}^b \times\right]\delta\mathbf{m}_g^b - \left[\boldsymbol{\omega}_{b/i,m}^b \mid\times\mid\right]\delta\mathbf{n}_g^b - \left[\boldsymbol{\omega}_{b/i,m}^b \diagdown\right]\delta\mathbf{s}_g^b - \delta\mathbf{b}_g^b - \boldsymbol{\eta}_g^b \tag{4.50}$$

where $\hat{\boldsymbol{\omega}}_{b/i}^b$ was defined in Eq. 4.47.

4.3.4 Bias Error Dynamics

Since sensors cannot be perfect, a large portion of the state vector is dedicated to estimating sensor biases. This enables the filter to correct for misaligned sensor axes, rate biases, etc. However, it is assumed that all biases are constant, and therefore when the bias terms are differentiated with respect to time, the results are all zero:

$$\delta\dot{\mathbf{b}}_a^b = \mathbf{0}_{3\times1} \tag{4.51}$$

$$\delta\dot{\mathbf{b}}_g^b = \mathbf{0}_{3\times1} \tag{4.52}$$

$$\delta\dot{\mathbf{b}}_{thrn}^{thrn} = \mathbf{0}_{3\times1} \tag{4.53}$$

$$\delta\dot{\mathbf{b}}_{vel}^{vel} = \mathbf{0}_{3\times1} \tag{4.54}$$

$$\delta\dot{\mathbf{b}}_{ilp}^{ilp} = \mathbf{0}_{3\times1} \tag{4.55}$$

$$\delta\dot{\mathbf{b}}_{alt}^{alt} = 0 \tag{4.56}$$

$$\delta\dot{\mathbf{b}}_{sc}^{sc} = \mathbf{0}_{3\times1} \tag{4.57}$$

$$\delta\dot{\mathbf{m}}_{a}^{b} = \mathbf{0}_{3\times1} \tag{4.58}$$

$$\delta\dot{\mathbf{n}}_{a}^{b} = \mathbf{0}_{3\times1} \tag{4.59}$$

$$\delta\dot{\mathbf{s}}_{a}^{b} = \mathbf{0}_{3\times1} \tag{4.60}$$

$$\delta\dot{\mathbf{m}}_{g}^{b} = \mathbf{0}_{3\times1} \tag{4.61}$$

$$\delta\dot{\mathbf{n}}_{g}^{b} = \mathbf{0}_{3\times1} \tag{4.62}$$

$$\delta\dot{\mathbf{s}}_{g}^{b} = \mathbf{0}_{3\times1} \tag{4.63}$$

which correspond to the accelerometer bias, gyro bias, THRN-sensor bias, velocimeter bias, ILP bias, altimeter bias, accelerometer axis misalignment, accelerometer axis nonorthogonality, accelerometer measurement scale factor, gyro axis misalignment, gyro axis nonorthogonality, and gyro measurement scale factor bias, respectively.

4.4 System Dynamics Summary

The previous sections detail the derivation of system dynamics for a spacecraft, including translational motion, rotational motion, and systematic error dynamics. The results are summarized here for readability.

4.4.1 State Dynamics Summary

The differential equations for position and velocity are:

$$\dot{\mathbf{r}}^i_{imu} = \mathbf{v}^i_{imu}$$

$$\dot{\mathbf{v}}^i_{imu} = \mathbf{a}^i_g + \mathbf{T}^i_b \mathbf{a}^b_{ng}$$

where $\mathbf{a}^i_g$ is a function of the position of the CG and $\mathbf{a}^b_{ng}$ is sensed by the IMU.

The differential equation for attitude is:

$$\dot{\mathbf{q}}^b_i = \begin{bmatrix} q^b_i \boldsymbol{\omega}^b_{b/i} - \boldsymbol{\omega}^b_{b/i} \times \mathbf{q}^b_i \\ -\left[\boldsymbol{\omega}^b_{b/i}\right]^T \mathbf{q}^b_i \end{bmatrix}$$

where $\bar{\boldsymbol{\omega}}^b_{b/i} = \begin{bmatrix} \boldsymbol{\omega}^b_{b/i} \\ 0 \end{bmatrix}$. Note that the resulting quaternion must be normalized after each propagation step.

4.4.2 Error Dynamics Summary

The error dynamics for position, velocity, and attitude are:

$$\delta\dot{\mathbf{r}}^i_{imu} = \delta\mathbf{v}^i_{imu}.$$

$$\begin{aligned}\delta\dot{\mathbf{v}}^i_{imu} =& \mathbf{G}^i\delta\mathbf{r}^i_{imu} - \mathbf{G}^i\hat{\mathbf{T}}^i_b\left[\mathbf{r}^b_{cg/imu}\times\right]\delta\boldsymbol{\alpha}^b - \hat{\mathbf{T}}^i_b\left[\hat{\mathbf{a}}^b_{ng}\times\right]\delta\boldsymbol{\alpha}^b - \hat{\mathbf{T}}^i_b\left[\mathbf{a}^b_{ng,m}\times\right]\delta\mathbf{m}^b_a \\ &- \hat{\mathbf{T}}^i_b\left[\mathbf{a}^b_{ng,m}\mid\times\mid\right]\delta\mathbf{n}^b_a - \hat{\mathbf{T}}^i_b\left[\mathbf{a}^b_{ng,m}\diagdown\right]\delta\mathbf{s}^b_a - \hat{\mathbf{T}}^i_b\delta\mathbf{b}^b_a - \hat{\mathbf{T}}^i_b\boldsymbol{\eta}^b_a.\end{aligned}$$

$$\delta\dot{\boldsymbol{\alpha}}^b = \left[\hat{\boldsymbol{\omega}}^b_{b/i}\times\right]\delta\boldsymbol{\alpha}^b - \left[\boldsymbol{\omega}^b_{b/i,m}\times\right]\delta\mathbf{m}^b_g - \left[\boldsymbol{\omega}^b_{b/i,m}\mid\times\mid\right]\delta\mathbf{n}^b_g - \left[\boldsymbol{\omega}^b_{b/i,m}\diagdown\right]\delta\mathbf{s}^b_g - \delta\mathbf{b}^b_g - \boldsymbol{\eta}^b_g.$$

Finally, the bias error dynamics are:

$$\delta\dot{\mathbf{b}}^b_a = \mathbf{0}_{3\times1}$$

$$\delta\dot{\mathbf{b}}^b_g = \mathbf{0}_{3\times1}$$

$$\delta\dot{\mathbf{b}}^{thrn}_{thrn} = \mathbf{0}_{3\times1}$$

$$\delta\dot{\mathbf{b}}^{vel}_{vel} = \mathbf{0}_{3\times1}$$

$$\delta\dot{\mathbf{b}}^{ilp}_{ilp} = \mathbf{0}_{3\times1}$$

$$\delta\dot{\mathbf{b}}^{alt}_{alt} = 0$$

$$\delta\dot{\mathbf{b}}^{sc}_{sc} = \mathbf{0}_{3\times1}$$

$$\delta\dot{\mathbf{m}}^b_a = \mathbf{0}_{3\times1}$$

$$\delta\dot{\mathbf{n}}^b_a = \mathbf{0}_{3\times1}$$

$$\delta\dot{\mathbf{s}}^b_a = \mathbf{0}_{3\times1}$$

$$\delta\dot{\mathbf{m}}^b_g = \mathbf{0}_{3\times1}$$

$$\delta\dot{\mathbf{n}}^b_g = \mathbf{0}_{3\times1}$$

$$\delta\dot{\mathbf{s}}^b_g = \mathbf{0}_{3\times1}.$$

CHAPTER 5
Sensor Modeling

Successful estimation relies on models of the sensors to create estimated measurements. The models are also differentiated with respect to the state vector to create a Jacobian matrix for computing the Kalman gain. The state variables used are shown in Table 5.1.

Table 5.1: State variable definitions

State variable	Definition
$\mathbf{r}^i_{imu}$	Inertial position of the IMU
$\mathbf{v}^i_{imu}$	Inertial velocity of the IMU
$\boldsymbol{\alpha}^b_{b/i}$	Body-fixed attitude deviation
$\mathbf{b}^b_{acc}$	Body-fixed accelerometer bias
$\mathbf{b}^b_{gyro}$	Body-fixed gyroscope bias
$\mathbf{b}^{thrn}_{thrn}$	THRN-frame THRN sensor bias
$\mathbf{b}^{vel}_{vel}$	Velocimeter-frame velocimeter sensor bias
$\mathbf{b}^{ilp}_{ilp}$	ILP-frame ILP bias (map-tie error)
b^{alt}_{alt}	Altimeter-frame altimeter sensor bias
$\mathbf{b}^{sc}_{sc}$	Star-camera frame star-camera sensor bias
$\mathbf{m}^b_{acc}$	Body-fixed accelerometer misalignment bias
$\mathbf{n}^b_{acc}$	Body-fixed accelerometer nonorthogonality bias
$\mathbf{s}^b_{acc}$	Body-fixed accelerometer scale-factor bias
$\mathbf{m}^b_{gyro}$	Body-fixed gyroscope misalignment bias
$\mathbf{n}^b_{gyro}$	Body-fixed gyroscope nonorthogonality bias
$\mathbf{s}^b_{gyro}$	Body-fixed gyroscope scale-factor bias

5.1 Accelerometer

The three-axis accelerometer is a necessary sensor for navigation as it provides the basis for translational dead-reckoning, where only the acceleration is integrated to propagate the system states. High-accuracy, high-precision accelerometers reduce error growth during dead-reckoning, but the measurements may be corrupted by axis misalignment, nonorthogonality of the axes, scale factor errors, random biases, and measurement noise. These error sources are estimated by the navigation filter and applied to the measured acceleration to provide a more accurate measurement.

The true measured acceleration can be written:

$$\mathbf{a}_{ng,m} = \left(\mathbf{I}_{3\times3} + \mathbf{\Gamma}_a\right)\left(\mathbf{I}_{3\times3} + \mathbf{S}_a\right)\left(\mathbf{a}_{ng} + \mathbf{b}_a + \boldsymbol{\eta}_a\right) \tag{5.1}$$

where $\mathbf{a}_{ng}$ is the true non-gravitational acceleration of the IMU, $\mathbf{b}_a$ is the random bias error, $\boldsymbol{\eta}_a$ is the random noise, $\mathbf{\Gamma}_a$ represents the axis-related errors:

$$\mathbf{\Gamma}_a = \begin{bmatrix} 0 & \gamma_{a,xz} & -\gamma_{a,xy} \\ -\gamma_{a,yz} & 0 & \gamma_{a,yx} \\ \gamma_{a,zy} & -\gamma_{a,zx} & 0 \end{bmatrix} \tag{5.2}$$

and $\mathbf{S}_a$ represents the measurement scale factor errors:

$$\mathbf{S}_a = \begin{bmatrix} S_{a,x} & 0 & 0 \\ 0 & S_{a,y} & 0 \\ 0 & 0 & S_{a,z} \end{bmatrix}. \tag{5.3}$$

The $\mathbf{\Gamma}_a$ matrix can be decomposed into a skew-symmetric matrix $\mathbf{M}_a = -\mathbf{M}_a^T$ and a symmetric matrix $\mathbf{N}_a = \mathbf{N}_a^T$. Then, the matrices $\mathbf{M}_a, \mathbf{N}_a, \mathbf{S}_a$ can be parameterized as three 3×1 vectors, $\mathbf{m}_a, \mathbf{n}_a, \mathbf{s}_a$, under the definitions found in Eqs. 1.2 - 1.4:

$$\mathbf{M}_a = -\left[\mathbf{m}_a \times\right] \tag{5.4}$$

$$\mathbf{N}_a = \left[\mathbf{n}_a \mid \times \mid\right] \tag{5.5}$$

$$\mathbf{S}_a = \left[\mathbf{s}_a \diagdown\right]. \tag{5.6}$$

Thus, substituting the parameterized quantities into the model yields:

$$\mathbf{a}_{ng,m} = \left(\mathbf{I}_{3\times3} - \left[\mathbf{m}_a \times\right] + \left[\mathbf{n}_a \mid \times \mid\right]\right)\left(\mathbf{I}_{3\times3} + \left[\mathbf{s}_a \diagdown\right]\right)\left(\mathbf{a}_{ng} + \mathbf{b}_a + \boldsymbol{\eta}_a\right) \tag{5.7}$$

A small-angle rotation matrix can be written as $\mathbf{T} = \mathbf{I}_{3\times3} - [\theta \times]$, and so it can be seen that $\mathbf{m}_a$ corresponds to the axis-misalignment parameters and $\mathbf{n}_a$ corresponds to the axis-nonorthogonality parameters.

The axis-error parameters, expanded and simplified to first-order terms, can be approximated by:

$$\left(\mathbf{I}_{3\times3} - \left[\mathbf{m}_a \times\right] + \left[\mathbf{n}_a \mid \times \mid\right]\right)\left(\mathbf{I}_{3\times3} + \left[\mathbf{s}_a \diagdown\right]\right) \approx \left(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_a\right) \tag{5.8}$$

where

$$\boldsymbol{\Delta}_a = \mathbf{M}_a + \mathbf{N}_a + \mathbf{S}_a \tag{5.9}$$

which leads to:

$$\mathbf{a}_{ng,m} = \left(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_a\right)\left(\mathbf{a}_{ng} + \mathbf{b}_a + \boldsymbol{\eta}_a\right). \tag{5.10}$$

Using the matrix inversion lemma[22]:

$$\left(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_a\right) = \mathbf{I}_{3\times3} - \boldsymbol{\Delta}_a\left(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_a\right)^{-1} \tag{5.11}$$

and applying the lemma recursively:

$$\left(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_a\right) = \sum_{k=0}^{n}\left(-\boldsymbol{\Delta}_a\right)^k + \left(-\boldsymbol{\Delta}_a\right)^{k+1}\left(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_a\right)^{-k}. \tag{5.12}$$

To first order, this yields the approximation:

$$\left(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_a\right) \approx \mathbf{I}_{3\times3} - \boldsymbol{\Delta}_a. \tag{5.13}$$

Solving Eq. 5.10 for $\mathbf{a}_{ng}$ and substituting in Eq. 5.13 yields:

$$\mathbf{a}_{ng} = \left(\mathbf{I}_{3\times3} - \boldsymbol{\Delta}_a\right)\mathbf{a}_{ng,m} - \mathbf{b}_a - \boldsymbol{\eta}_a \tag{5.14}$$

which expands and simplifies to:

$$\mathbf{a}_{ng} = \mathbf{a}_{ng,m} + \left[\mathbf{m}_a\times\right]\mathbf{a}_{ng,m} - \left[\mathbf{n}_a \mid \times \mid\right]\mathbf{a}_{ng,m} - \left[\mathbf{s}_a\diagdown\right]\mathbf{a}_{ng,m} - \mathbf{b}_a - \boldsymbol{\eta}_a. \tag{5.15}$$

Finally, rewriting gives the final form of the corrected measured acceleration:

$$\mathbf{a}_{ng} = \mathbf{a}_{ng,m} - \left[\mathbf{a}_{ng,m}\times\right]\mathbf{m}_a - \left[\mathbf{a}_{ng,m} \mid \times \mid\right]\mathbf{n}_a - \left[\mathbf{a}_{ng,m}\diagdown\right]\mathbf{s}_a - \mathbf{b}_a - \boldsymbol{\eta}_a \tag{5.16}$$

where $\mathbf{m}_a, \mathbf{n}_a, \mathbf{s}_a$, and $\mathbf{b}_a$ are all estimated in the filter state vector.

5.2 Gyroscope

The three-axis gyroscope measures angular velocity of the sensor, which is the same as the angular velocity of the spacecraft to which it is attached. These angular velocity measurements can be corrupted by axis misalignment, axis nonorthogonality, random biases, and noise. It is important to estimate these quantities, as the gyroscope is used to propagate the spacecraft attitude during dead-reckoning (in the absence of measurements).

The true measured angular velocity is written in terms of these errors:

$$\boldsymbol{\omega}_{b/i,m} = (\mathbf{I}_{3\times3} + \boldsymbol{\Gamma}_g)(\mathbf{I}_{3\times3} + \mathbf{S}_g)\left(\boldsymbol{\omega}_{b/i} + \mathbf{b}_g + \boldsymbol{\eta}_g\right) \tag{5.17}$$

where $\boldsymbol{\omega}_{b/i}$ is the true angular velocity of the gyroscope, $\mathbf{b}_g$ is the random bias error, $\boldsymbol{\eta}_g$ is the measurement noise error, $\boldsymbol{\Gamma}_g$ is the axis-related error:

$$\boldsymbol{\Gamma}_g = \begin{bmatrix} 0 & \gamma_{g,xz} & -\gamma_{g,xy} \\ -\gamma_{g,yz} & 0 & \gamma_{g,yx} \\ \gamma_{g,zy} & -\gamma_{g,zx} & 0 \end{bmatrix} \tag{5.18}$$

and $\mathbf{S}_g$ is the measurement scale-factor error:

$$\mathbf{S}_g = \begin{bmatrix} S_{g,x} & 0 & 0 \\ 0 & S_{g,y} & 0 \\ 0 & 0 & S_{g,z} \end{bmatrix}. \tag{5.19}$$

The $\boldsymbol{\Gamma}_g$ matrix can be decomposed into a skew-symmetric matrix and a symmetric matrix, defined $\mathbf{M}_g$ and $\mathbf{N}_g$ respectively, where $\mathbf{M}_g = -\mathbf{M}_g^T$ and $\mathbf{N}_g = \mathbf{N}_g^T$. Further, using the definitions in Eqs. 1.2 - 1.4, $\mathbf{M}_g, \mathbf{N}_g, \mathbf{S}_g$ can be rewritten in terms of three 3×1 vectors $\mathbf{m}_g, \mathbf{n}_g, \mathbf{s}_g$:

$$\mathbf{M}_g = -\left[\mathbf{m}_g \times\right] \tag{5.20}$$

$$\mathbf{N}_g = \left[\mathbf{n}_g \mid \times \mid\right] \tag{5.21}$$

$$\mathbf{S}_g = \left[\mathbf{s}_g \diagdown\right]. \tag{5.22}$$

This yields the true measured angular velocity as:

$$\boldsymbol{\omega}_{b/i,m} = \left(\mathbf{I}_{3\times3} - \left[\mathbf{m}_g \times\right] + \left[\mathbf{n}_g \mid \times \mid\right]\right)\left(\mathbf{I}_{3\times3} + \left[\mathbf{s}_g \diagdown\right]\right)\left(\boldsymbol{\omega}_{b/i} + \mathbf{b}_g + \boldsymbol{\eta}_g\right). \tag{5.23}$$

For small angles, a rotation transformation matrix can be written as $\mathbf{T} = \mathbf{I}_{3\times3} - [\boldsymbol{\theta}\times]$. This means that the quantity $[\mathbf{m}_g \times]$ corresponds to the axis misalignment error, while $[\mathbf{n}_g \mid \times \mid]$ is the axis nonorthogonality error.

Expanding the axis and scale factor error terms and keeping only the first-order terms allows writing them as a lump-sum parameter:

$$\left(\mathbf{I}_{3\times3} - \left[\mathbf{m}_g \times\right] + \left[\mathbf{n}_g \mid \times \mid\right]\right)\left(\mathbf{I}_{3\times3} + \left[\mathbf{s}_g \diagdown\right]\right) \approx \left(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_g\right) \tag{5.24}$$

where

$$\boldsymbol{\Delta}_g = \mathbf{M}_g + \mathbf{N}_g + \mathbf{S}_g \tag{5.25}$$

which leads to a simplification of the true measured angular velocity:

$$\boldsymbol{\omega}_{b/i,m} = \left(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_g\right)\left(\boldsymbol{\omega}_{b/i} + \mathbf{b}_g + \boldsymbol{\eta}_g\right). \tag{5.26}$$

Utilizing the matrix inversion lemma and applying it recursively yields an approximation for inverting the $(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_g)$ term:

$$\left(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_g\right) = \mathbf{I}_{3\times3} - \boldsymbol{\Delta}_g\left(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_g\right)^{-1}, \tag{5.27}$$

$$\left(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_g\right) = \sum_{k=0}^{n}\left(-\boldsymbol{\Delta}_g\right)^k + \left(-\boldsymbol{\Delta}_g\right)^{k+1}\left(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_g\right)^{-k}, \tag{5.28}$$

$$\left(\mathbf{I}_{3\times3} + \boldsymbol{\Delta}_g\right) \approx \mathbf{I}_{3\times3} - \boldsymbol{\Delta}_g. \tag{5.29}$$

Applying this approximation to the solution of Eq. 5.26 for the true angular velocity $\boldsymbol{\omega}_{b/i}$ yields:

$$\boldsymbol{\omega}_{b/i} = \left(\mathbf{I}_{3\times3} - \boldsymbol{\Delta}_g\right)\boldsymbol{\omega}_{b/i,m} - \mathbf{b}_g - \boldsymbol{\eta}_g. \tag{5.30}$$

Expanding $\boldsymbol{\Delta}_g$ as in Eq. 5.24 yields:

$$\boldsymbol{\omega}_{b/i} = \boldsymbol{\omega}_{b/i,m} + \left[\mathbf{m}_g\times\right]\boldsymbol{\omega}_{b/i,m} - \left[\mathbf{n}_g \mid \times \mid\right]\boldsymbol{\omega}_{b/i,m} - \left[\mathbf{s}_g\diagdown\right]\boldsymbol{\omega}_{b/i,m} - \mathbf{b}_g - \boldsymbol{\eta}_g. \tag{5.31}$$

Finally, this is rearranged according to Eqs. 1.2 - 1.4 as:

$$\boldsymbol{\omega}_{b/i} = \boldsymbol{\omega}_{b/i,m} - \left[\boldsymbol{\omega}_{b/i,m}\times\right]\mathbf{m}_g - \left[\boldsymbol{\omega}_{b/i,m} \mid \times \mid\right]\mathbf{n}_g - \left[\boldsymbol{\omega}_{b/i,m}\diagdown\right]\mathbf{s}_g - \mathbf{b}_g - \boldsymbol{\eta}_g \tag{5.32}$$

where $\mathbf{m}_g, \mathbf{n}_g, \mathbf{s}_g$, and $\mathbf{b}_g$ are estimated in the state vector.

5.3 Slant-Range Altimeter

Various forms of range finding exist, from radar to laser. Specifically, laser range finding using a lidar sensor has been useful to scientists for decades; the Apollo program placed a retroreflective array on the moon to enable high-accuracy earth-to-moon ranging[23]. Laser ranging is also used in navigation, and laser altimetry is one method of gathering highly accurate, very fast altitude measurements[24].

The purpose of an altimeter is to generate data about the distance from the sensor to the ground. There are two main kinds of altimeter: first-return, where a flash is output from the sensor and the first signal return is counted; and slant-range, where a focused beam is pointed at the target and only the return along that beam is counted. The slant-range altimeter is a more focused sensor, and the model is slightly more complicated as a result. For this analysis, a slant-range laser altimeter measurement model is derived.

5.3.1 Altimeter Measurement Model

In general, a slant-range altimeter returns a scalar quantity of distance to ground. To model this accurately, a few things must be known: altimeter pointing direction, altimeter position with respect to the planet's center, and information about the local terrain altitude. This derivation assumes that the local terrain is well-modeled by the oblate spheroid model of Earth, where two axes' radii are equal (co-planar with the equator) and greater than the radius to the north/south poles. Although the Earth's actual surface varies greatly, the modeled flight path across just hundreds of meters over a generally flat section of ground enables this simplifying assumption. The altimeter pointing direction is assumed well-known and static in the body-frame of reference. Finally, the altimeter position relative to the planet center is a function of the IMU position, the position of the altimeter with respect to the IMU, and the spacecraft attitude.

The derivation is done in the planet-fixed (ECEF) frame, since the spacecraft position is easily converted into the ECEF frame and the planetary parameters are well-defined in the ECEF frame.

The measurement distance to ground magnitude is defined as the scalar, ρ. Thus, the measurement is written:

$$y_k = \rho\left(\mathbf{r}^f_{alt}, \mathbf{k}^f_{alt}\right) + b_{alt} + \eta_{alt} \tag{5.33}$$

where b_{alt} is the altimeter bias, η_{alt} is the altimeter measurement noise, $\mathbf{r}^f_{alt}$ is the altimeter position in the ECEF frame defined as:

$$\mathbf{r}^f_{alt} = \mathbf{T}^f_i \mathbf{r}^i_{imu} + \mathbf{T}^f_i \mathbf{T}^i_b \mathbf{r}^b_{alt/imu} \tag{5.34}$$

where $\mathbf{T}^f_i$ is the planet-centered-inertial (ECI) to ECEF rotation transformation matrix, $\mathbf{T}^i_b$ is the spacecraft body frame to ECI rotation transformation matrix, and $\mathbf{r}^b_{alt/imu}$ is the altimeter position with respect to the IMU in the spacecraft body frame and is assumed well-known. Then, $\mathbf{k}^f_{alt}$ is the altimeter beam direction vector in the the ECEF frame, defined as:

$$\mathbf{k}^f_{alt} = \mathbf{T}^f_i \mathbf{T}^i_b \mathbf{T}^b_{alt} \mathbf{k}^{alt}_{alt} \tag{5.35}$$

where $\mathbf{T}^b_{alt}$ is the altimeter frame to spacecraft body frame rotation transformation matrix, and $\mathbf{k}^{alt}_{alt}$ is the altimeter pointing direction in the altimeter frame. Note that for simplicity, it is common to define $\mathbf{k}^{alt}_{alt}$ as wholly in a principle direction of the altimeter frame, i.e.:

$$\mathbf{k}^{alt}_{alt} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{5.36}$$

The estimated measurement quantity is equivalent to Eq. 5.33 evaluated at the estimated values for altimeter position and pointing direction:

$$\hat{y}_k = \hat{\rho}\left(\hat{\mathbf{r}}^f_{alt}, \hat{\mathbf{k}}^f_{alt}\right) + \hat{b}_{alt} \tag{5.37}$$

where

$$\hat{\mathbf{r}}^f_{alt} = \mathbf{T}^f_i \hat{\mathbf{r}}^i_{imu} + \mathbf{T}^f_i \hat{\mathbf{T}}^i_b \mathbf{r}^b_{alt/imu} \tag{5.38}$$

and

$$\hat{\mathbf{k}}^f_{alt} = \mathbf{T}^f_i \hat{\mathbf{T}}^i_b \mathbf{T}^b_{alt} \mathbf{k}^{alt}_{alt}. \tag{5.39}$$

5.3.2 Computing the Altimeter Measurement Magnitude

To create a valid measurement, the point of intersection on the planet's surface must be found. A geometrical approach to the problem leads to finding the intersection point(s) between a vector and the reference surface, most likely a spheroid or ellipsoid. Since the altimeter beam direction is known relative to the planetary body and the magnitude of a planet-center-to-surface vector is readily computable, equating vector magnitudes provides the answer.

The vector from the planet center to the altimeter beam intersection point, $\mathbf{r}_p^f$ must be equivalent regardless of what path is taken to the point:

$$\mathbf{r}_p^f = \mathbf{r}_{alt}^f + \rho \mathbf{k}_{alt}^f. \tag{5.40}$$

A point $\mathbf{r} = [x\, y\, z]^T$ is on the reference ellipsoid with semi-principal a, b, c if:

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2 = 1. \tag{5.41}$$

The Earth is modeled as an oblate ellipsoid where $a = b > c$. A diagonal scaling matrix is defined:

$$\mathbf{S} = \begin{bmatrix} \frac{1}{a} & 0 & 0 \\ 0 & \frac{1}{b} & 0 \\ 0 & 0 & \frac{1}{c} \end{bmatrix} \tag{5.42}$$

such that

$$\|\mathbf{Sr}\|_2^2 = 1. \tag{5.43}$$

Applying the scaling matrix $\mathbf{S}$ to Eq. 5.40 yields:

$$\mathbf{Sr}^f_{alt} + \rho\mathbf{Sk}^f_{alt} = \mathbf{Sr}^f_p \tag{5.44}$$

which under the definition in Eq. 5.43 can be written:

$$\left\|\mathbf{Sr}^f_{alt} + \rho\mathbf{Sk}^f_{alt}\right\|_2^2 = \|\mathbf{Sr}^f_p\|_2^2 = 1 \tag{5.45}$$

since $\mathbf{r}^f_p$ is defined as a point on the reference ellipsoid surface.

The vector dot product is equivalent to the vector 2-norm squared, so Eq. 5.45 can be expanded to an equation that is quadratic in ρ:

$$\rho^2\left(\mathbf{Sk}^f_{alt}\right)\cdot\left(\mathbf{Sk}^f_{alt}\right) + 2\rho\left(\mathbf{Sr}^f_{alt}\right)\cdot\left(\mathbf{Sk}^f_{alt}\right) + \left(\mathbf{Sr}^f_{alt}\right)\cdot\left(\mathbf{Sr}^f_{alt}\right) = 1. \tag{5.46}$$

Defining the quantities

$$\gamma = \frac{-\left(\mathbf{Sr}^f_{alt}\right)\cdot\left(\mathbf{Sk}^f_{alt}\right)}{\left(\mathbf{Sk}^f_{alt}\right)\cdot\left(\mathbf{Sk}^f_{alt}\right)} \tag{5.47}$$

and

$$\beta = \frac{1-\left(\mathbf{Sr}^f_{alt}\right)\cdot\left(\mathbf{Sr}^f_{alt}\right)}{\left(\mathbf{Sk}^f_{alt}\right)\cdot\left(\mathbf{Sk}^f_{alt}\right)} \tag{5.48}$$

and substituting into Eq. 5.46 yields:

$$0 = \rho^2 - 2\gamma\rho - \beta, \tag{5.49}$$

which can then be solved for ρ:

$$\rho = \gamma \pm \sqrt{\gamma^2 + \beta} \tag{5.50}$$

By geometry, this is a solution for where a vector intersects an ellipsoid. The desired result is the first intersection, so the final solution for ρ is:

$$\rho = \gamma - \sqrt{\gamma^2 + \beta} \tag{5.51}$$

This ends the solution for calculating the measurement vector magnitude. To generate the estimated measurement, simply substitute the estimated quantities $\hat{\mathbf{r}}^f_{alt}$ and $\hat{\mathbf{k}}^f_{alt}$ in for $\mathbf{r}^f_{alt}$ and $\mathbf{k}^f_{alt}$ respectively.

5.3.3 Altimeter Measurement Deviation

The previous two sections define how to calculate the measurement vector magnitude and generate a true or estimated measurement. The final requirement for the estimation framework is a measurement deviation matrix, **H**, or the partial derivative matrix of the measurement error with respect to the state vector deviation (also known as the Jacobian matrix).

Solving for the deviation is similar to differentiating an equation. Since there are multiple variables in the measurement equation, this results in a partial differential equation. The deviation is defined by $\delta\{\cdot\}$, and it is used as an operator on terms of an equation. All rules of differentiation hold, such as the chain rule and quotient rule.

Measurement deviation is defined as the true measurement minus the estimated measurement. For the altimeter measurement model, this yields:

$$\delta y_k = \delta \rho + \delta b_{alt} + \eta_{alt} \tag{5.52}$$

where $\delta \rho = \rho \left(\mathbf{r}^f_{alt}, \mathbf{k}^f_{alt} \right) - \hat{\rho} \left(\hat{\mathbf{r}}^f_{alt}, \hat{\mathbf{k}}^f_{alt} \right)$ and $\delta b_{alt} = b_{alt} - \hat{b}_{alt}$.

Some mathematic rules simplify the calculation of this Jacobian matrix. First, the vector dot product can be written using the vector transpose, and the order does not change the result:

$$\mathbf{q} \cdot \mathbf{v} = \mathbf{q}^T \mathbf{v} = \mathbf{v}^T \mathbf{q}. \tag{5.53}$$

Next, the partial derivative of two vector's dot product is obtained by the chain rule. However, since the other vector is assumed independent of the first, one of the two terms resulting from the chain rule drops out:

$$\frac{\partial}{\partial \mathbf{v}} \left(\mathbf{q} \cdot \mathbf{v} \right) = \mathbf{q} \cdot \frac{\partial \mathbf{v}}{\partial \mathbf{v}} + \mathbf{v} \cdot \frac{\partial \mathbf{q}}{\partial \mathbf{v}} = \mathbf{q}^T \mathbf{I}_{3\times3} + \mathbf{v}^T \mathbf{0}_{3\times1} = \mathbf{q}^T \tag{5.54}$$

which extends to the deviation operator as:

$$\delta \left(\mathbf{q} \cdot \mathbf{v} \right) = \mathbf{q}^T \delta \mathbf{v} + \mathbf{v}^T \delta \mathbf{q}. \tag{5.55}$$

Finally, $\mathbf{S}$ is a diagonal matrix, so the transpose operator does nothing to change it. Also, taking the transpose of quantities that are multiplied reverses the order and transposes each quantity, i.e. $(ab)^T = b^T a^T$. When $\mathbf{S}$ premultiplies into a vector, and that quantity is part of a dot product, the result simplifies to:

$$(\mathbf{Sq}) \cdot (\mathbf{Sv}) = (\mathbf{Sq})^T (\mathbf{Sv}) = \mathbf{q}^T \mathbf{SSv}. \tag{5.56}$$

Returning to Eq. 5.46 and substituting in Eqs. 5.34 and 5.35 brings the equation for ρ into terms of the state vector only:

$$\begin{aligned}
&\rho^2 \left(\mathbf{S}\mathbf{T}_i^f \mathbf{T}_b^i \mathbf{T}_{alt}^b \mathbf{k}_{alt}^{alt}\right) \cdot \left(\mathbf{S}\mathbf{T}_i^f \mathbf{T}_b^i \mathbf{T}_{alt}^b \mathbf{k}_{alt}^{alt}\right) \\
&+ 2\rho \left(\mathbf{S}\mathbf{T}_i^f \mathbf{r}_{imu}^i + \mathbf{S}\mathbf{T}_i^f \mathbf{T}_b^i \mathbf{r}_{alt/imu}^b\right) \cdot \left(\mathbf{S}\mathbf{T}_i^f \mathbf{T}_b^i \mathbf{T}_{alt}^b \mathbf{k}_{alt}^{alt}\right) \\
&+ \left(\mathbf{S}\mathbf{T}_i^f \mathbf{r}_{imu}^i + \mathbf{S}\mathbf{T}_i^f \mathbf{T}_b^i \mathbf{r}_{alt/imu}^b\right) \cdot \left(\mathbf{S}\mathbf{T}_i^f \mathbf{r}_{imu}^i + \mathbf{S}\mathbf{T}_i^f \mathbf{T}_b^i \mathbf{r}_{alt/imu}^b\right) = 1
\end{aligned} \tag{5.57}$$

which expands and rearranges to:

$$\begin{aligned}
&\rho^2 \left(\mathbf{T}_i^f \mathbf{T}_b^i \mathbf{T}_{alt}^b \mathbf{k}_{alt}^{alt}\right)^T \mathbf{SS} \left(\mathbf{T}_i^f \mathbf{T}_b^i \mathbf{T}_{alt}^b \mathbf{k}_{alt}^{alt}\right) \\
&+ 2\rho \left(\mathbf{T}_i^f \mathbf{r}_{imu}^i\right)^T \mathbf{SS} \left(\mathbf{T}_i^f \mathbf{T}_b^i \mathbf{T}_{alt}^b \mathbf{k}_{alt}^{alt}\right) \\
&+ 2\rho \left(\mathbf{T}_i^f \mathbf{T}_b^i \mathbf{r}_{alt/imu}^b\right)^T \mathbf{SS} \left(\mathbf{T}_i^f \mathbf{T}_b^i \mathbf{T}_{alt}^b \mathbf{k}_{alt}^{alt}\right) \\
&+ \left(\mathbf{T}_i^f \mathbf{r}_{imu}^i\right)^T \mathbf{SS} \left(\mathbf{T}_i^f \mathbf{r}_{imu}^i\right) \\
&+ 2 \left(\mathbf{T}_i^f \mathbf{r}_{imu}^i\right)^T \mathbf{SS} \left(\mathbf{T}_i^f \mathbf{T}_b^i \mathbf{r}_{alt/imu}^b\right) \\
&+ \left(\mathbf{T}_i^f \mathbf{T}_b^i \mathbf{r}_{alt/imu}^b\right)^T \mathbf{SS} \left(\mathbf{T}_i^f \mathbf{T}_b^i \mathbf{r}_{alt/imu}^b\right) = 1.
\end{aligned} \tag{5.58}$$

Taking the deviation of Eq. 5.58 is done under the fact that $\delta \mathbf{T}_b^i = \hat{\mathbf{T}}_b^i [\delta \boldsymbol{\alpha} \times]$, and $\delta \mathbf{r}_{imu}^i$ and $\delta \rho$ are also defined as $\rho - \hat{\rho}$ and $\mathbf{r}_{imu}^i - \hat{\mathbf{r}}_{imu}^i$, respectively. After taking the deviation with extensive use of the rules outlined in Eqs. 5.53, 5.55, and

5.56, and substituting the following terms:

$$
\begin{aligned}
a_0 &= \hat{\rho} \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \mathbf{T}_{alt}^b \mathbf{k}_{alt}^{alt} \right)^T \mathbf{SS} \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \mathbf{T}_{alt}^b \mathbf{k}_{alt}^{alt} \right) \\
a_1 &= \left(\mathbf{T}_i^f \hat{\mathbf{r}}_{imu}^i \right)^T \mathbf{SS} \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \mathbf{T}_{alt}^b \mathbf{k}_{alt}^{alt} \right) \\
a_2 &= \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \mathbf{r}_{alt/imu}^b \right)^T \mathbf{SS} \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \mathbf{T}_{alt}^b \mathbf{k}_{alt}^{alt} \right) \\
b_0 &= \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \mathbf{T}_{alt}^b \mathbf{k}_{alt}^{alt} \right)^T \mathbf{SS} \left(\mathbf{T}_i^f \right) \\
b_1 &= \left(\mathbf{T}_i^f \hat{\mathbf{r}}_{imu}^i \right)^T \mathbf{SS} \left(\mathbf{T}_i^f \right) \\
b_2 &= \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \mathbf{r}_{alt/imu}^b \right)^T \mathbf{SS} \left(\mathbf{T}_i^f \right) \\
c_0 &= -\hat{\rho}^2 \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \mathbf{T}_{alt}^b \mathbf{k}_{alt}^{alt} \right)^T \mathbf{SS} \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \mathbf{T}_{alt}^b \left[\mathbf{k}_{alt}^{alt} \times \right] \right) \\
c_1 &= -\hat{\rho} \left(\mathbf{T}_i^f \hat{\mathbf{r}}_{imu}^i \right)^T \mathbf{SS} \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \mathbf{T}_{alt}^b \left[\mathbf{k}_{alt}^{alt} \times \right] \right) \\
c_2 &= -\hat{\rho} \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \mathbf{r}_{alt/imu}^b \right)^T \mathbf{SS} \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \mathbf{T}_{alt}^b \left[\mathbf{k}_{alt}^{alt} \times \right] \right) \\
c_3 &= -\hat{\rho} \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \mathbf{T}_{alt}^b \mathbf{k}_{alt}^{alt} \right)^T \mathbf{SS} \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \left[\mathbf{r}_{alt/imu}^b \times \right] \right) \\
c_4 &= -\left(\mathbf{T}_i^f \hat{\mathbf{r}}_{imu}^i \right)^T \mathbf{SS} \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \left[\mathbf{r}_{alt/imu}^b \times \right] \right) \\
c_5 &= -\left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \mathbf{r}_{alt/imu}^b \right)^T \mathbf{SS} \left(\mathbf{T}_i^f \hat{\mathbf{T}}_b^i \left[\mathbf{r}_{alt/imu}^b \times \right] \right),
\end{aligned}
\tag{5.59}
$$

the result yields an equation in $\delta\rho$, $\delta\mathbf{r}_{imu}^i$, and $\delta\boldsymbol{\alpha}$:

$$
(a_0 + a_1 + a_2)\,\delta\rho + (b_0 + b_1 + b_2)\,\delta\mathbf{r}_{imu}^i + (c_0 + c_1 + c_2 + c_3 + c_4 + c_5)\,\delta\boldsymbol{\alpha} = 0. \tag{5.60}
$$

Solving for $\delta\rho$ yields:

$$
\delta\rho = \frac{b_0 + b_1 + b_2}{a_0 + a_1 + a_2}\delta\mathbf{r}_{imu}^i + \frac{c_0 + c_1 + c_2 + c_3 + c_4 + c_5}{a_0 + a_1 + a_2}\delta\boldsymbol{\alpha}. \tag{5.61}
$$

Finally, substituting Eq. 5.61 into Eq. 5.52 yields the final altimeter measurement deviation (or Jacobian) matrix:

$$\delta y_k = \frac{b_0 + b_1 + b_2}{a_0 + a_1 + a_2}\delta \mathbf{r}^i_{imu} + \frac{c_0 + c_1 + c_2 + c_3 + c_4 + c_5}{a_0 + a_1 + a_2}\delta \boldsymbol{\alpha} + \delta b_{alt} + \eta_{alt}. \tag{5.62}$$

There are scenarios where this measurement model will yield a measurement even if the hardware sensor does not. Specific for an altimeter, there are minimum and maximum range constraints. A validity check to ensure beam intersection with the planetary body is also required. These must be implemented in the sensor modeling code to return an invalid measurement when appropriate.

5.4 Velocimeter

Light Detection and Ranging (LIDAR) velocimetry provides a relative velocity measurement of the velocimeter sensor relative to the planetary body surface. The LIDAR velocimeter may have multiple sensing beams, resulting in multiple individual measurements. These are combined into a single measurement vector of the relative velocity of the sensor with respect to the ground.

5.4.1 Velocimeter Measurement Model

The relative velocity can be easily computed using the known information of inertial velocity of the sensor and planetary angular velocity. The true relative velocity measurement is defined in the velocimeter frame of reference as:

$$y_k = \mathbf{T}_b^{vel}\mathbf{T}_i^b\left[\mathbf{v}_{vel}^i - \boldsymbol{\omega}_{p/i}^i \times \mathbf{r}_{vel}^i\right] + \mathbf{b}_{vel}^{vel} + \boldsymbol{\eta}_{vel}^{vel} \tag{5.63}$$

where

$$\mathbf{v}_{vel}^i = \mathbf{v}_{imu}^i + \mathbf{T}_b^i\left(\boldsymbol{\omega}_{b/i}^b \times \mathbf{r}_{vel/imu}^b\right) \tag{5.64}$$

and

$$\mathbf{r}_{vel}^i = \mathbf{r}_{imu}^i + \mathbf{T}_b^i\mathbf{r}_{vel/imu}^b. \tag{5.65}$$

Recall Eq. 5.32 that defines $\boldsymbol{\omega}_{b/i}^b$.

The estimated measurement is then:

$$\hat{y}_k = \mathbf{T}_b^{vel}\hat{\mathbf{T}}_i^b\left[\hat{\mathbf{v}}_{vel}^i - \boldsymbol{\omega}_{p/i}^i \times \hat{\mathbf{r}}_{vel}^i\right] + \hat{\mathbf{b}}_{vel}^{vel} \tag{5.66}$$

where

$$\hat{\mathbf{v}}_{vel}^i = \hat{\mathbf{v}}_{imu}^i + \hat{\mathbf{T}}_b^i\left(\hat{\boldsymbol{\omega}}_{b/i}^b \times \mathbf{r}_{vel/imu}^b\right) \tag{5.67}$$

and

$$\hat{\mathbf{r}}_{vel}^i = \hat{\mathbf{r}}_{imu}^i + \hat{\mathbf{T}}_b^i\mathbf{r}_{vel/imu}^b. \tag{5.68}$$

The rotation transformation from the spacecraft body frame to the velocimeter frame $\mathbf{T}_b^{vel}$, the rotation transformation from the planet-fixed frame to the inertial frame $\mathbf{T}_f^i$, the position vector from the IMU to the velocimeter $\mathbf{r}_{vel/imu}^b$, and the angular velocity of the planet in the inertial frame $\boldsymbol{\omega}_{p/i}^f$ are assumed well-known.

5.4.2 Velocimeter Measurement Deviation

The previous section defines how to calculate the velocimeter measurement, or the velocity of the velocimeter sensor relative to the planetary body surface. To complete the modeling for integration into the Extended Kalman Filter framework, the measurement deviation matrix, **H**, also known as the Jacobian matrix, has to be calculated.

The velocimeter measurement deviation is found by differencing the true and estimated measurement equations. This is done using a few definitions:

$$\begin{aligned}
\mathbf{T}_i^b &= \left(\mathbf{I}_{3\times 3} - \delta\boldsymbol{\alpha}\times\right)\hat{\mathbf{T}}_i^b \\
\mathbf{T}_b^i &= \hat{\mathbf{T}}_b^i\left(\mathbf{I}_{3\times 3} + \delta\boldsymbol{\alpha}\times\right) \\
\mathbf{v}_{vel}^i &= \hat{\mathbf{v}}_{vel}^i + \delta\mathbf{v}_{vel}^i \\
\mathbf{r}_{vel}^i &= \hat{\mathbf{r}}_{vel}^i + \delta\mathbf{r}_{vel}^i
\end{aligned} \tag{5.69}$$

Then, the measurement deviation, to first order, becomes:

$$\begin{aligned}
\delta y_k &= y_k - \hat{y}_k \\
&= \mathbf{T}_b^{vel}\hat{\mathbf{T}}_i^b\delta\mathbf{v}_{vel}^i - \mathbf{T}_b^{vel}\hat{\mathbf{T}}_i^b\boldsymbol{\omega}_{p/i}^i \times \delta\mathbf{r}_{vel}^i - \mathbf{T}_b^{vel}\left[\delta\boldsymbol{\alpha}\times\right]\hat{\mathbf{T}}_i^b\hat{\mathbf{v}}_{vel}^i \\
&\quad + \mathbf{T}_b^{vel}\left[\delta\boldsymbol{\alpha}\times\right]\hat{\mathbf{T}}_i^b\left(\boldsymbol{\omega}_{p/i}^i \times \hat{\mathbf{r}}_{vel}^i\right) + \delta\mathbf{b}_{vel}^{vel} + \boldsymbol{\eta}_{vel}^{vel}.
\end{aligned} \tag{5.70}$$

However, this equation contains variables that are not a part of the state vector, and so must be changed accordingly. Returning to Eqs. 5.64, 5.67, and 5.69, the velocity terms are differenced to yield the velocimeter velocity deviation in terms of state variables:

$$\begin{aligned}\delta\mathbf{v}^i_{vel} &= \mathbf{v}^i_{vel} - \hat{\mathbf{v}}^i_{vel} \\ &= \delta\mathbf{v}^i_{imu} - \hat{\mathbf{T}}^i_b \left[\mathbf{r}^b_{vel/imu}\times\right]\delta\boldsymbol{\omega}^b_{b/i} - \hat{\mathbf{T}}^i_b\left[\left(\hat{\boldsymbol{\omega}}^b_{b/i}\times\mathbf{r}^b_{vel/imu}\right)\times\right]\delta\boldsymbol{\alpha}.\end{aligned} \tag{5.71}$$

where Eq. 4.48 defines $\delta\boldsymbol{\omega}^b_{b/i}$.

Examining Eqs. 5.65, 5.68, and 5.69, the position terms are differenced to yield the velocimeter position deviation:

$$\begin{aligned}\delta\mathbf{r}^i_{vel} &= \mathbf{r}^i_{vel} - \hat{\mathbf{r}}^i_{vel} \\ &= \delta\mathbf{r}^i_{imu} - \hat{\mathbf{T}}^i_b\left[\mathbf{r}^b_{vel/imu}\times\right]\delta\boldsymbol{\alpha}.\end{aligned} \tag{5.72}$$

These expansions allow Eq. 5.70 to be written in terms of the state vector only. After substituting and simplifying, the final measurement deviation equation is:

$$\begin{aligned}\delta y_k =& - \mathbf{T}^{vel}_b\hat{\mathbf{T}}^b_i\left[\boldsymbol{\omega}^i_{p/i}\times\right]\delta\mathbf{r}^i_{imu} + \left[\mathbf{T}^{vel}_b\hat{\mathbf{T}}^b_i\right]\delta\mathbf{v}^i_{imu} + \mathbf{T}^{vel}_b\left[\mathbf{r}^b_{vel/imu}\times\right]\left[\boldsymbol{\omega}^b_{b/i,meas}\times\right]\delta\mathbf{m}^b_g \\ &+ \mathbf{T}^{vel}_b\left[\mathbf{r}^b_{vel/imu}\times\right]\left[\boldsymbol{\omega}^b_{b/i,meas}\mid\times\mid\right]\delta\mathbf{n}^b_g + \mathbf{T}^{vel}_b\left[\mathbf{r}^b_{vel/imu}\times\right]\left[\boldsymbol{\omega}^b_{b/i,meas}\diagdown\right]\delta\mathbf{s}^b_g \\ &+ \mathbf{T}^{vel}_b\left[\mathbf{r}^b_{vel/imu}\times\right]\delta\mathbf{b}^b_g + \mathbf{T}^{vel}_b\left[\mathbf{r}^b_{vel/imu}\times\right]\delta\boldsymbol{\eta}^b_g + \mathbf{T}^{vel}_b\hat{\mathbf{T}}^b_i\left[\boldsymbol{\omega}^i_{p/i}\times\right]\hat{\mathbf{T}}^i_b\left[\mathbf{r}^b_{vel/imu}\times\right]\delta\boldsymbol{\alpha}^b \\ &- \mathbf{T}^{vel}_b\left[\left(\hat{\boldsymbol{\omega}}^b_{b/i}\times\mathbf{r}^b_{vel/imu}\right)\times\right]\delta\boldsymbol{\alpha}^b + \mathbf{T}^{vel}_b\left[\hat{\mathbf{T}}^b_i\left[\hat{\mathbf{v}}^i_{vel} - \left(\boldsymbol{\omega}^i_{p/i}\times\hat{\mathbf{r}}^i_{vel}\right)\right]\times\right]\delta\boldsymbol{\alpha}^b + \delta\mathbf{b}^{vel}_{vel} + \boldsymbol{\eta}^{vel}_{vel}.\end{aligned} \tag{5.73}$$

This concludes the velocimeter model derivation. Note that this model may return a numerical result in spite of a meaningful measurement not existing due to sensor limitations or the lack of beam intersection with the planet surface. Proper precautions must be included in the flight software to handle these edge cases gracefully.

5.5 Terrain/Hazard Relative Navigation Sensor

The Terrain/Hazard Relative Navigation (THRN) sensor is a lidar-enabled camera system that serves two functions: first, it creates a digital elevation map (DEM) of the potential landing area, recognizing safe landing surfaces based on a maximum slope criteria and the lack of hazards and also selecting and computing intended landing position (ILP)-relative positions of features, such as rocks or craters, to track; then, it continuously tracks the selected features during the landing approach and returns a THRN-sensor-to-feature vector measurement. Figure 5.1 details the two measured vector types as found during DEM creation: the vector from the sensor to the ILP, and the vectors from the ILP to the detected features. Since this DEM-creation is done based on an estimated position that includes map-tie errors, the true inertial position of the ILP and tracked features is not known; this results in the $\mathbf{b}_{ill}$ term for map-tie error, which is estimated as part of the state vector. The technology capable of this type of sensing is possible the topic of current research[25][26][27][28][29].

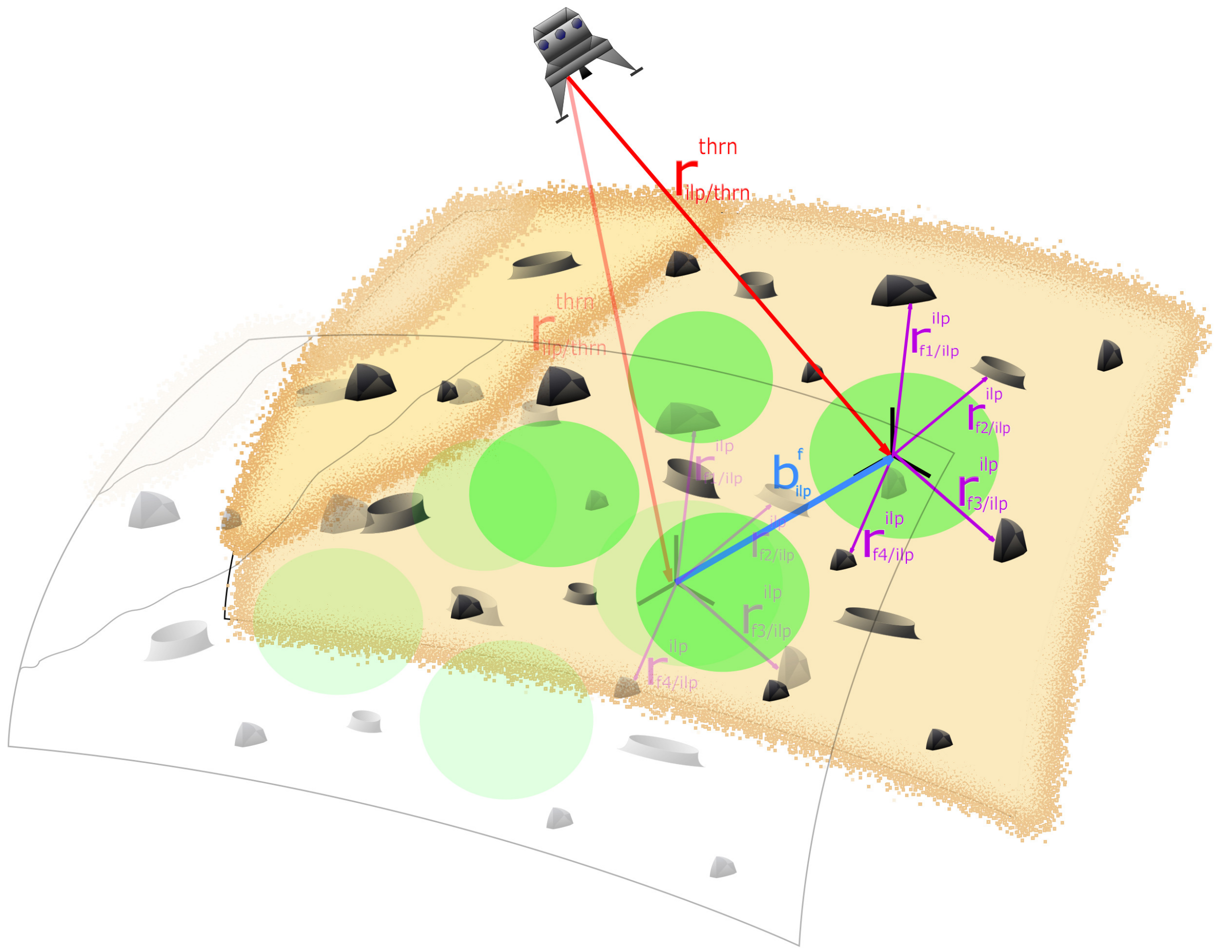

Figure 5.1: THRN sensor measurement vectors as defined during DEM creation

The sensor is called THRN because it can function as both a terrain-relative sensor and a hazard-relative sensor. THRN acts as a terrain-relative and hazard-relative sensor when it is given a predefined map of the landing area. In this sense, an ILP can be chosen to be near an asset or feature that is of interest. This type of map-predefinition is inherently flawed by map-tie errors, which are estimated by the $\mathbf{b}_{ilp}^{f}$ state variable. Note that in Figure 5.1, the $\mathbf{b}_{ilp}^{f}$ vector is shown to specify the map tie error between the true feature positions in the true ILP frame, and the estimated feature positions as predefined by some map of the surface. In reality, the true position of a predefined ILP cannot be known, which is why this $\mathbf{b}_{ilp}^{f}$ term is estimated. This sensor is considered to do both terrain-relative navigation (TRN) and hazard-relative navigation (HRN) because the sensor fuses the tracked feature positions, which are relative measurements, to both reduce position estimate error in the inertial frame and map-tie error in the planet-fixed frame.

There are two parts to consider for the THRN model: the truth model and the measurement model. The truth model is used in simulation to create the sensor measurement, and the measurement model is used by the filter to calculate an expected measurement so it can then compute the measurement residual. Once defined, the truth and measurement models are differenced and, using Taylor-series small angle approximations, the partial derivative model is created. Finally, from the measurement model the measurement mapping matrix is created.

The THRN sensor outputs two different measurements; the feature position with respect to the ILP frame center in the ILP frame, $\hat{\mathbf{r}}_{f/ilp}^{ilp}$, and the feature position with respect to the THRN sensor in the THRN sensor frame, $\hat{\mathbf{r}}^{thrn}_{f/thrn}$. Both are estimated vectors because they require knowledge of the THRN sensor position, which is only known as a function of the estimated states. The feature position vectors are assumed to be fixed in the ILP frame, and thus once calculated they are used to create measurement estimates for hazard-relative navigation.

Part of the inputs to the navigation system is an intended landing point, or ILP. This is the landing target, and could be a point of interest for science or rendezvous for example. The ILP position is based on map knowledge of the target, and thus is subject to map-tie error. The ILP position relative to other prominent features is assumed well-known, but the inertial position of these features may be skewed. Since all navigation is done in the inertial frame, it is important to drive these map-tie errors to zero to enable precise landing in a safe zone. This is done by use of the $\mathbf{b}^{f}_{ilp}$ term.

To illustrate the THRN sensor measurements, Figure 5.2 shows both the true spacecraft trajectory, the estimated trajectory, and the resulting measurement differences incorporating $\mathbf{b}^{f}_{ilp}$, $\hat{\mathbf{r}}^{ilp}_{f/ilp}$, and $\hat{\mathbf{r}}^{thrn}_{f/thrn}$.

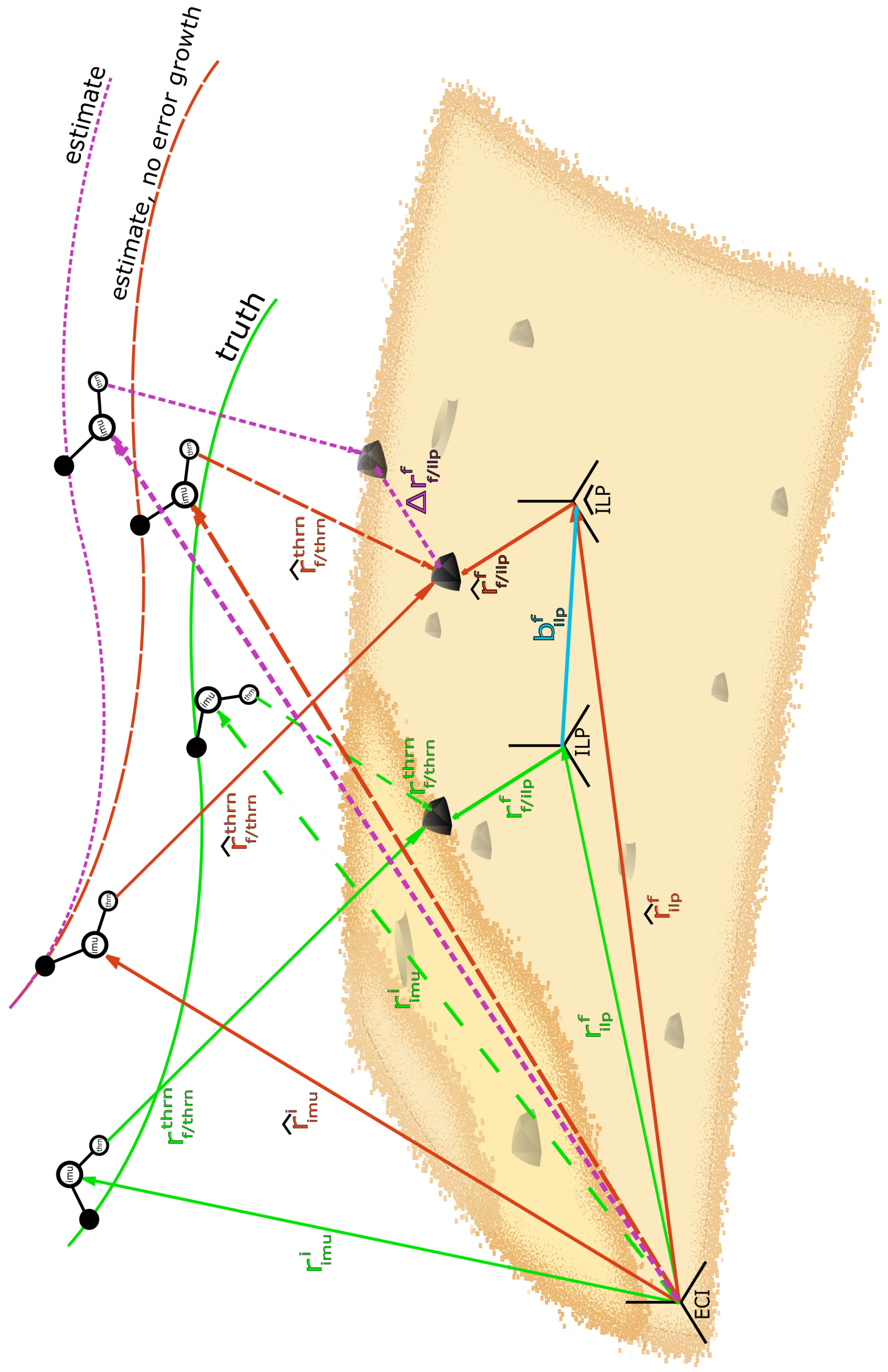

Figure 5.2: THRN overview illustration

5.5.1 THRN Measurement Model

The true THRN measurement is a vector from the sensor to the tracked feature in the THRN sensor frame and is a function of the ILP position, the THRN sensor position, and the feature position with respect to the ILP, which is defined by the THRN sensor during the DEM creation phase. The true THRN measurement model is then:

$$\mathbf{y}_k = \mathbf{r}^{thrn}_{f/thrn} = \mathbf{T}^{thrn}_b \mathbf{T}^b_i \mathbf{T}^i_f \mathbf{r}^f_{ilp} + \mathbf{T}^{thrn}_b \mathbf{T}^b_i \mathbf{T}^i_f \mathbf{T}^f_{ilp} \hat{\mathbf{r}}^{ilp}_{f/ilp} - \mathbf{T}^{thrn}_b \mathbf{T}^b_i \mathbf{r}^i_{thrn} + \mathbf{b}^{thrn}_{thrn} + \boldsymbol{\eta}^{thrn}_{thrn} \tag{5.74}$$

where the ILP position is the predefined ILP position estimate plus the true map-tie error bias:

$$\mathbf{r}^f_{ilp} = \mathbf{r}^f_{ilp,preload} + \mathbf{b}^f_{ilp} \tag{5.75}$$

and the THRN sensor position is the inertial position of the IMU plus the position of the THRN sensor relative to the IMU:

$$\mathbf{r}^i_{thrn} = \mathbf{r}^i_{imu} + \mathbf{T}^i_b \mathbf{r}^b_{thrn/imu}. \tag{5.76}$$

The estimated THRN measurement is:

$$\hat{\mathbf{y}}_k = \hat{\mathbf{r}}^{thrn}_{f/thrn} = \mathbf{T}^{thrn}_b \hat{\mathbf{T}}^b_i \mathbf{T}^i_f \hat{\mathbf{r}}^f_{ilp} + \mathbf{T}^{thrn}_b \hat{\mathbf{T}}^b_i \mathbf{T}^i_f \mathbf{T}^f_{ilp} \hat{\mathbf{r}}^{ilp}_{f/ilp} - \mathbf{T}^{thrn}_b \hat{\mathbf{T}}^b_i \hat{\mathbf{r}}^i_{thrn} + \hat{\mathbf{b}}^{thrn}_{thrn} \tag{5.77}$$

where the estimated ILP position is the predefined ILP position plus the estimated map-tie error bias:

$$\hat{\mathbf{r}}^f_{ilp} = \mathbf{r}^f_{ilp,preload} + \hat{\mathbf{b}}^f_{ilp} \tag{5.78}$$

and

$$\hat{\mathbf{r}}^i_{thrn} = \hat{\mathbf{r}}^i_{imu} + \hat{\mathbf{T}}^i_b \mathbf{r}^b_{thrn/imu}. \tag{5.79}$$

5.5.2 THRN Measurement Deviation

The measurement deviation is found by differencing the true and estimated measurements, and is done using the following definitions for state variable deviations:

$$\begin{aligned}
\mathbf{T}^b_i &= (\mathbf{I}_{3\times 3} + \delta\boldsymbol{\alpha}\times)\,\hat{\mathbf{T}}^b_i \\
\mathbf{T}^i_b &= \hat{\mathbf{T}}^i_b\,(\mathbf{I}_{3\times 3} + \delta\boldsymbol{\alpha}\times) \\
\mathbf{r}^i_{thrn} &= \hat{\mathbf{r}}^i_{thrn} + \delta\mathbf{r}^i_{thrn} \\
\mathbf{r}^f_{ilp} &= \hat{\mathbf{r}}^f_{ilp} + \delta\mathbf{r}^f_{ilp}
\end{aligned} \tag{5.80}$$

The THRN measurement deviation is:

$$\begin{aligned}
\delta\mathbf{y}_k = {} & \mathbf{T}^{thrn}_b \hat{\mathbf{T}}^b_i \mathbf{T}^i_f \delta\mathbf{r}^f_{ilp} - \mathbf{T}^{thrn}_b \hat{\mathbf{T}}^b_i \delta\mathbf{r}^i_{thrn} - \mathbf{T}^{thrn}_b \left[\hat{\mathbf{T}}^b_i \mathbf{T}^i_f \hat{\mathbf{r}}^f_{ilp}\times\right]\delta\boldsymbol{\alpha} \\
& - \mathbf{T}^{thrn}_b \left[\hat{\mathbf{T}}^b_i \mathbf{T}^i_f \mathbf{T}^f_{ilp} \hat{\mathbf{r}}^{ilp}_{f/ilp}\times\right]\delta\boldsymbol{\alpha} + \mathbf{T}^{thrn}_b \left[\hat{\mathbf{T}}^b_i \hat{\mathbf{r}}^i_{thrn}\times\right]\delta\boldsymbol{\alpha} + \delta\mathbf{b}^{thrn}_{thrn} + \boldsymbol{\eta}^{thrn}_{thrn}.
\end{aligned} \tag{5.81}$$

This equation contains deviation terms that are not state variable deviations, so further substitutions are required. Eqs. 5.75 and 5.78, when differenced, yield:

$$\begin{aligned}\delta\mathbf{r}^f_{ilp} &= \mathbf{r}^f_{ilp} - \hat{\mathbf{r}}^f_{ilp} = \mathbf{r}^f_{ilp,preload} - \mathbf{r}^f_{ilp,preload} + \mathbf{b}^f_{ilp} - \hat{\mathbf{b}}^f_{ilp} \\ \delta\mathbf{r}^f_{ilp} &= \delta\mathbf{b}^f_{ilp},\end{aligned} \tag{5.82}$$

which is a defined in the state vector, and then Eqs. 5.76 and 5.79 are differenced to provide:

$$\delta\mathbf{r}^i_{thrn} = \mathbf{r}^i_{thrn} - \hat{\mathbf{r}}^i_{thrn} = \mathbf{r}^i_{imu} - \hat{\mathbf{r}}^i_{imu} + \mathbf{T}^i_b\mathbf{r}^b_{thrn/imu} - \hat{\mathbf{T}}^i_b\mathbf{r}^b_{thrn/imu} \tag{5.83}$$

which, when simplified using Eq. 5.80 yields:

$$\delta\mathbf{r}^i_{thrn} = \delta\mathbf{r}^i_{imu} - \hat{\mathbf{T}}^i_b\left[\mathbf{r}^b_{thrn/imu}\times\right]\delta\boldsymbol{\alpha} \tag{5.84}$$

Making substitutions of Eqs. 5.82 and 5.84 into Eq. 5.81 yields the final THRN measurement deviation equation in terms of the state variable deviations:

$$\begin{aligned}\delta\mathbf{y}_k =& \mathbf{T}^{thrn}_b\left(\left[\mathbf{r}^b_{thrn/imu}\times\right] + \left[\hat{\mathbf{T}}^b_i\hat{\mathbf{r}}^i_{thrn}\times\right] - \left[\hat{\mathbf{T}}^b_i\mathbf{T}^i_f\hat{\mathbf{r}}^f_{ilp}\times\right] - \left[\hat{\mathbf{T}}^b_i\mathbf{T}^i_f\mathbf{T}^f_{ilp}\hat{\mathbf{r}}^{ilp}_{f/ilp}\times\right]\right)\delta\boldsymbol{\alpha} \\ &- \mathbf{T}^{thrn}_b\hat{\mathbf{T}}^b_i\delta\mathbf{r}^i_{imu} + \mathbf{T}^{thrn}_b\hat{\mathbf{T}}^b_i\mathbf{T}^i_f\delta\mathbf{b}^f_{ilp} + \delta\mathbf{b}^{thrn}_{thrn} + \boldsymbol{\eta}^{thrn}_{thrn}.\end{aligned} \tag{5.85}$$

This finalizes the THRN measurement derivation.

This derivation assumes that the THRN sensor properly tracks a selected feature, and is smart enough to redefine the $\hat{\mathbf{r}}^{ilp}_{f/ilp}$ vector if a new feature is chosen mid-descent. Also, proper methods must be programmed into the flight software to handle cases where the sensor cannot return a valid measurement.

5.6 Star Camera

A star camera sensor uses an imaging sensor and a preloaded map of star positions to calculate a pointing vector from the sensor to a star in the star camera frame. This pointing vector is used to calculate the sensor's attitude relative to the star map. Various star tracking methods exist for multiple imaging sensor types[30][31]. For this analysis, it is assumed that the sensor sends a measured attitude quaternion to the filter.

5.6.1 Star Camera Measurement Model

The star camera measurement is a direction vector to the star in the star camera frame:

$$\mathbf{u}^{sc}_{s/sc} = \mathbf{T}^{sc}_{b}\mathbf{T}^{b}_{i}\mathbf{T}^{i}_{sr}\mathbf{u}^{sr}_{s/sc} \tag{5.86}$$

where $\mathbf{u}^{sr}_{s/sc}$ is the true vector direction from the star camera to the measured star, in the star-reference (SR) frame. The SR frame is a well-known reference frame in which relative star positions are mapped. The rotation transformation from the SR frame to the inertial frame is also well-known.

The measurement can also be written as a the rotation transformation matrix from the SR frame to the star camera frame since :

$$\mathbf{T}^{sc}_{sr} = \mathbf{T}^{sc}_{b}\mathbf{T}^{b}_{i}\mathbf{T}^{i}_{sr} \tag{5.87}$$

which can further be written using rotation quaternions, required for updating the spacecraft attitude quaternion:

$$\bar{\mathbf{q}}_{sr}^{sc} = \bar{\mathbf{q}}_{b}^{sc} \otimes \bar{\mathbf{q}}_{i}^{b} \otimes \bar{\mathbf{q}}_{sr}^{i}. \tag{5.88}$$

The true measurement may be corrupted by random bias and measurement noise, parameterized by:

$$\boldsymbol{\theta}_{sc} = \mathbf{b}_{sc}^{sc} + \boldsymbol{\eta}_{sc}^{sc} \tag{5.89}$$

This is written as the bias quaternion:

$$\bar{\mathbf{q}}_{b,\eta}^{sc} = \begin{bmatrix} \sin\left(\frac{\theta_{sc}}{2}\right)\frac{\boldsymbol{\theta}_{sc}}{\theta_{sc}} \\ \\ \cos\left(\frac{\theta_{sc}}{2}\right) \end{bmatrix} \tag{5.90}$$

where

$$\theta_{sc} = \|\boldsymbol{\theta}_{sc}\| \tag{5.91}$$

Thus, the true quaternion measurement from the star camera is:

$$\bar{\mathbf{q}}_{sr}^{sc} = \bar{\mathbf{q}}_{b,\eta}^{sc} \otimes \bar{\mathbf{q}}_{b}^{sc} \otimes \bar{\mathbf{q}}_{i}^{b} \otimes \bar{\mathbf{q}}_{sr}^{i} \tag{5.92}$$

and the estimated measurement is:

$$\hat{\bar{\mathbf{q}}}_{sr}^{sc} = \hat{\bar{\mathbf{q}}}_{b,\eta}^{sc} \otimes \bar{\mathbf{q}}_{b}^{sc} \otimes \hat{\bar{\mathbf{q}}}_{i}^{b} \otimes \bar{\mathbf{q}}_{sr}^{i} \tag{5.93}$$

where:

$$\hat{\bar{\mathbf{q}}}^{sc}_{b,\eta} = \begin{bmatrix} \sin\left(\frac{\hat{\theta}_{sc}}{2}\right)\frac{\hat{\boldsymbol{\theta}}_{sc}}{\hat{\theta}_{sc}} \\ \cos\left(\frac{\hat{\theta}_{sc}}{2}\right) \end{bmatrix} \tag{5.94}$$

$$\hat{\boldsymbol{\theta}}_{sc} = \hat{\mathbf{b}}^{sc}_{sc} \tag{5.95}$$

and

$$\hat{\theta}_{sc} = \|\hat{\boldsymbol{\theta}}_{sc}\| \tag{5.96}$$

5.6.2 Star Camera Measurement Deviation

The star camera measurement deviation is unlike the deviations for the other sensors because quaternions cannot be differenced. The quaternion deviation is defined by quaternion multiplication:

$$\delta\bar{\mathbf{q}} = \bar{\mathbf{q}} \otimes [\bar{\mathbf{q}}]^{-1} \tag{5.97}$$

Thus,

$$\delta\bar{\mathbf{q}}^{sc}_{sr} = \bar{\mathbf{q}}^{sc}_{sr} \otimes \left[\hat{\bar{\mathbf{q}}}^{sc}_{sr}\right]^{-1} \tag{5.98}$$

and the quaternion inverse is defined[19]:

$$[\bar{\mathbf{q}}]^{-1} = \frac{\begin{bmatrix} -\mathbf{q} \\ q \end{bmatrix}}{\|\bar{\mathbf{q}}\|^2}. \tag{5.99}$$

Note that a quaternion multiplied by or into its inverse is the unit quaternion:

$$\bar{\mathbf{q}} \otimes [\bar{\mathbf{q}}]^{-1} = [\bar{\mathbf{q}}]^{-1} \otimes \bar{\mathbf{q}} = [0\,0\,0\,1]^T. \tag{5.100}$$

Thus, substituting Eqs. 5.92 and 5.93 into 5.97 yields:

$$\delta\bar{\mathbf{q}}_{sr}^{sc} = \bar{\mathbf{q}}_{b,\eta}^{sc} \otimes \bar{\mathbf{q}}_{b}^{sc} \otimes \bar{\mathbf{q}}_{i}^{b} \otimes \bar{\mathbf{q}}_{sr}^{i} \otimes \left[\bar{\mathbf{q}}_{sr}^{i}\right]^{-1} \otimes \left[\hat{\bar{\mathbf{q}}}_{i}^{b}\right]^{-1} \otimes \left[\bar{\mathbf{q}}_{b}^{sc}\right]^{-1} \otimes \left[\hat{\bar{\mathbf{q}}}_{b,\eta}^{sc}\right]^{-1}. \tag{5.101}$$

The above equation simplifies using Eq. 5.100 and the definition of the quaternion deviation in Eq. 5.97 to:

$$\delta\bar{\mathbf{q}}_{sr}^{sc} = \bar{\mathbf{q}}_{b,\eta}^{sc} \otimes \bar{\mathbf{q}}_{b}^{sc} \otimes \delta\bar{\mathbf{q}}_{i}^{b} \otimes \left[\bar{\mathbf{q}}_{b}^{sc}\right]^{-1} \otimes \left[\hat{\bar{\mathbf{q}}}_{b,\eta}^{sc}\right]^{-1}. \tag{5.102}$$

Then, the quaternion multiplication series $\bar{\mathbf{q}}_{b}^{sc} \otimes \delta\bar{\mathbf{q}}_{i}^{b} \otimes [\bar{\mathbf{q}}_{b}^{sc}]^{-1}$ can be simplified to the quaternion deviation with the vector part transformed by a rotation matrix as follows:

$$\bar{\mathbf{q}}_{b}^{sc} \otimes \delta\bar{\mathbf{q}}_{i}^{b} \otimes [\bar{\mathbf{q}}_{b}^{sc}]^{-1} = \begin{bmatrix} \mathbf{T}_{b}^{sc}\delta\mathbf{q}_{i}^{b} \\ \delta q_{i}^{b} \end{bmatrix} \tag{5.103}$$

and so the quaternion measurement deviation becomes:

$$\delta\bar{\mathbf{q}}_{sr}^{sc} = \bar{\mathbf{q}}_{b,\eta}^{sc} \otimes \begin{bmatrix} \mathbf{T}_{b}^{sc}\delta\mathbf{q}_{i}^{b} \\ \delta q_{i}^{b} \end{bmatrix} \otimes \left[\hat{\bar{\mathbf{q}}}_{b,\eta}^{sc}\right]^{-1}. \tag{5.104}$$

Next, the true bias quaternion $\bar{\mathbf{q}}_{b,\eta}^{sc}$ can be written in terms of the estimated bias quaternion and a deviation term:

$$\bar{\mathbf{q}}_{b,\eta}^{sc} = \delta\bar{\mathbf{q}}_{b,\eta}^{sc} \otimes \hat{\bar{\mathbf{q}}}_{b,\eta}^{sc} \tag{5.105}$$

which, when substituted into Eq. 5.104, lends to further simplification of the form found in Eq. 5.103, yielding:

$$\delta\bar{\mathbf{q}}^{sc}_{sr} = \delta\bar{\mathbf{q}}^{sc}_{b,\eta} \otimes \begin{bmatrix} \hat{\mathbf{T}}_{b,\eta}\mathbf{T}^{sc}_{b}\delta\mathbf{q}^{b}_{i} \\ \delta q^{b}_{i} \end{bmatrix} \tag{5.106}$$

Under a small angle assumption, the scalar of a quaternion deviation becomes unity and the vector portion is the small-angle vector:

$$\delta\bar{\mathbf{q}} = \begin{bmatrix} \delta\mathbf{q} \\ 1 \end{bmatrix} \tag{5.107}$$

Expanding Eq. 5.106 under the definition of qauternion multiplication, and applying the small angle approximation, yields:

$$\delta\bar{\mathbf{q}}^{sc}_{sr} = \begin{bmatrix} \delta\mathbf{q}^{sc}_{b,\eta} + \hat{\mathbf{T}}_{b,\eta}\mathbf{T}^{sc}_{b}\delta\mathbf{q}^{b}_{i} \\ 1 \end{bmatrix}. \tag{5.108}$$

As previously defined in Eq. 4.13, the small angle quaternion-vector becomes:

$$\delta\mathbf{q}^{sc}_{b,\eta} = \frac{1}{2}\delta\boldsymbol{\theta}_{sc} \tag{5.109}$$

and

$$\delta\mathbf{q}^{b}_{i} = \frac{1}{2}\delta\boldsymbol{\alpha}. \tag{5.110}$$

Recalling the definition of $\boldsymbol{\theta}_{sc} = \mathbf{b}^{sc}_{sc} + \boldsymbol{\eta}^{sc}_{sc}$ and $\hat{\boldsymbol{\theta}}_{sc} = \hat{\mathbf{b}}^{sc}_{sc}$, Eq. 5.109 further reduces to:

$$\delta\mathbf{q}_{b,\eta}^{sc} = \frac{1}{2}\delta\mathbf{b}_{sc}^{sc} + \frac{1}{2}\boldsymbol{\eta}_{sc}^{sc} \tag{5.111}$$

Finally, substituting Eqs. 5.110 and 5.111 into 5.108 and writing only the vector part of the quaternion deviation yields the final form of the star camera measurement deviation. Note that since all quantities on the right hand side of the equation are small angles, the left hand side is also small angles, which means that $\delta\mathbf{q}_{sr}^{sc}$ can be written as $\frac{1}{2}\delta\boldsymbol{\gamma}$. Substituting and simplifying yields:

$$\delta\boldsymbol{\gamma} = \hat{\mathbf{T}}_{b,\eta}\mathbf{T}_{b}^{sc}\delta\boldsymbol{\alpha} + \delta\mathbf{b}_{sc}^{sc} + \boldsymbol{\eta}_{sc}^{sc} \tag{5.112}$$

5.6.3 Star Camera Measurement Residual

The difference between two quaternions is determined through quaternion multiplication rather than subtraction, and thus the star camera measurement residual is computed by multiplying the measured quaternion and the inverse of the estimated measurement. Since the attitude quaternion is a unit quaternion, the quaternion inverse simplifies, yielding:

$$\bar{\mathbf{q}}_{residual} = \bar{\mathbf{q}}_{sr}^{sc} \otimes \left[\hat{\bar{\mathbf{q}}}_{sr}^{sc}\right]^{-1} = \bar{\mathbf{q}}_{sr}^{sc} \otimes \begin{bmatrix} -\hat{\mathbf{q}}_{sr}^{sc} \\ \\ \hat{q}_{sr}^{sc} \end{bmatrix}. \tag{5.113}$$

Assuming that the resulting quaternion represents a set of small-angle deviations between the measured and estimated quaternions, the vector part can be taken as the small-angle attitude deviation, defined in the filter as $\delta\boldsymbol{\alpha}$:

$$\bar{\mathbf{q}}_{residual} \approx \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\alpha} \\ 1 \end{bmatrix} \tag{5.114}$$

which implies:

$$\delta\boldsymbol{\alpha} = 2\left(\hat{q}_{sr}^{sc}\mathbf{q}_{sr}^{sc} + q_{sr}^{sc}\left(-\hat{\mathbf{q}}_{sr}^{sc}\right) + \left[\mathbf{q}_{sr}^{sc} \times \left(-\hat{\mathbf{q}}_{sr}^{sc}\right)\right]\right). \tag{5.115}$$

The residual value as computed above is combined additively to the most current state values for $\delta\boldsymbol{\alpha}$, which are then updated as per the Kalman Filter equations outlined in Section 2.2.

Finally, since the attitude quaternion cannot be updated additively, the post-update filter state values for $\delta\boldsymbol{\alpha}$ are applied to the estimated attitude quaternion $\bar{\mathbf{q}}_i^b$ multiplicatively (assuming that $\delta\boldsymbol{\alpha}$ can be represented as a small-angle quaternion), and the filter states for $\delta\boldsymbol{\alpha}$ are zeroed-out, as attitude deviation does not carry between time steps.

CHAPTER 6
Simulation Framework

To ensure that a filter design works as expected, and to aid in the tuning process, the filter is simulated for some nominal trajectory. This chapter explains the development of the simulation software, properties and capabilities of the filter simulation, and the simulated trajectory.

6.1 Simulation Overview

The simulation software is written entirely in MATLAB version 2014b[9]. This section aims to provide an understanding of the simulation framework, as the software was developed to prove the THRN sensor model's effectiveness as part of the Extended Kalman Filter navigation system. Functional descriptions of the different simulation modules are provided, as well as high-level flow diagrams to illustrate the information pathways in the simulation.

The main parts of the simulation framework are described in Table 6.1. Each module is made up of many functions that together make up the simulation. For example, the Initialization module preallocates memory based on the predefined time step size and length of time to simulate, which are set as inputs in the main simulation driver file to enable quick and easy adjustment. Most of the simulation is modular, and the user-changeable parameters are in only three files that feed all the pertinent information into the rest of the simulation. In this way, it is easy to turn a particular sensor on or off, enable or disable sensor measurement noise, and change the nominal trajectory that is simulated.

Table 6.1: Main simulation framework modules

Module	Description
Initialization	Setting of simulation parameters and flags, preallocation of memory, trajectory creation
Simulation	Looping through propagation and update phases for all time steps
Analysis	End-result data manipulation, error data computation, plotting, and report generation

The simulation can run in one of three modes defined in Table 6.2. Each individual mode is used to verify the sensor models and filter tuning in different ways. A separate simulation driver file is used for each mode since they all require special runtime configurations, but the underlying simulation module is unchanged for each of the modes. This ensures that the simulation, trajectory, and filter tuning parameters are constant throughout testing of the single-run, *Monte Carlo*, and error-budget modes. It is also important to note that the different simulation modes are performed at the same time step resolution to provide the most consistent data for post-processing and analysis.

Figure 6.1 shows the series of steps that each simulation goes through to complete a single run. Because there are different possible simulation modes as shown in Table 6.2, there are multiple paths available as the software goes through each module. Activation flags set at the beginning of each simulation run define what specific path is followed.

Table 6.2: Simulation testing and verification modes

Mode	Description
Single-run	A single flight is simulated and state error and measurement residual information analyzed
Monte Carlo	Hundreds of single-run simulations are done with different initial state errors, then aggregate error and covariance data is analyzed
Error-budget	A series of single-runs are done with all error sources turned off except the ones corresponding to the tested error group, then complete error data is put through sensitivity analysis

The "Simulation Configuration" step includes user input to activate the proper simulation flags, which select the activated sensors, noise sources, and simulation mode. Once the simulation is configured, the proper data structures and initial conditions are set based on the previously selected simulation flags. The choice between each of the three simulation modes defines particular simulation looping logic that is followed during the "Simulation Loop" phase. The Loop phase includes the logic to operate the EKF, mainly the propagation and state update equations. All data is accumulated during the Loop phase. Finally, "Post-Processing and Plotting" computes the required outputs which may include state error vectors and error budget tables. Any required figures are generated during this phase, and depending on a simulation flag, MATLAB generates a PDF report for easy viewing of the results.

Functions handle the input, processing, and output for each of the phases above. Each phase is further broken into different functions to complete the requisite tasks. The specific function names and tasks are not included, as the information

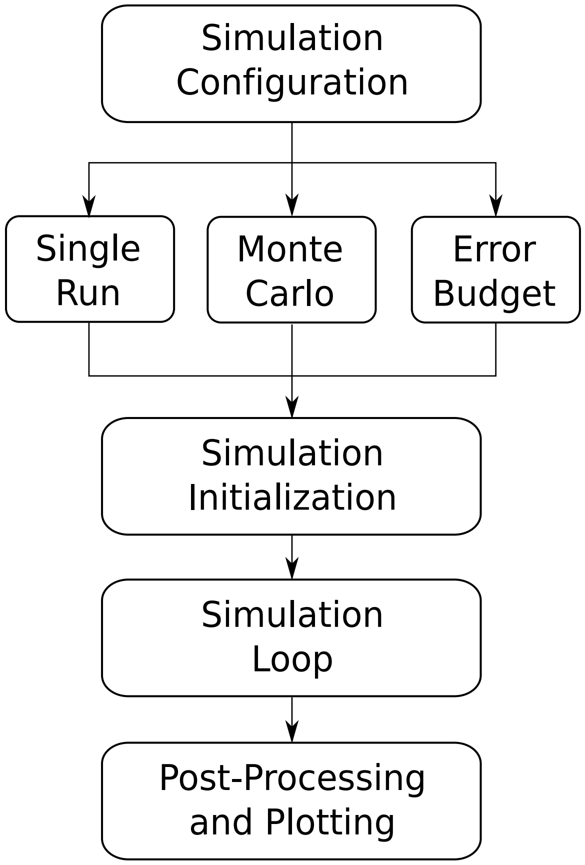

Figure 6.1: Software flow diagram

would replicate the derived system dynamics, sensor models, and Extended Kalman Filter equations found in previous chapters. However, the software is broken into logical chunks; for example, each sensor model is written as two functions: one for creating the true measured quantity, and one for creating the filter's estimate of the measurement. Although this results in some code redundancy, the inputs to each truth or estimate function differ, so the functions are separated for clarity.

The software, as created, can easily be included as part of a larger simulation that includes the controller and system blocks as seen in Figure 2.1. This would require writing software to implement a controller, as well as a system model that could be of varying complexity, including or excluding the following parts: a thrust model, variable vehicle CG position based on fuel loading, environmental effects such as wind and dust clouds, and system fault modeling, among others. As the focus of this thesis is to prove that the proposed THRN sensor model is effective in an EKF navigation system, the larger simulation is deemed unnecessary and left for future work.

A final note about using MATLAB for this simulation: beyond the well-known optimizations to be made to any MATLAB code (preallocation of memory and vectorization), the simulation time was reduced by more than fifty percent when all but the simulation driver file were converted into functions rather than scripts. Once optimized in this fashion, a single-run simulation would take approximately ten seconds, while each single-run in a *Monte Carlo* simulation (which utilized all eight 2.6GHz cores in the computer) took approximately five seconds. Compared to the total simulated time of 130 seconds, this is a 13x-26x speedup over real-time.

6.2 Simulation Parameters

The following simulation time parameters found in Table 6.3 were held constant throughout each simulation mode. Their variables, values, and descriptions are summarized.

Table 6.4 indicates time values for important trajectory-related events.

Table 6.3: Constant simulation parameters

Variable	Value	Description
t_0	0 (s)	Simulation start time
t_f	130 (s)	Simulation end time
$t_{0,thrn}$	70 (s)	THRN sensor activation time
$t_{f,thrn}$	100 (s)	THRN sensor deactivation time
dt_{prop}	0.02 (s)	Propagation time step size
dt_{kf}	0.2 (s)	Filter time step size

These event times were also held constant for each simulation mode, yielding the exact same true trajectory for each single-run, *Monte Carlo* single-run, or error-budget series single-run.

Table 6.4: Key trajectory timed events

Time	Event description
0 (s)	Simulation start
30 (s)	Liftoff to ascent
50 (s)	Begin commanded pitch-over
54 (s)	Pitch-over recovery for slant-range descent
60 (s)	Begin slant-range descent
110 (s)	Final descent preparation pitch-over recovery
115 (s)	Begin final descent
130 (s)	Final touchdown

Finally, the flight test is simulated on Earth, so the Earth-specific values for gravitational parameters, reference ellipsoid parameters, and angular velocity were used. Adapting these values for a different planet would be trivial.

6.3 Modeled Trajectory

The trajectory used to simulate a landing scenario is modeled after an ascent, slant-range descent, descent flight plan, where an initial altitude is achieved, lateral motion begins with gradual descent, and final descent occurs to touchdown on the landing site. The slant-range descent models an atmospheric entry and is where the THRN sensor is active.

The trajectory shown here is anchored in the planet-fixed frame with axes aligned with the vehicle's initial UVW axes, where U is the "up" direction and is found by taking the position vector of the IMU in the planet-fixed frame over its magnitude:

$$\hat{\mathbf{U}} = \frac{\mathbf{r}^f_{imu}}{\|\mathbf{r}^f_{imu}\|}. \tag{6.1}$$

The V direction is found by unitizing the initial IMU velocity vector as found by taking its position vector cross-product with the planet's angular velocity, $\omega^f_{f/i}$, which is assumed known based on previously-made astronomical measurements:

$$\hat{\mathbf{V}} = \frac{\mathbf{r}^f_{imu} \times \omega^f_{f/i}}{\|\mathbf{r}^f_{imu} \times \omega^f_{f/i}\|}. \tag{6.2}$$

The final direction, W, is found to complete the right-handed coordinate system by means of a unitized cross-product between U and V:

$$\hat{\mathbf{W}} = \frac{\hat{\mathbf{U}} \times \hat{\mathbf{V}}}{\|\hat{\mathbf{U}} \times \hat{\mathbf{V}}\|}. \tag{6.3}$$

This coordinate system is denoted with a superscript ^{UVW}, and is fixed in the planet-fixed frame, because it is defined at the initial position of the spacecraft and held static. As such, the transformation between the ECEF frame and the UVW frame is a constant matrix.

In real test flight scenarios, it is desirable to control not only the position and velocity, but acceleration and jerk too, imposed by thrust limitations of the engine, controller time constants, or chosen to limit wear on the vehicle. An 8th order spline was used to achieve control over the change in acceleration. At the start and end of each spline section, the position, velocity, acceleration, and/or change in acceleration can be defined and a trajectory meeting those conditions is created. Using an analytic formula also allows for perfect derivative information, which is useful for computing state errors. The trajectories are first generated in the UVW frame and then transformed into the inertial frame, as the accelerometer and gyroscope measure acceleration and angular velocity of the IMU relative to the fixed inertial system.

The same spline formula was used to create the translational trajectory and rotational trajectory.

6.3.1 Trajectory Plots

In creating the trajectory, a set of initial and final position, velocity, acceleration, and jerk values were input for each section of the flight: pre-ascent,

ascent, slant-range descent, and final descent. The rotational trajectory had more sections: pre-ascent, ascent, pitch-over, slant-range descent part 1, slant-range descent part 2, pitch-over recovery, and final descent. Figure 6.2 shows the ascent, slant-range descent, and final descent flight path in UVW coordinates. The middle vertical lines are included to improve readability of the graph.

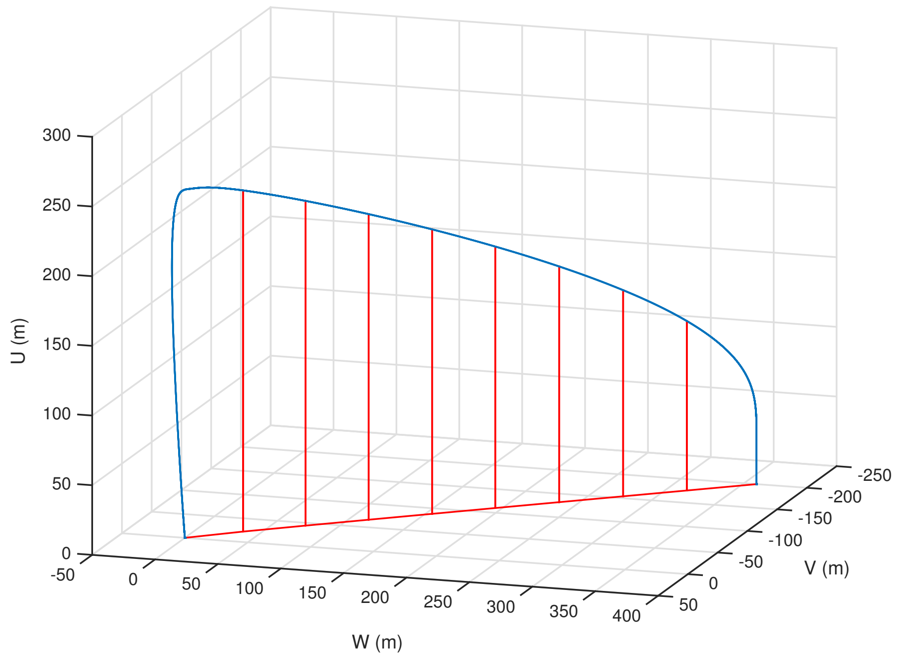

Figure 6.2: Three-dimensional fight path in UVW coordinates

The following Figures 6.3 and 6.4 break down the translational trajectory into position, velocity, and acceleration all defined in the UVW frame, followed by the rotational trajectory about the body-fixed frame.

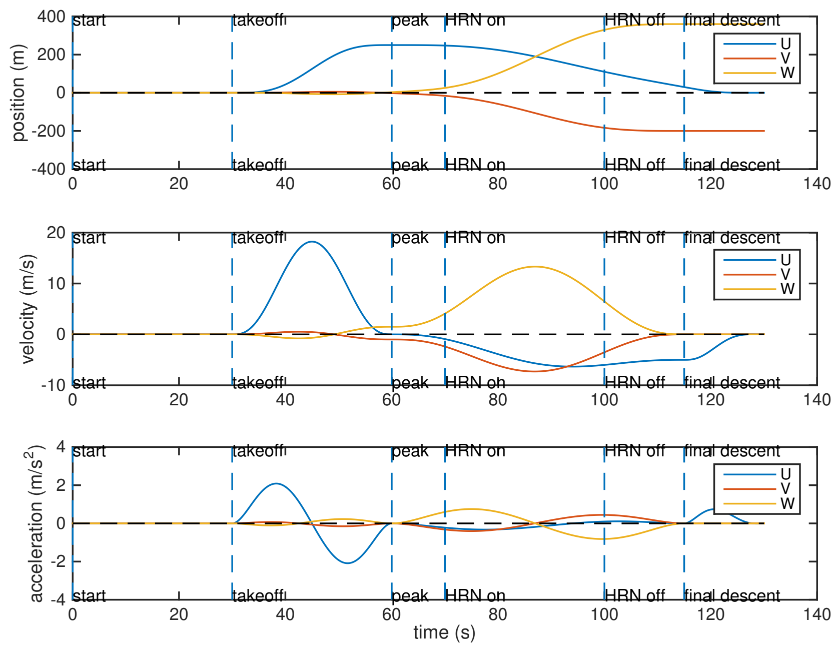

Figure 6.3: UVW-frame translational trajectory

6.3.2 Trajectory Integration Error

Although the perfect states are known for position, velocity, acceleration, and attitude, the only measured quantities are acceleration and angular velocity of the IMU relative to the inertial system. The propagation stage uses a 4th-order Runge-Kutta integration method for accuracy with the understanding that the time steps over which integration occurs are small enough to accumulate very little error. The following plots are given to show the scale of the error due to numerical integration under an assumption of perfect measurements. The input acceleration and angular velocity were taken from the true state values. Also, each state was initialized with the true state values. There is no attitude integration error, as the

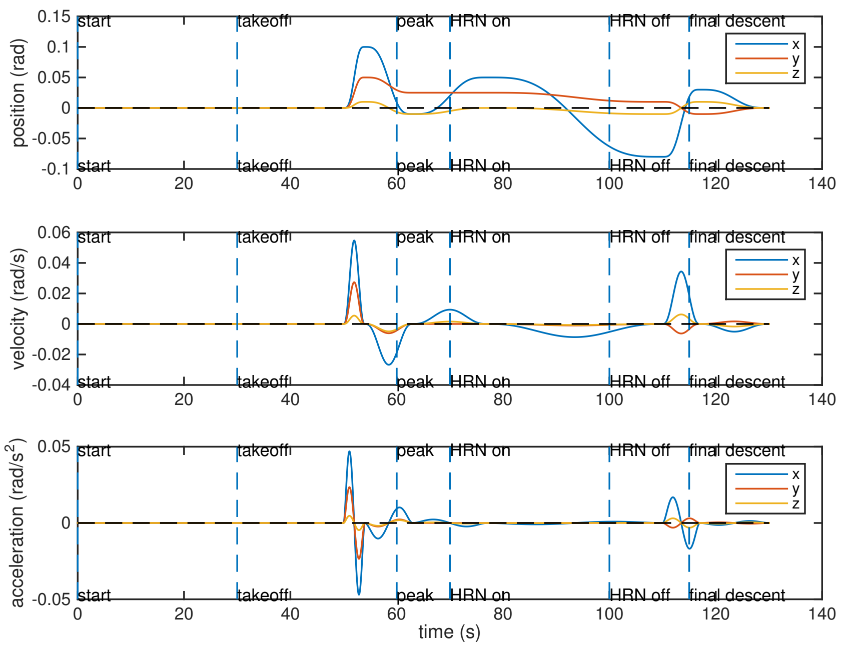

Figure 6.4: Body-frame rotational trajectory

true attitude quaternions are created using the same integration method as is used to propagate the estimated attitude quaternion.

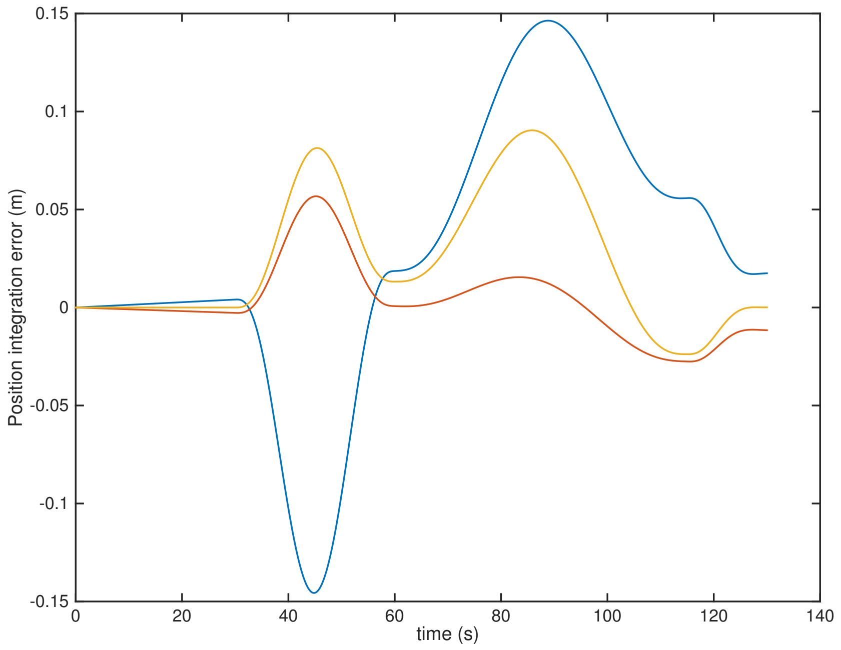

Figure 6.5: Position error due to numerical integration

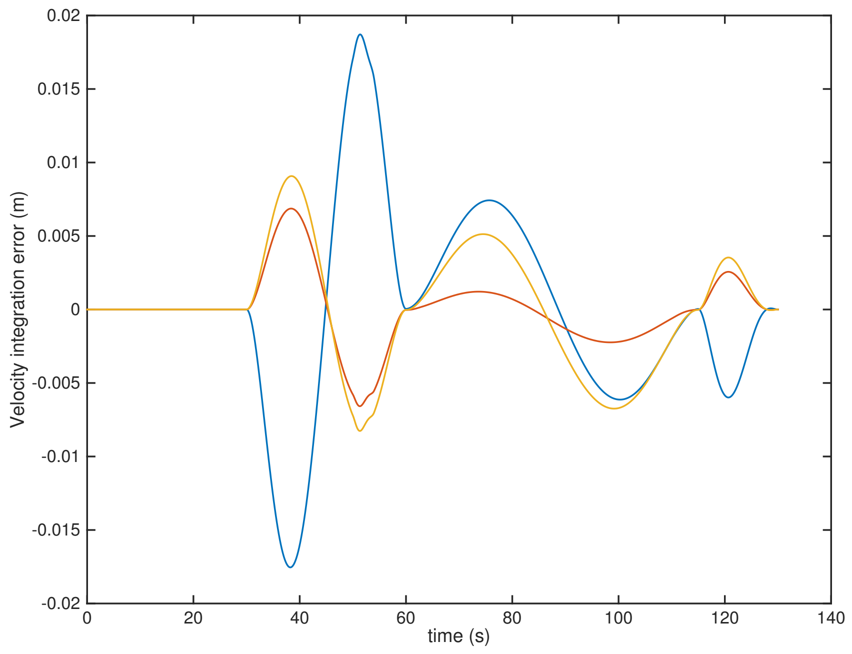

Figure 6.6: Velocity error due to numerical integration

CHAPTER 7
Simulation Results

This chapter presents the results from each simulation mode. First, the *Monte Carlo* aggregate error and covariance results are shown, proving that the filter accurately estimates the error statistics of the state variables. Next, the error-budget table is shown for two key moments of the flight, showing error sources and total estimation error right before the THRN sensor activates, and as final descent begins. This is followed by select figures from the sensitivity analysis, showing the major contributors to state estimate error. Finally, some of the pertinent single-run figures are shown, proving that the THRN sensor effectively reduces state estimate error, ultimately proving that the sensor model formulation is effective.

7.1 *Monte Carlo* Analysis

The *Monte Carlo* analysis is done to ensure the filter functions properly based on two main measures. First, the mean estimation error across all simulations is calculated. A mean of zero verifies that the filter is acting as an unbiased estimator. Second, the estimation error variance is calculated across all simulations. If the calculated variance matches the diagonal covariance values for each state, the filter accurately estimates the state variable error.

These tests are necessary for verifying the performance of an extended

Kalman filter because of the inherent nonlinearity of nature and the systems that are modeled. Thus, if the EKF passes these two tests, the statistical parameters within the filter, many of which are initialized or set as tuning parameters, account for nonlinearities that are not modeled or linearized during filter processing.

The nominal values used to create the varying initial error conditions are in Table 7.1. The values given for initial state errors are the square-root diagonal initial covariances, effectively used as the standard deviation values for transforming normally-distributed random variables with unit variance to be of matching variance to the initial state covariance diagonal value. For example, the position state variables are initialized with an error standard deviation of $25(m)$, so to create one initial position error value, the nominal (true) state value is modified by adding a random value created by multiplying $25(m)$ by some random variable sampled from $\sim N(0, 1)$ where $\sim N(\bar{x}, \sigma)$ is the normal distribution with mean $\bar{x}$ and standard deviation σ.

Then, once the initial errors are created, they are processed to ensure that the both the variance truly matches the initial state error covariance value and that the mean is zero[32]. These processed initial error values are then fed to the *Monte Carlo* simulation runs.

Table 7.2 holds the sensor standard deviation values. These values are applied to the true measured quantities similarly to how the initial errors are created, by starting with a zero-mean normally-distributed variable and modifying its variance to match the noise characteristics of the sensor.

Table 7.1: Initial state error standard deviations

State	Variable	Initial uncertainty standard deviation
Position	$\mathbf{r}$	$25(m)$
Velocity	$\mathbf{v}$	$0.1(m/s)$
Attitude	$\boldsymbol{\alpha}$	$0.001(rad)$
Accelerometer bias	$\mathbf{b}_{acc}$	$1 \times 10^{-5}(m/s^2)$
Gyroscope bias	$\mathbf{b}_{gyro}$	$1 \times 10^{-5}(rad/s)$
THRN bias	$\mathbf{b}_{thrn}$	$0.1(m)$
Velocimeter bias	$\mathbf{b}_{vel}$	$0.005(m/s)$
ILP bias (map-tie error)	$\mathbf{b}_{ilp}$	$1(m)$
Altimeter bias	$\mathbf{b}_{alt}$	$0.1(m)$
Star camera bias	$\mathbf{b}_{sc}$	$5 \times 10^{-5}(rad)$
Accelerometer misalignment	$\mathbf{m}_{acc}$	$1 \times 10^{-6}(rad)$
Accelerometer nonorthogonality	$\mathbf{n}_{acc}$	$1 \times 10^{-6}(rad)$
Accelerometer scale factor	$\mathbf{s}_{acc}$	$1 \times 10^{-6}(ppm)$
Gyroscope misalignment	$\mathbf{m}_{gyro}$	$1 \times 10^{-6}(rad)$
Gyroscope nonorthogonality	$\mathbf{n}_{gyro}$	$1 \times 10^{-6}(rad)$
Gyroscope scale factor	$\mathbf{s}_{gyro}$	$1 \times 10^{-6}(ppm)$

Table 7.2: Sensor noise standard deviations

IMU noise		
Accelerometer	$\boldsymbol{\eta}_{acc}$	$1 \times 10^{-4}(m/s^2)$
Gyroscope	$\boldsymbol{\eta}_{gyro}$	$1 \times 10^{-5}(rad/s)$
Sensor Noise		
THRN	$\boldsymbol{\eta}_{thrn}$	$1(m)$
Velocimeter	$\boldsymbol{\eta}_{vel}$	$0.05(m/s)$
Altimeter	η_{alt}	$1(m)$
Star Camera	$\boldsymbol{\eta}_{sc}$	$5 \times 10^{-4}(rad)$

Figure A.1 shows a sample of the plots that are generated from the *Monte Carlo* analysis data. These plots show the mean estimation error, variance of the error vectors across all simulation runs, and the mean estimation covariance values. The rest of the plots of this type are included in Appendix A. The error-variance and covariance values match very closely for the inertial X-position estimate, showing that the filter is tuned properly to account for nonlinearities that are not modeled, and it is acting as an unbiased estimator. The same close matching is seen across all plots for the first nine state variables, showing that the filter is accurately estimating position, velocity, and attitude.

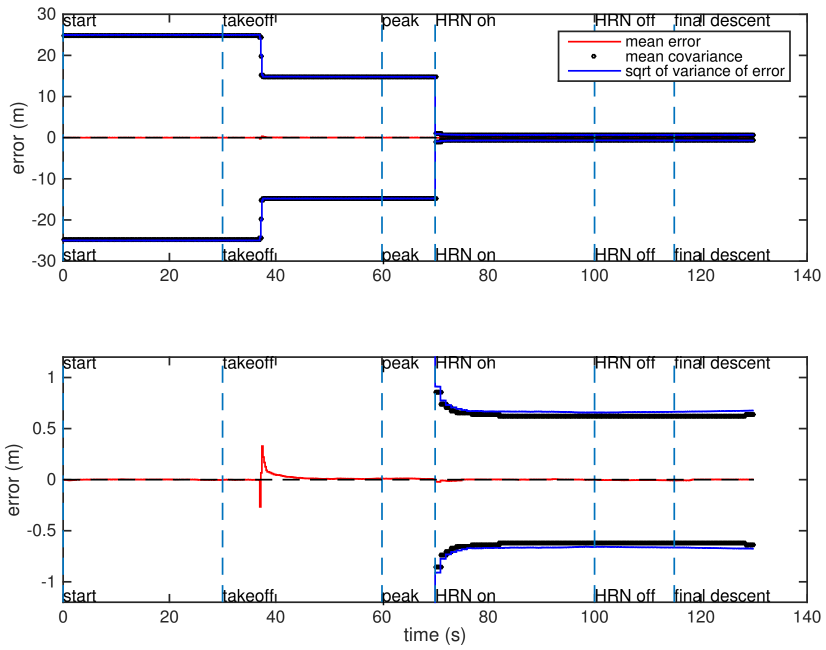

Figure 7.1: *Monte Carlo* inertial X position mean error, error variance, and mean covariance

The rest of the state variables follow suit, with the mean estimate error about zero and the error variance matching the state covariance. Most of the other estimated states do not see a marked reduction in estimation error covariance, likely due to the states not being highly observable. However, the ILP (map-tie error) bias states and the velocimeter bias states do show a reduction in estimation error covariance, and so these are shown.

The graphs shown in Figures 7.2 and 7.3, show the ILP bias states and velocimeter bias states. The reduction in error covariance in the ILP bias graphs occurs as the THRN sensor activates, proving that the THRN sensor does reduce the map-tie errors that may be associated with a preloaded intended landing position. The velocimeter graphs show a similar reduction in error covariance over all time. It is interesting to see a larger reduction in the velocimeter Z-channel error covariance which is caused by the large amount of measurement information in that channel, including the altimeter, velocimeter, and THRN sensor when active.

The velocimeter bias state plots are followed by Figures 7.4 through 7.6, which show the mean velocimeter residual data for all *Monte Carlo* simulation runs, as well as the standard deviation for all residuals. This is to prove that the measurement noise is truly zero-mean, with a standard deviation prescribed by the sensor noise values found in Table 7.2.

The figures show that the extended Kalman filter is well-tuned, as all error means are zero, proving the filter is an unbiased estimator, and the error variance matches the filter covariance, proving that the filter accurately estimates the state error uncertainty.

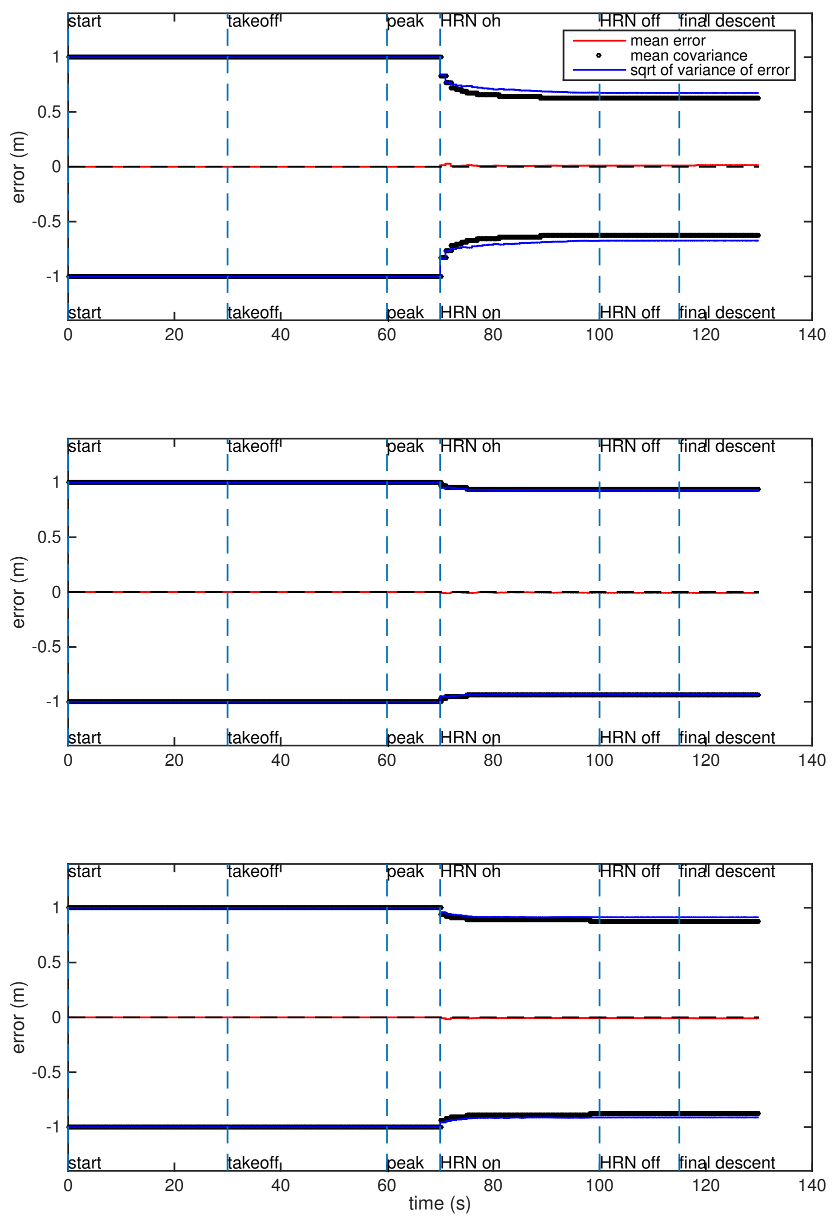

Figure 7.2: *Monte Carlo* ILP-frame ILP bias mean error, error variance, and mean covariance

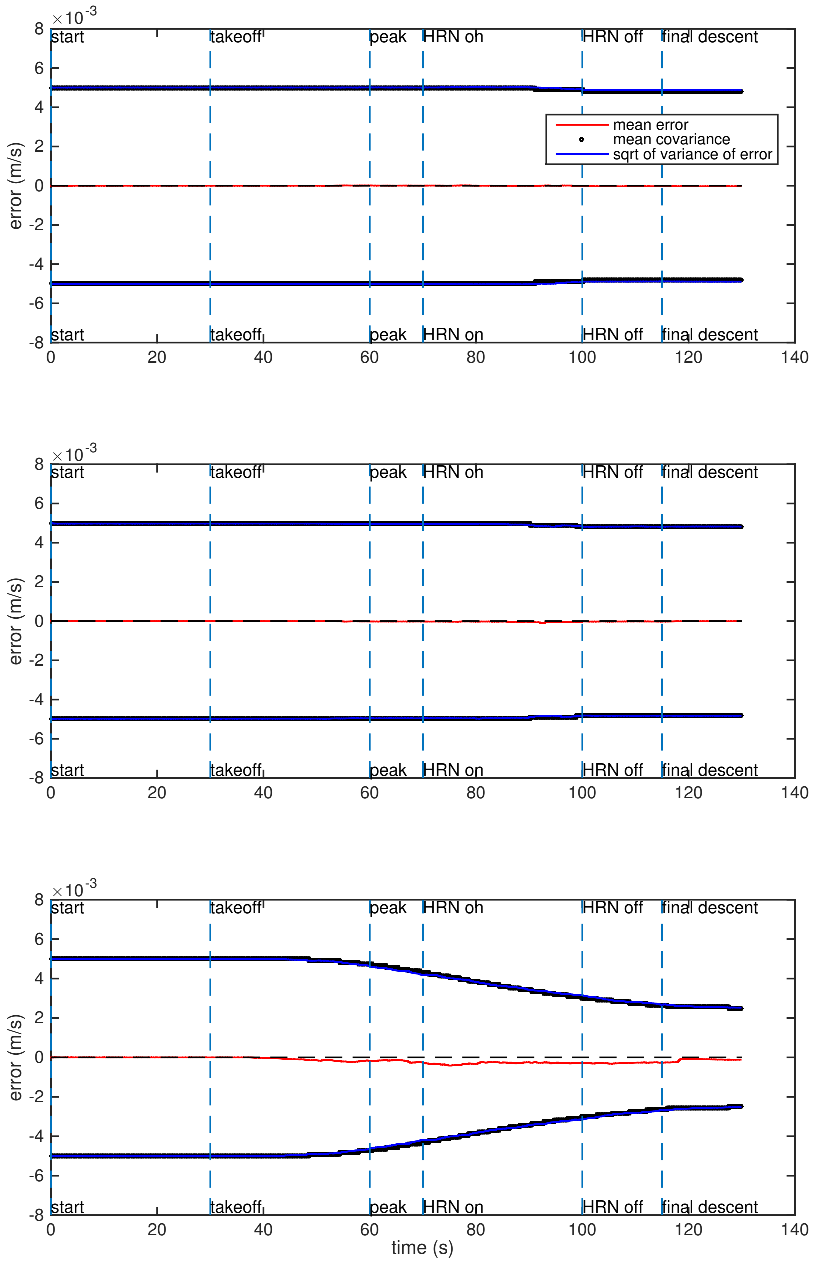

Figure 7.3: *Monte Carlo* velocimeter-frame velocimeter bias mean error, error variance, and mean covariance

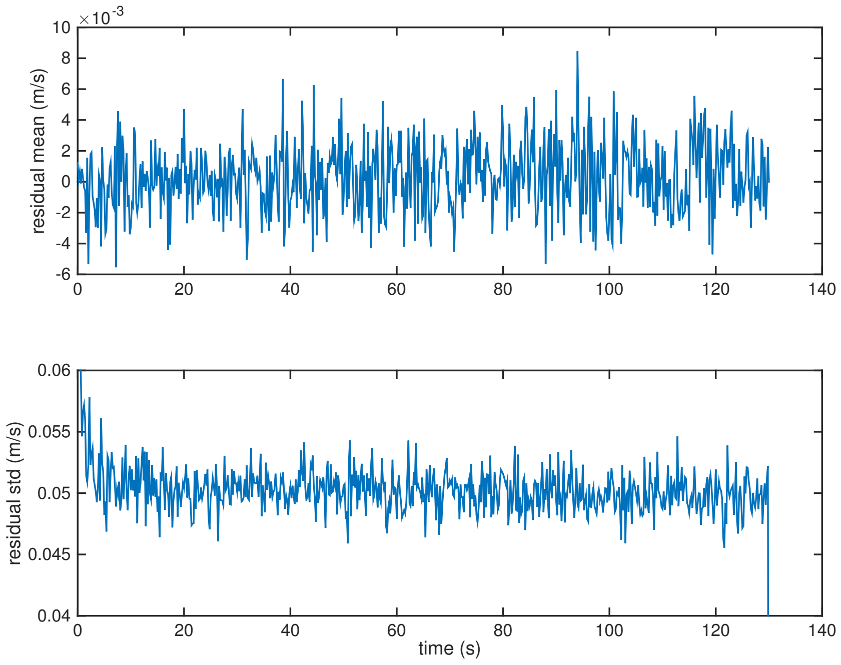

Figure 7.4: *Monte Carlo* velocimeter-frame X velocimeter residual and standard deviation

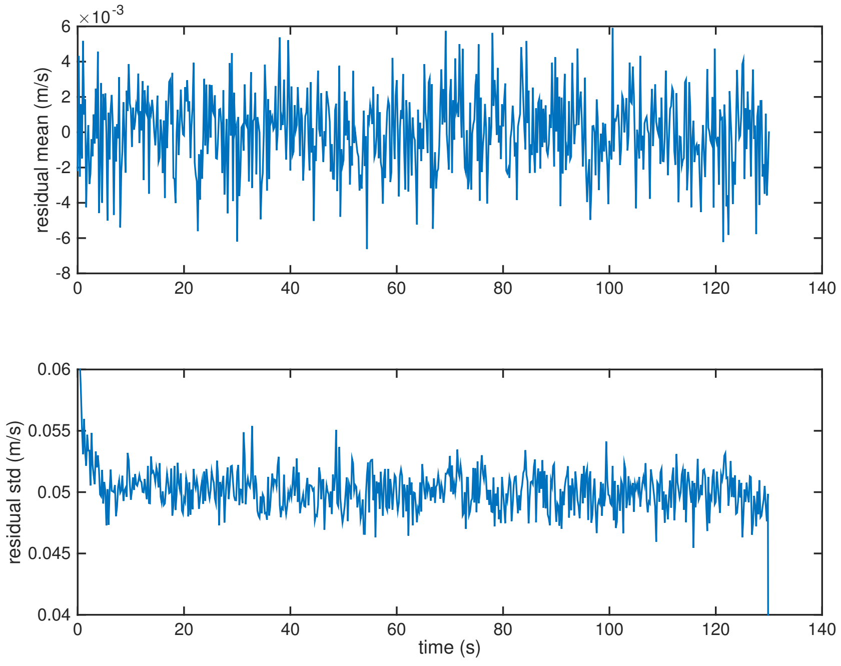

Figure 7.5: *Monte Carlo* velocimeter-frame Y velocimeter residual and standard deviation

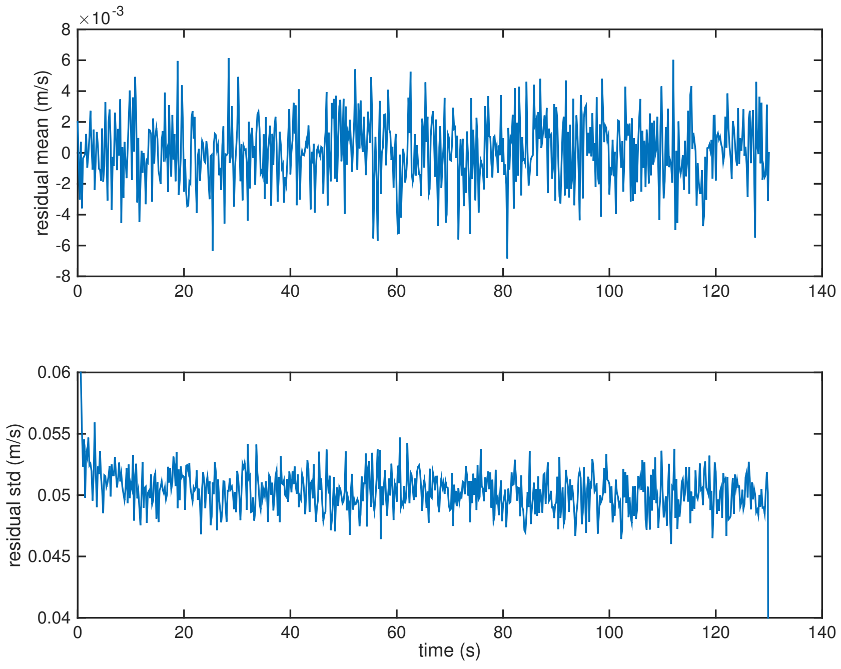

Figure 7.6: *Monte Carlo* velocimeter-frame Z velocimeter residual and standard deviation

7.2 Error-Budget Analysis

Once the filter is confirmed to be well-tuned through *Monte Carlo* covariance analysis, an error budget is created to show the effect of different error groups on the total estimation error at some point in time. The process for completing an error budget analysis is as follows:

1. Complete a single-run with all error sources active
2. From the single-run, save the Kalman gain matrix for all time, as well as the covariance matrix diagonal for times of interest
3. For each error group, do the following:
    (a) Zero all error sources that are not part of the error group being tested, including process noise, initial covariance values, and measurement noise variance for all states and sensors
    (b) Run the simulation, using the saved Kalman gain matrix to update the state and covariance values
    (c) Once the run is complete, save the covariance matrix diagonals for the times of interest
4. Create a table with rows dedicated to error groups and columns to state variables, and populate the cells with each error group's affect on that state variable's estimation error, which are the square-root values of the associated covariance diagonals
5. Compute the root-sum-square value for each column of error group values and compare to the root-covariance values from the initial full single-run. If they are close, the error sources are well-accounted for

Figure 7.7 illustrates the error budget simulation process and looping structure. Completing this process allows the filter designer to understand which error groups affect which estimated states the most. In some cases, an error group may greatly impact the estimation error for certain states. To mitigate the effects of these error groups on the overall state estimate, sometimes more information can be included in the filter by means of higher-fidelity models or more measurements.

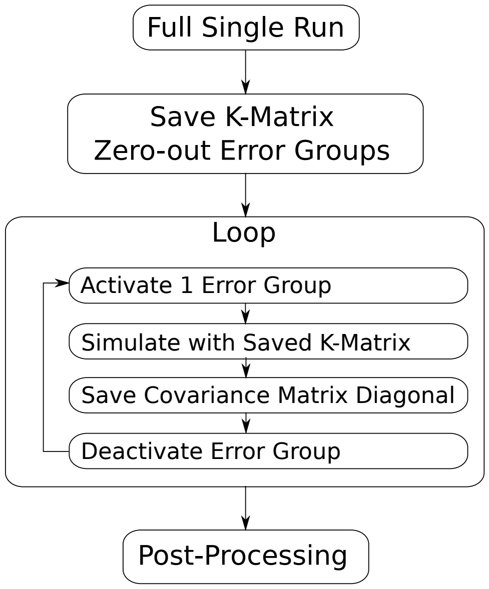

Figure 7.7: Error budget simulation flow

Error groups are chosen based on combinations of sources of uncertainty in the filter. This means that some state variables are grouped together; for example, the three dimensions of inertial position uncertainty may make up one error group. Other groups can be made for process noise and measurement noise variance values. Table 7.3 shows the error groups used for this analysis.

Table B.1 shows a sample part of the complete error budget table for the system at the simulation time immediately preceding THRN activation. The complete error budget tables for this time and for the simulation time immediately before final descent are located in Appendix B. For compactness, the state variables

Table 7.3: Error group descriptions

Error group	Description
1	Inertial position uncertainty
2	Inertial velocity uncertainty
3	Body-fixed attitude uncertainty
4	Accelerometer bias uncertainty
5	Gyroscope bias uncertainty
6	THRN sensor bias uncertainty
7	Velocimeter sensor bias uncertainty
8	ILP bias (map-tie error) uncertainty
9	Altimeter sensor bias uncertainty
10	Star camera sensor bias uncertainty
11	Accelerometer misalignment uncertainty
12	Accelerometer nonorthogonality uncertainty
13	Accelerometer scale factor uncertainty
14	Gyroscope misalignment uncertainty
15	Gyroscope nonorthogonality uncertainty
16	Gyroscope scale factor uncertainty
17	THRN sensor measurement noise variance uncertainty
18	Velocimeter measurement noise variance uncertainty
19	Altimeter measurement noise variance uncertainty
20	Star camera measurement noise variance uncertainty

as found in Table 7.1 are used to label each column.

It is clear that only some error groups greatly affect each state variable. The summary Tables 7.5 and 7.6 show the five error groups that affect each state variable the most for the pre-THRN time and as final descent begins, while Figures 7.8 and 7.9 show histograms of the error groups' overall top effects. It can be seen that error groups corresponding to the velocimeter measurement noise variance, the star camera measurement noise variance, and the inertial position error uncertainty affect the state estimate the most before the THRN sensor activates, while as final descent begins the greatest effects are seen from the THRN, velocimeter, and star camera measurement noise variance values.

Table 7.4: Error budget for position, velocity, and attitude states- pre-THRN

	$\mathbf{r}_x$	$\mathbf{r}_y$	$\mathbf{r}_z$	$\mathbf{v}_x$	$\mathbf{v}_y$	$\mathbf{v}_z$	$\boldsymbol{\alpha}_x$	$\boldsymbol{\alpha}_y$	$\boldsymbol{\alpha}_z$
1	14.82	23.43	21.93	0.007623	0.003409	0.004437	6.393e-07	6.085e-07	2.052e-08
2	0.2024	0.4208	0.192	0.0002576	0.0003685	0.0003071	7.808e-08	8.371e-08	1.191e-08
3	0.000402	0.0005441	0.0005091	9.31e-06	8.346e-06	4.417e-06	1.064e-06	1.059e-06	1.052e-06
4	0.001078	0.00322	0.002369	0.0003036	0.0003305	0.0003249	2.028e-07	2.003e-07	2.257e-08
5	0.0004542	0.0005985	0.0004297	4.819e-06	3.635e-05	3.686e-05	6.266e-06	6.262e-06	6.286e-06
6	0	0	0	0	0	0	0	0	0
7	0.2068	0.3176	0.2975	0.00407	0.004826	0.004672	1.875e-07	6.299e-08	9.507e-09
8	0	0	0	0	0	0	0	0	0
9	0.08041	0.03457	0.0479	3.108e-05	6.584e-06	1.705e-05	1.032e-09	1.758e-09	5.229e-11
10	0.008656	0.01163	0.01025	0.0001118	0.000198	0.0001611	4.987e-05	4.977e-05	4.976e-05
11	0.0003785	0.00172	0.0006518	0.0002009	0.0003015	0.000281	1.978e-07	1.975e-07	2.24e-08
12	0.0004512	0.001659	0.0006675	0.0001864	0.0003043	0.0002866	1.982e-07	1.974e-07	2.237e-08
13	0.0009489	0.002681	0.002225	0.0002183	9.946e-05	0.0001325	2.805e-08	2.022e-09	1.882e-10
14	1.039e-06	4.84e-07	1.735e-06	1.263e-07	5.157e-07	2.151e-07	4.323e-09	2.17e-08	2.066e-08
15	6.441e-07	1.027e-06	2.326e-06	3.09e-07	4.398e-07	2.124e-07	7.393e-09	2.133e-08	2.014e-08
16	5.901e-07	1.581e-06	4.694e-07	2.536e-07	2.348e-07	4.552e-07	2.055e-08	6.872e-10	6.055e-09
17	0	0	0	0	0	0	0	0	0
18	0.1137	0.1737	0.1629	0.002228	0.002626	0.002553	3.048e-06	3.027e-06	3.371e-07
19	0.07323	0.02875	0.04424	0.00206	0.0008911	0.00123	7.49e-07	2.881e-07	1.379e-08
20	0.005774	0.007764	0.006436	8.299e-05	0.0003016	0.0002956	5.344e-05	5.348e-05	5.363e-05
root sum of squares	14.83	23.44	21.93	0.009175	0.00657	0.007073	7.344e-05	7.339e-05	7.344e-05
full run root cov	14.83	23.44	21.93	0.005418	0.005726	0.005521	7.344e-05	7.339e-05	7.344e-05
difference	-2.556e-07	-4.088e-05	-8.502e-06	-0.003757	-0.0008444	-0.001552	-2.595e-09	-2.359e-09	-2.797e-11

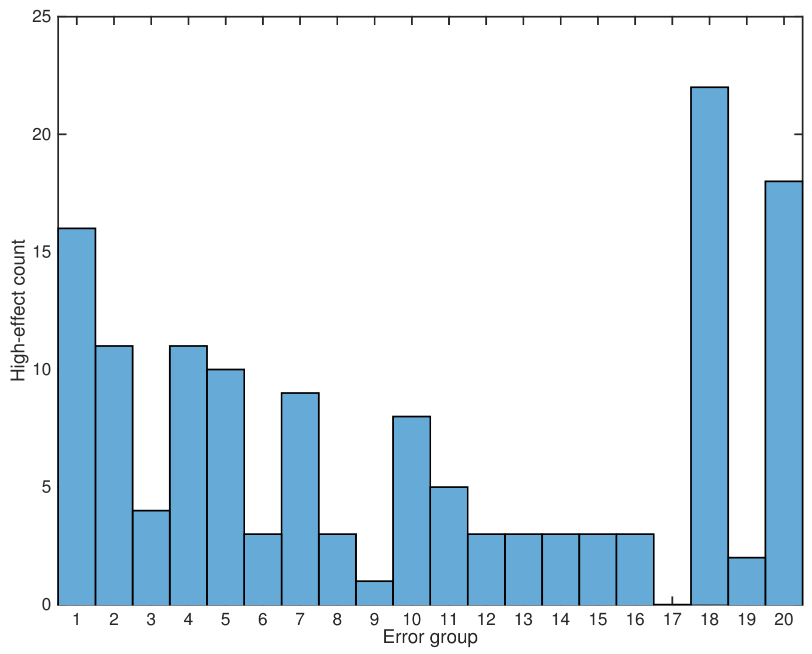

Figure 7.8: Error budget greatest-effect group count - pre-THRN

Table 7.5: Error budget greatest-effect summary - pre-THRN

	Primary	Second	Third	Fourth	Fifth
$\mathbf{r}_x$	1	7	2	18	9
$\mathbf{r}_y$	1	2	7	18	9
$\mathbf{r}_z$	1	7	2	18	9
$\mathbf{v}_x$	1	7	18	19	4
$\mathbf{v}_y$	7	1	18	19	2
$\mathbf{v}_z$	7	1	18	19	4
$\boldsymbol{\alpha}_x$	20	10	5	18	3
$\boldsymbol{\alpha}_y$	20	10	5	18	3
$\boldsymbol{\alpha}_z$	20	10	5	3	18
$\mathbf{b}_{acc,x}$	4	18	11	12	2
$\mathbf{b}_{acc,y}$	4	18	11	12	2
$\mathbf{b}_{acc,z}$	4	18	1	19	7
$\mathbf{b}_{gyro,x}$	20	5	18	3	10
$\mathbf{b}_{gyro,y}$	20	5	18	3	10
$\mathbf{b}_{gyro,z}$	20	5	3	10	18
$\mathbf{b}_{hrn,x}$	6	1	2	3	4
$\mathbf{b}_{hrn,y}$	6	1	2	3	4
$\mathbf{b}_{hrn,z}$	6	1	2	3	4
$\mathbf{b}_{vel,x}$	7	2	18	1	19
$\mathbf{b}_{vel,y}$	7	2	18	1	19
$\mathbf{b}_{vel,z}$	1	7	19	18	2
$\mathbf{b}_{ilp,x}$	8	1	2	3	4
$\mathbf{b}_{ilp,y}$	8	1	2	3	4
$\mathbf{b}_{ilp,z}$	8	1	2	3	4
b_{alt}	9	19	1	7	18
$\mathbf{b}_{sc,x}$	10	3	20	18	19
$\mathbf{b}_{sc,y}$	10	3	20	18	1
$\mathbf{b}_{sc,z}$	10	20	3	18	5
$\mathbf{m}_{acc,x}$	11	18	4	12	2
$\mathbf{m}_{acc,y}$	11	18	4	12	2
$\mathbf{m}_{acc,z}$	11	18	4	12	20
$\mathbf{n}_{acc,x}$	12	18	4	11	2
$\mathbf{n}_{acc,y}$	12	18	4	11	2
$\mathbf{n}_{acc,z}$	12	18	4	1	11
$\mathbf{s}_{acc,x}$	13	18	4	12	11
$\mathbf{s}_{acc,y}$	13	18	4	12	11
$\mathbf{s}_{acc,z}$	13	18	1	19	7
$\mathbf{m}_{gyro,x}$	14	20	5	10	18
$\mathbf{m}_{gyro,y}$	14	20	10	5	18
$\mathbf{m}_{gyro,z}$	14	20	18	5	10
$\mathbf{n}_{gyro,x}$	15	20	5	10	18
$\mathbf{n}_{gyro,y}$	15	20	5	10	18
$\mathbf{n}_{gyro,z}$	15	20	18	5	10
$\mathbf{s}_{gyro,x}$	16	20	18	5	10
$\mathbf{s}_{gyro,y}$	16	20	5	18	10
$\mathbf{s}_{gyro,z}$	16	20	10	18	5

Table 7.6: Error budget greatest-effect summary - final descent

	First	Second	Third	Fourth	Fifth
$\mathbf{r}_x$	8	17	1	9	19
$\mathbf{r}_y$	8	17	7	6	1
$\mathbf{r}_z$	8	17	7	6	1
$\mathbf{v}_x$	1	7	19	17	18
$\mathbf{v}_y$	7	17	18	1	19
$\mathbf{v}_z$	7	1	17	18	19
$\boldsymbol{\alpha}_x$	10	20	1	19	7
$\boldsymbol{\alpha}_y$	10	20	1	5	18
$\boldsymbol{\alpha}_z$	10	20	5	3	18
$\mathbf{b}_{acc,x}$	4	18	11	12	17
$\mathbf{b}_{acc,y}$	4	18	11	12	17
$\mathbf{b}_{acc,z}$	4	18	1	19	13
$\mathbf{b}_{gyro,x}$	20	5	18	3	10
$\mathbf{b}_{gyro,y}$	20	5	3	18	10
$\mathbf{b}_{gyro,z}$	20	5	18	3	10
$\mathbf{b}_{hrn,x}$	6	17	1	8	7
$\mathbf{b}_{hrn,y}$	6	17	1	19	7
$\mathbf{b}_{hrn,z}$	6	8	17	7	9
$\mathbf{b}_{vel,x}$	7	17	18	2	1
$\mathbf{b}_{vel,y}$	7	17	18	2	1
$\mathbf{b}_{vel,z}$	1	19	18	7	17
$\mathbf{b}_{ilp,x}$	8	17	9	6	19
$\mathbf{b}_{ilp,y}$	8	17	1	9	6
$\mathbf{b}_{ilp,z}$	8	17	9	6	1
b_{alt}	9	8	17	6	19
$\mathbf{b}_{sc,x}$	10	1	19	3	7
$\mathbf{b}_{sc,y}$	10	20	1	3	19
$\mathbf{b}_{sc,z}$	10	20	3	18	17
$\mathbf{m}_{acc,x}$	11	18	4	12	17
$\mathbf{m}_{acc,y}$	11	18	4	12	17
$\mathbf{m}_{acc,z}$	11	18	17	4	12
$\mathbf{n}_{acc,x}$	12	18	4	11	17
$\mathbf{n}_{acc,y}$	12	18	4	11	17
$\mathbf{n}_{acc,z}$	12	18	17	4	11
$\mathbf{s}_{acc,x}$	13	18	17	4	12
$\mathbf{s}_{acc,y}$	13	18	17	4	12
$\mathbf{s}_{acc,z}$	13	18	1	19	4
$\mathbf{m}_{gyro,x}$	14	20	10	18	5
$\mathbf{m}_{gyro,y}$	14	20	18	10	17
$\mathbf{m}_{gyro,z}$	14	20	18	10	1
$\mathbf{n}_{gyro,x}$	15	20	18	10	5
$\mathbf{n}_{gyro,y}$	15	20	18	10	17
$\mathbf{n}_{gyro,z}$	15	20	18	10	1
$\mathbf{s}_{gyro,x}$	16	20	18	1	19
$\mathbf{s}_{gyro,y}$	16	20	18	10	5
$\mathbf{s}_{gyro,z}$	16	20	18	10	5

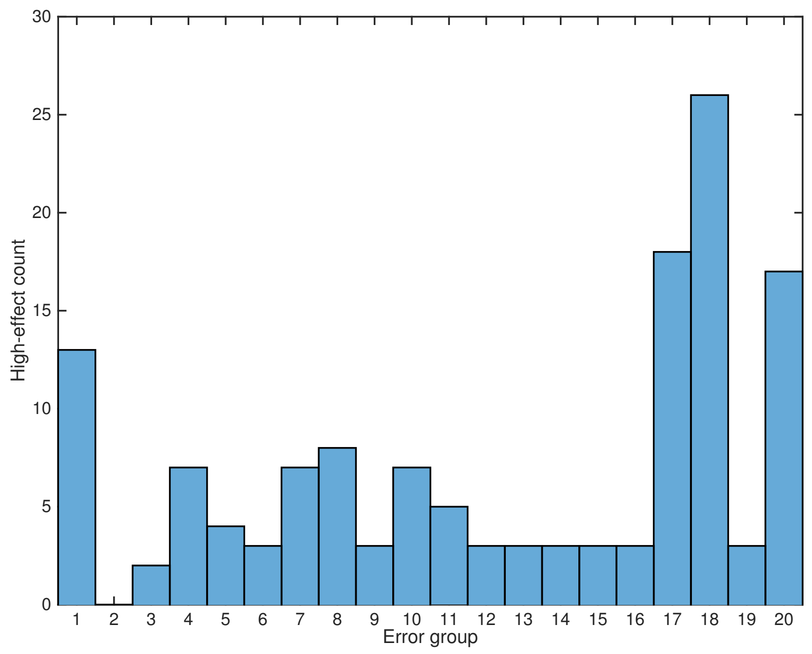

Figure 7.9: Error budget greatest-effect group count - final descent

7.3 Sensitivity Analysis

The sensitivity analysis follows directly from the error budget. Once an error budget is made, and the greatest sources of estimation error are determined, it is useful to know how the estimation error might change as a result in more or less uncertainty in the greatest-effect error groups. The process for sensitivity analysis is as follows:

1. Determine for which state variables and related error groups the sensitivity should be calculated based on greatest-effect (or compute sensitivity for all pairings)
2. For the designated state variable and error group, do the following:
   (a) Take the column of error group values for a single state variable, and single out the error group value being tested

(b) Scale the error group value along the desired test range (e.g. from 0.1x to 10x)

(c) Recompute the total estimation error of the column via the root-sum-square method used for the error budget

(d) Plot the total estimation error (or a measure of the change) as a function of scale-factor

The end result of the sensitivity analysis is to find which error sources affect the overall state estimation error the least, which would enable the designer to remove those error sources from the filter design. This results in a simpler filter, which takes less memory and computational power to operate. Finally, with a more efficient filter, the navigation system may be able to run state updates more often, potentially taking in more measurement data and resulting in a better overall navigation solution.

For this thesis, the sensitivity was computed for all state variables an error group pairs. However, for brevity, the following pairs are presented to illustrate the greatest-effect sensitivities, and all plots can be found in Appendix C: position and velocity error sensitivity to position and velocimeter bias uncertainty for pre-THRN activation in Figures C.1 and C.2, attitude error sensitivity to star camera bias and noise variance uncertainty for pre-THRN activation in Figures C.3 and C.4, followed by position and velocity error sensitivity to velocimeter bias, ILP (map-tie) error, and THRN sensor noise variance uncertainty for final descent in Figures C.5, C.6, and C.7, and finally attitude error sensitivity to star camera bias and noise variance uncertainty for final descent in Figures C.8 and C.9. The first of this series of plots is provided here for reference.

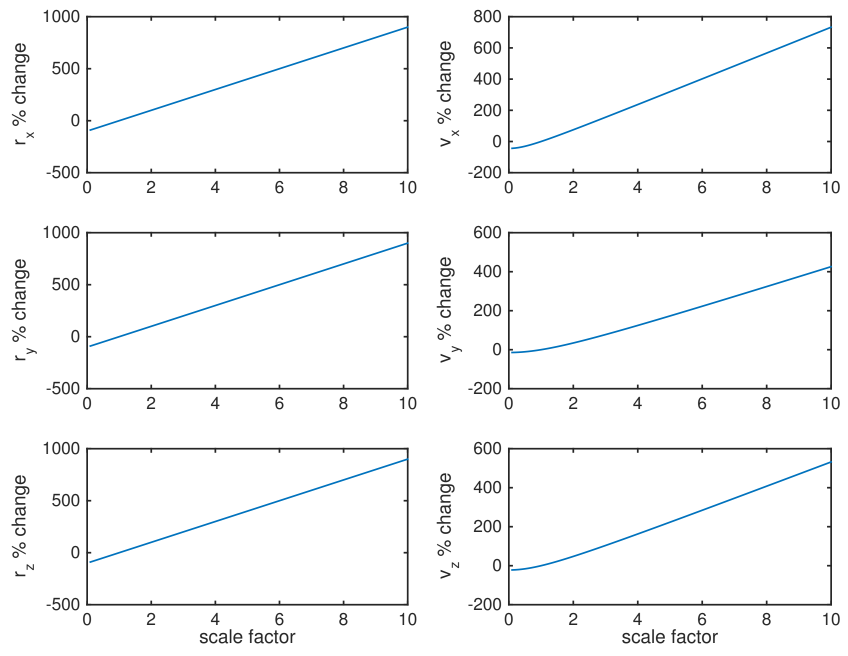

Figure 7.10: Position and velocity estimation error variance sensitivity to error group 1, pre-THRN

In general, high sensitivity to an error group means that that error groups' parameters should be as close to truth as possible; in some cases, this is dependent on accurately portraying a sensor's capabilities including the measurement noise variance and sensor bias uncertainty. In other cases, such as for the ILP (map-tie) error uncertainty, the THRN sensor reduces the bias error variance, which would in turn reduce the estimation error variance for other states of interest such as position and velocity. As seen in Figure C.6, an increase of 10x for the ILP (map-tie) error variance amounts to a 9x increase in position estimate error variance. As such, the fact that the THRN sensor reduces the ILP bias error greatly impacts the end-game navigation solution.

Following the sensitivity curves that show the highest changes in error variance, Figures 7.11 through 7.14 show how little the error variance changes due to one error group, the gyroscope misalignment error states. The greatest increase in estimation error variance due to this error group is seen in the final descent attitude states, with a maximum of 0.004% increase. Since the sensitivity for this error group is low for the attitude, and other, state variables, the next revision of this filter's design would likely take this error group out entirely. The same trend holds for the gyroscope nonorthogonality error states and the gyroscope scale factor error states, which affected the state estimation error variance very little. These error groups are also candidates for removal from the state vector.

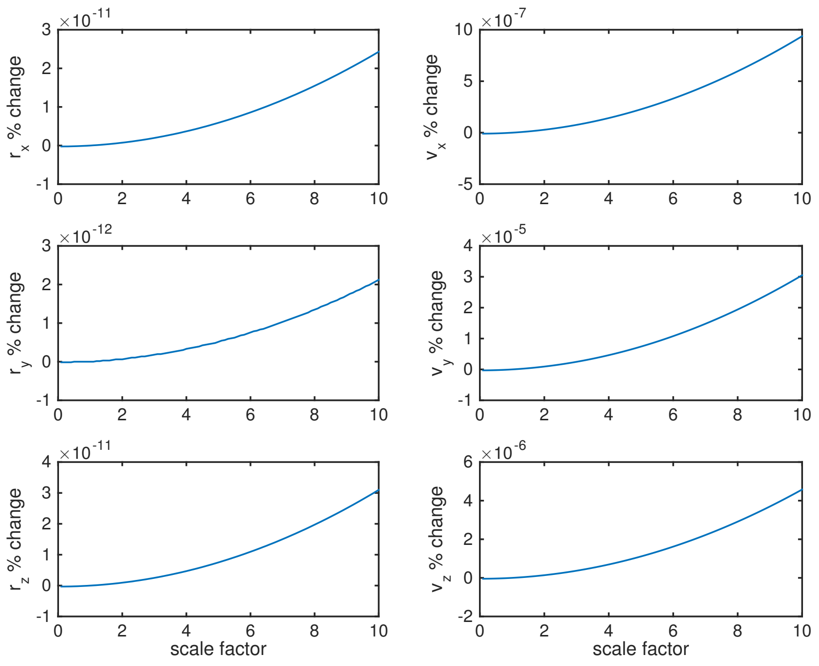

Figure 7.11: Position and velocity estimation error variance sensitivity to error group 14, pre-THRN

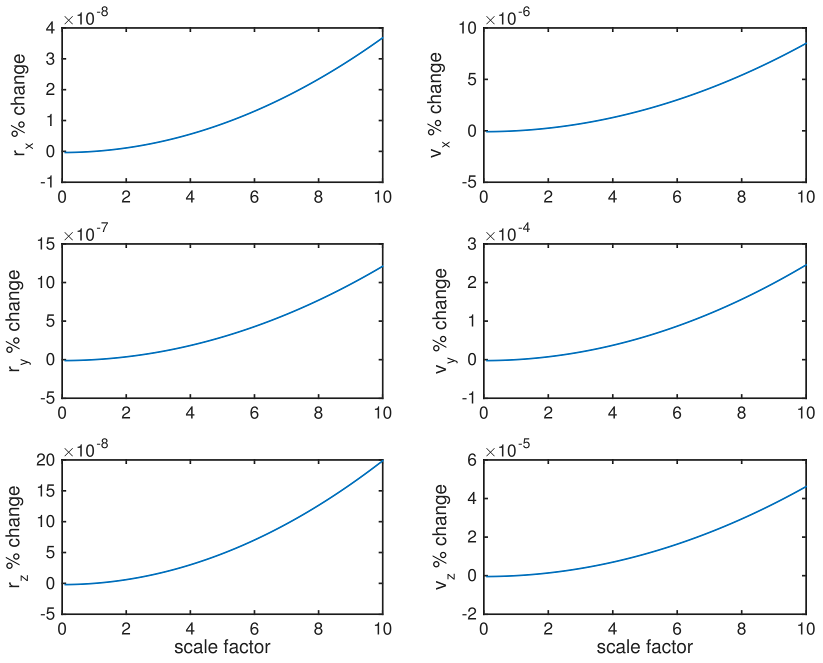

Figure 7.12: Position and velocity estimation error variance sensitivity to error group 14, final descent

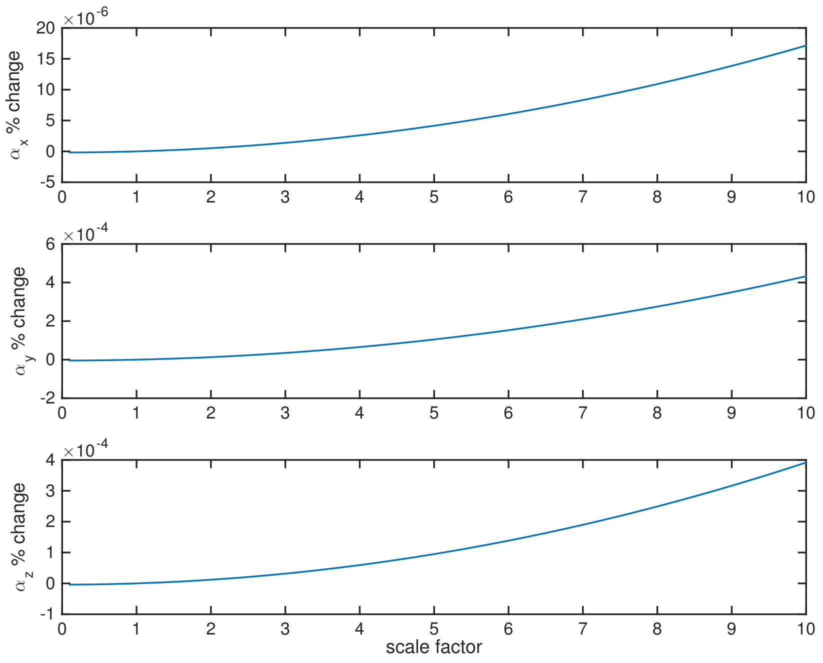

Figure 7.13: Attitude estimation error variance sensitivity to error group 14, pre-THRN

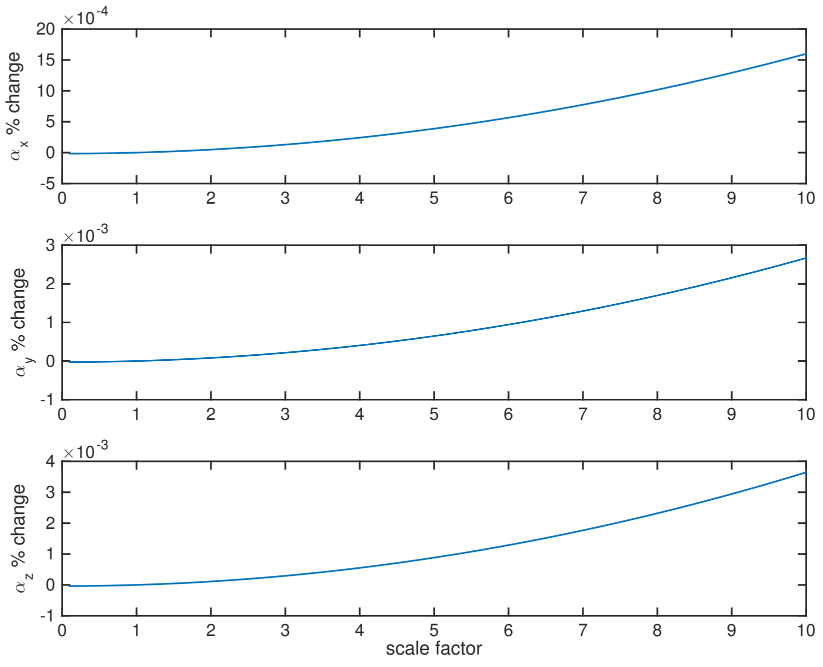

Figure 7.14: Attitude estimation error variance sensitivity to error group 14, final descent

CHAPTER 8
Conclusion

8.1 Thesis Results

The purpose of this thesis was to verify a dual-state extended Kalman filter (EKF) navigation system for planetary landing utilizing a lidar-enabled feature mapping/tracking sensor known as the Terrain/Hazard Relative Navigation (THRN) sensor. Through software simulation of the EKF, including a velocimeter, altimeter, star camera, and THRN sensors, the effectiveness of the THRN sensor was examined through *Monte Carlo* covariance analysis, error budget analysis, and sensitivity analysis.

The simulated trajectory included an ascent phase, a long slant-range descent, and a final descent phase. This simulated entry-descent-landing trajectory was chosen to model a flight test that might be done on Earth for a vehicle that may be used on other planets. Simulation performance was exceptional, with a single simulation covering 130 seconds of flight taking only 10 seconds to compute. This indicates the filter can be used for real-time flight applications.

The *Monte Carlo* covariance analysis proved that the filter worked properly and was well-tuned by matching the covariance values to the variance of estimation error across all *Monte Carlo* runs. The error budget and sensitivity analyses showed that for the controllable states of position, velocity, and attitude, the greatest sources of estimation error were the position, velocity, velocimeter bias, star camera bias, velocimeter measurement variance, and star camera measurement variance errors (pre-THRN activation) and position, ILP (map-tie) bias, star camera bias, THRN measurement variance, and star camera measurement variance errors (final descent).

The THRN sensor worked as expected, greatly reducing positional estimate error variance upon activation. It also reduced the ILP (map-tie) errors, but not to the same degree as the position states. This was done over a timespan of 30 seconds with the THRN sensor running at $1Hz$. In spite of the small operational window, the THRN sensor was able to greatly decrease navigation error, enabling precision landing.

8.2 Future Work

This thesis covers the initial filter design and verification, with emphasis put on complicated modeling of the sensors and environment. To extend this work, a natural next step to take would be simplifying the filter by reducing the length of the state vector. This requires analysis of the error budget and sensitivity data and and intelligent choice of state variables to remove. Then, the navigation system has to be verified again through similar *Monte Carlo*, error budget, and sensitivity analyses. If the resulting navigation system meets the mission-specific estimation error requirements, the reduced filter may be used.

Further, the tested Extended Kalman Filter implementation could be paired with a modeled control system, with the filter estimate feeding into the controller and a thrust model, atmosphere model, wind model, and other disturbance models providing trajectory feedback. This would further prove the THRN sensor in a more complex simulation environment. Success with a more complicated, complete simulation environment would prime the system for testing in real flight conditions.

Finally, the THRN sensor model requires the use of a LIDAR camera system to produce its measurements. This work may be extendable to other means of distance measurement such as sonar, radar, or multi-camera image processing.

APPENDIX A
Monte Carlo Plots

The following plots show the *Monte Carlo* analysis results for the first nine state variables covering position, velocity, and attitude state estimation. They prove the Extended Kalman Filter is behaving nominally, as the mean estimation error is zero and the error-variance and mean-covariance values match closely.

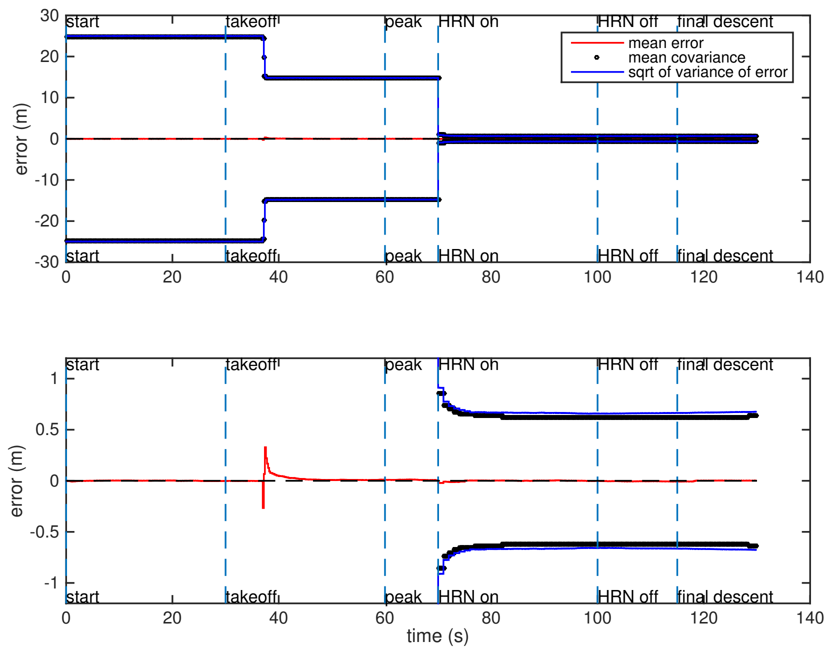

Figure A.1: *Monte Carlo* inertial X position mean error, error variance, and mean covariance

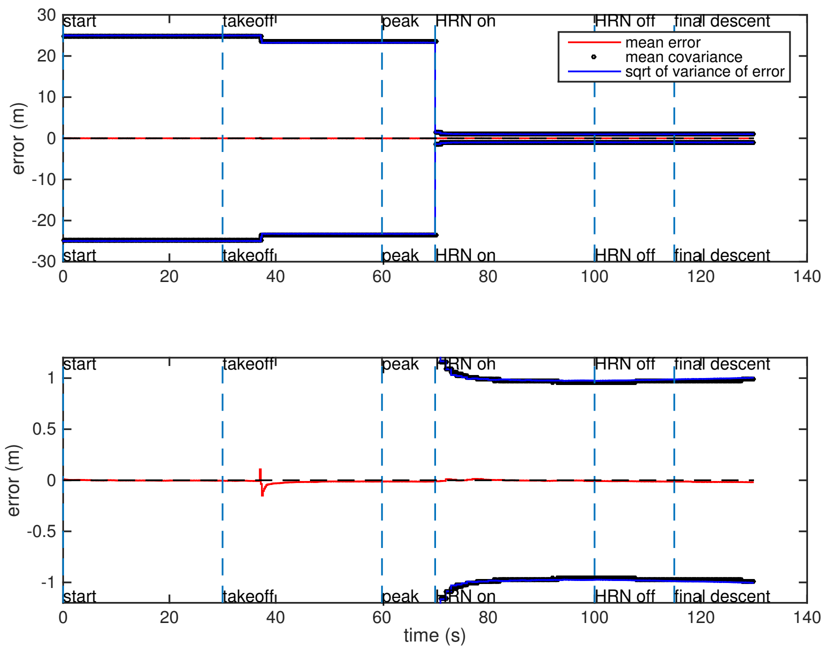

Figure A.2: *Monte Carlo* inertial Y position mean error, error variance, and mean covariance

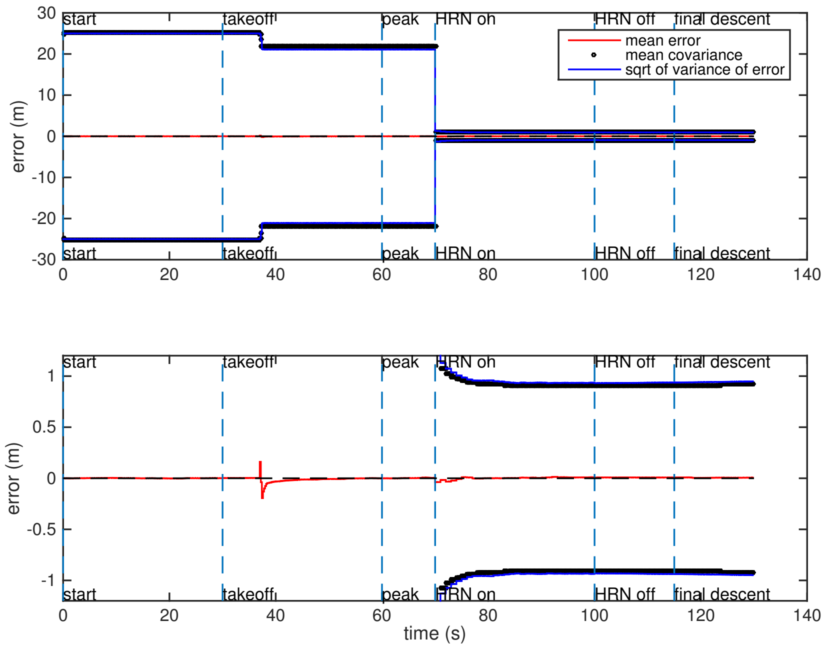

Figure A.3: *Monte Carlo* inertial Z position mean error, error variance, and mean covariance

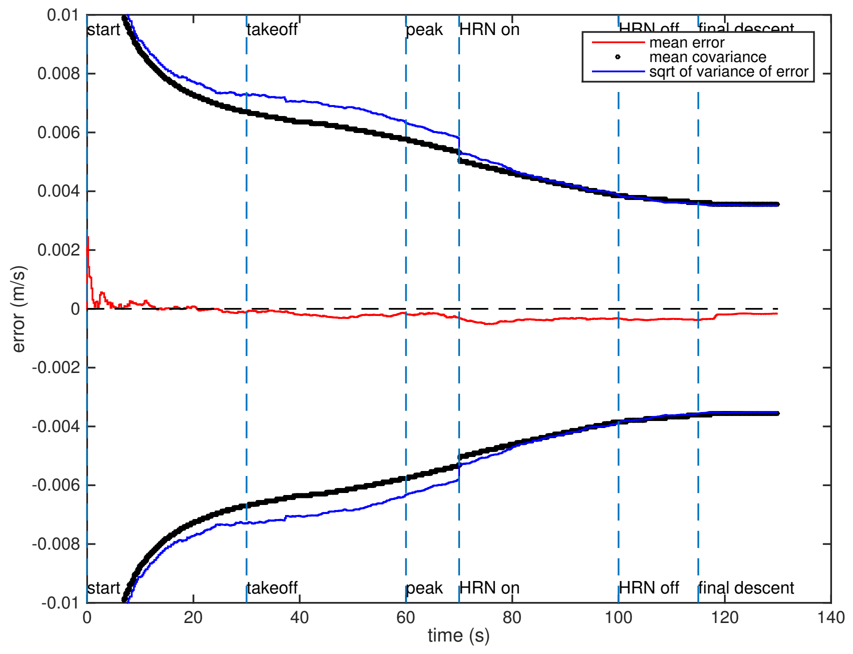

Figure A.4: *Monte Carlo* inertial X velocity mean error, error variance, and mean covariance

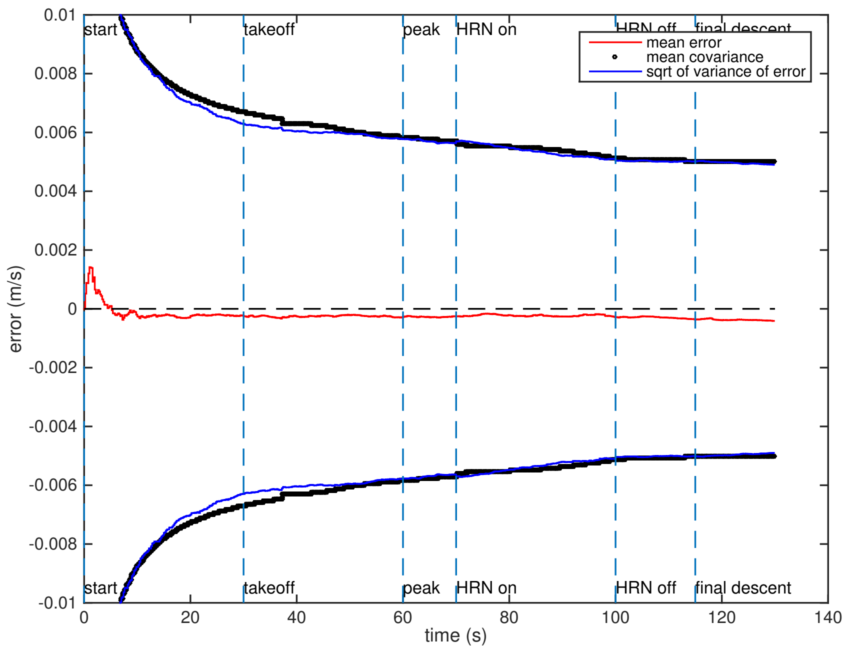

Figure A.5: *Monte Carlo* inertial Y velocity mean error, error variance, and mean covariance

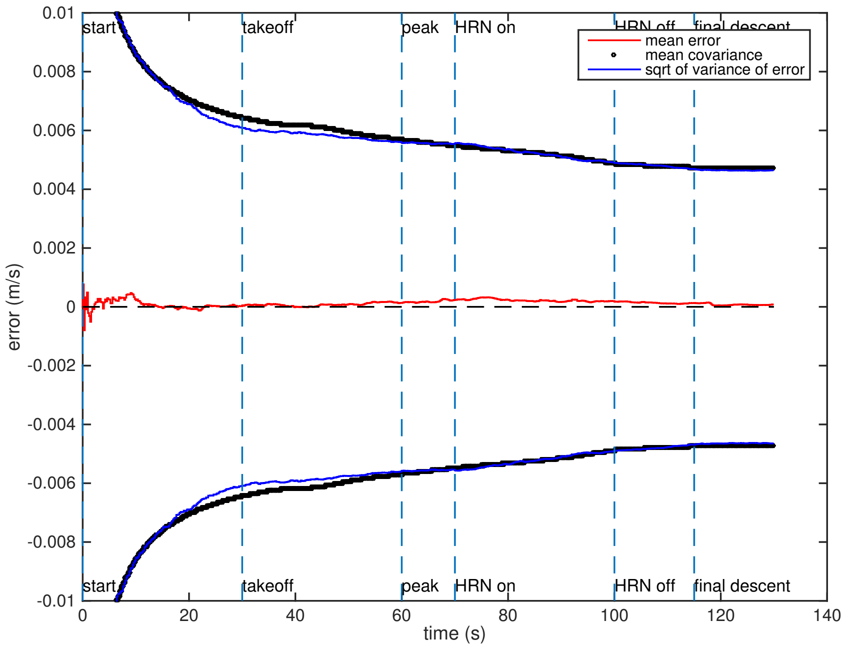

Figure A.6: *Monte Carlo* inertial Z velocity mean error, error variance, and mean covariance

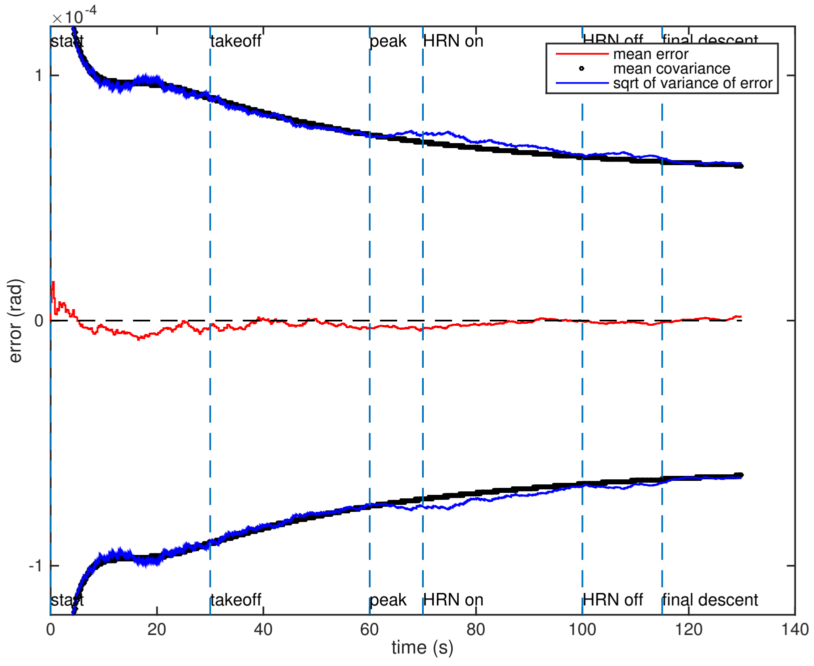

Figure A.7: *Monte Carlo* body-fixed X attitude mean error, error variance, and mean covariance

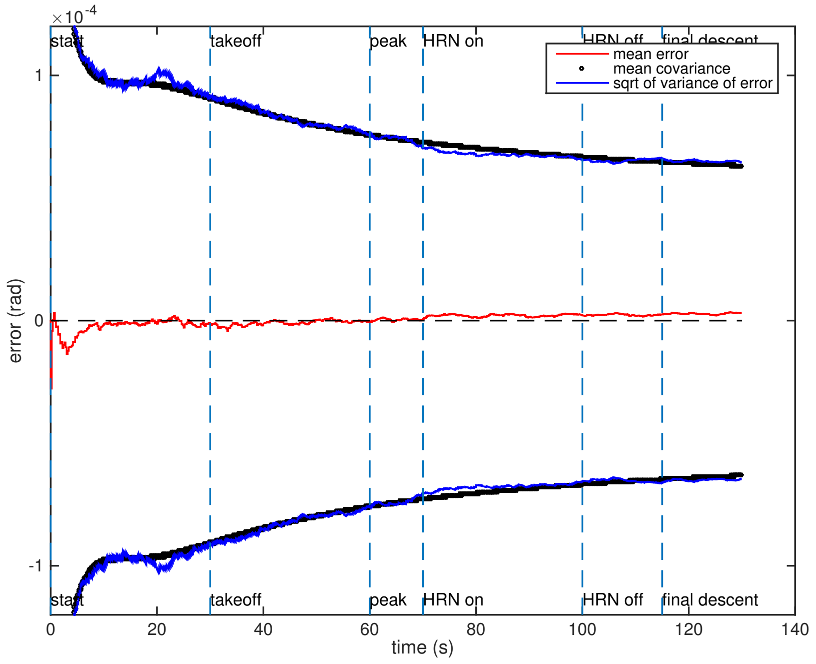

Figure A.8: *Monte Carlo* body-fixed Y attitude mean error, error variance, and mean covariance

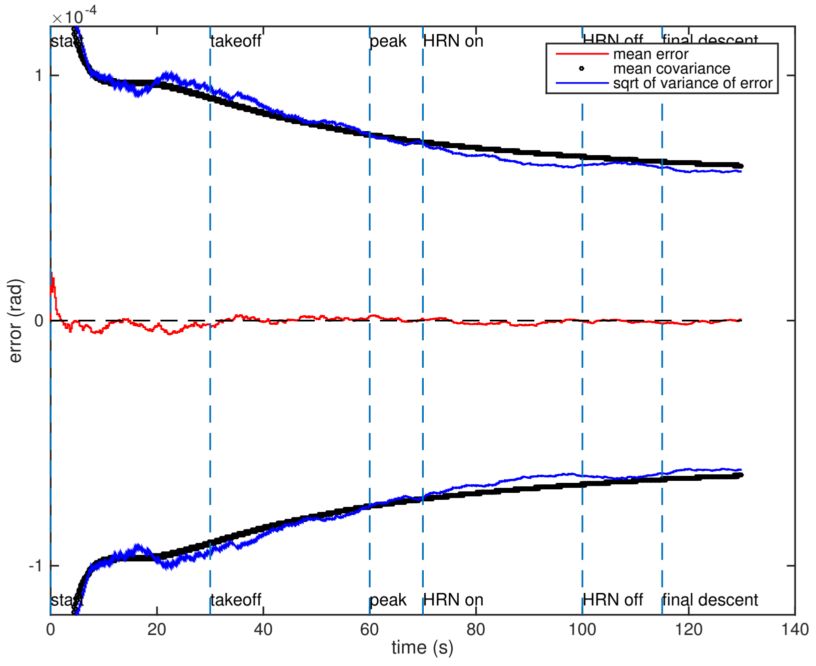

Figure A.9: *Monte Carlo* body-fixed Z attitude mean error, error variance, and mean covariance

APPENDIX B
Error Budget Tables

The full error budget tables that were created during the error budget analysis are shown here. The first set is for the simulation time immediately preceding THRN sensor activation, and the second set covers the simulation time right before final descent begins. Due to the length of the state vector and number of error groups, the tables are broken into six parts for each simulation time.

Table B.1: Error budget for position, velocity, and attitude states- pre-THRN

	$\mathbf{r}_x$	$\mathbf{r}_y$	$\mathbf{r}_z$	$\mathbf{v}_x$	$\mathbf{v}_y$	$\mathbf{v}_z$	$\boldsymbol{\alpha}_x$	$\boldsymbol{\alpha}_y$	$\boldsymbol{\alpha}_z$
1	14.82	23.43	21.93	0.007623	0.003409	0.004437	6.393e-07	6.085e-07	2.052e-08
2	0.2024	0.4208	0.192	0.0002576	0.0003685	0.0003071	7.808e-08	8.371e-08	1.191e-08
3	0.000402	0.0005441	0.0005091	9.31e-06	8.346e-06	4.417e-06	1.064e-06	1.059e-06	1.052e-06
4	0.001078	0.00322	0.002369	0.0003036	0.0003305	0.0003249	2.028e-07	2.003e-07	2.257e-08
5	0.0004542	0.0005985	0.0004297	4.819e-06	3.635e-05	3.686e-05	6.266e-06	6.262e-06	6.286e-06
6	0	0	0	0	0	0	0	0	0
7	0.2068	0.3176	0.2975	0.00407	0.004826	0.004672	1.875e-07	6.299e-08	9.507e-09
8	0	0	0	0	0	0	0	0	0
9	0.08041	0.03457	0.0479	3.108e-05	6.584e-06	1.705e-05	1.032e-09	1.758e-09	5.229e-11
10	0.008656	0.01163	0.01025	0.0001118	0.000198	0.0001611	4.987e-05	4.977e-05	4.976e-05
11	0.0003785	0.00172	0.0006518	0.0002009	0.0003015	0.000281	1.978e-07	1.975e-07	2.24e-08
12	0.0004512	0.001659	0.0006675	0.0001864	0.0003043	0.0002866	1.982e-07	1.974e-07	2.237e-08
13	0.0009489	0.002681	0.002225	0.0002183	9.946e-05	0.0001325	2.805e-08	2.022e-09	1.882e-10
14	1.039e-06	4.84e-07	1.735e-06	1.263e-07	5.157e-07	2.151e-07	4.323e-09	2.17e-08	2.066e-08
15	6.441e-07	1.027e-06	2.326e-06	3.09e-07	4.398e-07	2.124e-07	7.393e-09	2.133e-08	2.014e-08
16	5.901e-07	1.581e-06	4.694e-07	2.536e-07	2.348e-07	4.552e-07	2.055e-08	6.872e-10	6.055e-09
17	0	0	0	0	0	0	0	0	0
18	0.1137	0.1737	0.1629	0.002228	0.002626	0.002553	3.048e-06	3.027e-06	3.371e-07
19	0.07323	0.02875	0.04424	0.00206	0.0008911	0.00123	7.49e-07	2.881e-07	1.379e-08
20	0.005774	0.007764	0.006436	8.299e-05	0.0003016	0.0002956	5.344e-05	5.348e-05	5.363e-05
root sum of squares	14.83	23.44	21.93	0.009175	0.00657	0.007073	7.344e-05	7.339e-05	7.344e-05
full run root cov	14.83	23.44	21.93	0.005418	0.005726	0.005521	7.344e-05	7.339e-05	7.344e-05
difference	-2.556e-07	-4.088e-05	-8.502e-06	-0.003757	-0.0008444	-0.001552	-2.595e-09	-2.359e-09	-2.797e-11

Table B.2: Error budget for accelerometer and gyroscope bias states- pre-THRN

	$\mathbf{b}_{acc,x}$	$\mathbf{b}_{acc,y}$	$\mathbf{b}_{acc,z}$	$\mathbf{b}_{gyro,x}$	$\mathbf{b}_{gyro,y}$	$\mathbf{b}_{gyro,z}$
1	2.634e-08	1.691e-08	3.107e-07	5.182e-09	5.95e-09	1.847e-10
2	3.368e-08	3.328e-08	3.928e-08	1.346e-09	1.433e-09	1.128e-10
3	1.119e-09	1.092e-09	3.3e-10	6.298e-08	6.301e-08	6.321e-08
4	9.948e-06	9.948e-06	9.945e-06	4.609e-09	4.596e-09	5.488e-11
5	4.097e-09	4.112e-09	1.075e-10	1.766e-07	1.767e-07	1.773e-07
6	0	0	0	0	0	0
7	1.325e-08	9.472e-09	8.373e-08	2.579e-09	1.486e-09	6.214e-11
8	0	0	0	0	0	0
9	1.023e-10	6.219e-11	1.212e-09	7.459e-12	1.837e-11	3.906e-13
10	5.277e-09	4.446e-09	4.406e-09	2.887e-08	2.062e-08	3.524e-08
11	5.098e-08	5.099e-08	8.62e-10	4.545e-09	4.53e-09	4.629e-11
12	5.096e-08	5.094e-08	4.965e-10	4.537e-09	4.525e-09	4.634e-11
13	3.449e-10	5.412e-10	5.445e-08	6.47e-11	6.473e-11	1.517e-11
14	5.737e-11	2.335e-11	3.169e-12	5.862e-10	4.192e-10	7.116e-10
15	5.703e-11	2.246e-11	1.165e-12	5.776e-10	4.293e-10	7.129e-10
16	2.141e-11	5.686e-11	2.574e-12	4.242e-10	5.811e-10	2.735e-11
17	0	0	0	0	0	0
18	7.109e-07	7.109e-07	7.174e-07	7.317e-08	7.298e-08	3.618e-09
19	6.646e-09	3.586e-09	1.528e-07	6.296e-09	2.29e-09	1.608e-10
20	3.297e-08	3.294e-08	2.408e-09	1.347e-06	1.348e-06	1.352e-06
root sum of squares	9.974e-06	9.974e-06	9.977e-06	1.363e-06	1.363e-06	1.365e-06
full run root cov	9.974e-06	9.974e-06	9.972e-06	1.363e-06	1.363e-06	1.365e-06
difference	-3.465e-11	-1.406e-11	-4.934e-09	-1.628e-11	-1.265e-11	-3.278e-12

Table B.3: Error budget for THRN and ILP bias states- pre-THRN

	$\mathbf{b}_{hrn,x}$	$\mathbf{b}_{hrn,y}$	$\mathbf{b}_{hrn,z}$	$\mathbf{b}_{ilp,x}$	$\mathbf{b}_{ilp,y}$	$\mathbf{b}_{ilp,z}$
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0.1	0.1	0.1	0	0	0
7	0	0	0	0	0	0
8	0	0	0	1	1	1
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0
12	0	0	0	0	0	0
13	0	0	0	0	0	0
14	0	0	0	0	0	0
15	0	0	0	0	0	0
16	0	0	0	0	0	0
17	0	0	0	0	0	0
18	0	0	0	0	0	0
19	0	0	0	0	0	0
20	0	0	0	0	0	0
root sum of squares	0.1	0.1	0.1	1	1	1
full run root cov	0.1	0.1	0.1	1	1	1
difference	0	0	0	0	0	0

Table B.4: Error budget for velocimeter, altimeter, and star camera bias states- pre-THRN

	$\mathbf{b}_{vel,x}$	$\mathbf{b}_{vel,y}$	$\mathbf{b}_{vel,z}$	b_{alt}	$\mathbf{b}_{sc,x}$	$\mathbf{b}_{sc,y}$	$\mathbf{b}_{sc,z}$
1	4.744e-05	7.954e-05	0.007123	0.0001332	3.605e-07	3.11e-07	8.88e-09
2	0.0002496	0.0002485	0.0002068	2.56e-06	2.804e-08	2.686e-08	1.113e-08
3	6.055e-08	2.683e-08	2.051e-07	6.87e-08	2.486e-06	2.483e-06	2.481e-06
4	8.271e-06	5.343e-06	4.031e-05	4.024e-07	4.049e-08	3.364e-08	1.607e-08
5	7.145e-08	3.972e-08	1.334e-06	1.431e-07	1.429e-07	1.089e-07	1.807e-07
6	0	0	0	0	0	0	0
7	0.004982	0.004976	0.003839	2.177e-05	7.769e-08	8.802e-09	6.406e-09
8	0	0	0	0	0	0	0
9	1.976e-07	3.056e-07	2.736e-05	0.1	6.101e-10	8.709e-10	2.593e-11
10	4.705e-07	3.803e-07	7.808e-06	1.812e-06	4.984e-05	4.976e-05	4.975e-05
11	9.784e-07	6.697e-07	7.48e-06	2.539e-08	3.306e-08	3.325e-08	1.59e-08
12	9.418e-07	8.027e-07	8.101e-06	7.868e-08	3.358e-08	3.326e-08	1.588e-08
13	7.995e-06	5.17e-06	3.962e-05	4.759e-07	2.206e-08	3.615e-10	1.194e-10
14	9.214e-10	2.454e-10	1.117e-08	7.022e-10	9.644e-10	2.256e-09	2.544e-09
15	5.039e-10	2.049e-09	2.054e-08	2.522e-09	8.966e-10	2.159e-09	1.676e-09
16	6.524e-10	2.157e-09	1.319e-08	2.582e-09	1.065e-10	4.108e-10	1.085e-09
17	0	0	0	0	0	0	0
18	0.0001609	0.000243	0.0006623	1.195e-05	5.288e-07	4.635e-07	2.247e-07
19	1.134e-05	2.206e-05	0.001969	0.0002619	4.149e-07	1.634e-07	4.485e-09
20	6.316e-07	4.462e-07	1.16e-05	1.422e-06	1.11e-06	2.298e-06	2.5e-06
root sum of squares	0.004991	0.004988	0.008357	0.1	4.992e-05	4.988e-05	4.987e-05
full run root cov	0.004991	0.004988	0.004411	0.1	4.992e-05	4.988e-05	4.987e-05
difference	-2.077e-07	-6.412e-07	-0.003946	-3.58e-07	-1.251e-09	-9.142e-10	-3.929e-11

Table B.5: Error budget for accelerometer misalignment, nonorthogonality, and scale factor bias states- pre-THRN

	$\mathbf{m}_{acc,x}$	$\mathbf{m}_{acc,y}$	$\mathbf{m}_{acc,z}$	$\mathbf{n}_{acc,x}$	$\mathbf{n}_{acc,y}$	$\mathbf{n}_{acc,z}$	$\mathbf{s}_{acc,x}$	$\mathbf{s}_{acc,y}$	$\mathbf{s}_{acc,z}$
1	1.386e-09	2.344e-09	3.164e-11	1.863e-09	2.765e-09	6.241e-11	2.857e-11	3.505e-11	3.279e-08
2	3.293e-09	3.329e-09	1.856e-11	3.291e-09	3.334e-09	1.894e-11	8.303e-12	1.631e-11	3.898e-09
3	1.079e-10	1.101e-10	1.951e-12	1.07e-10	1.1e-10	1.93e-12	1.023e-12	1.621e-12	3.219e-11
4	5.1e-09	5.098e-09	6.353e-11	5.094e-09	5.096e-09	6.33e-11	3.279e-11	5.427e-11	5.445e-09
5	4.052e-10	4.036e-10	6.625e-12	4.044e-10	4.031e-10	6.61e-12	3.45e-12	5.655e-12	1.144e-11
6	0	0	0	0	0	0	0	0	0
7	8.615e-10	1.257e-09	1.67e-11	9.812e-10	1.316e-09	2.902e-11	1.344e-11	1.513e-11	8.396e-09
8	0	0	0	0	0	0	0	0	0
9	5.273e-12	9.252e-12	1.239e-13	6.718e-12	1.062e-11	2.231e-13	1.053e-13	1.172e-13	1.273e-10
10	4.312e-10	5.222e-10	2.201e-12	4.467e-10	5.224e-10	4.36e-12	1.727e-12	2.721e-12	4.339e-10
11	9.949e-07	9.95e-07	1e-06	5.042e-09	5.043e-09	6.102e-11	3.154e-11	5.234e-11	7.771e-11
12	5.042e-09	5.043e-09	6.129e-11	9.95e-07	9.95e-07	1e-06	3.154e-11	5.235e-11	5.36e-11
13	8.104e-11	5.655e-11	9.786e-13	7.216e-11	3.708e-11	3.014e-12	1e-06	1e-06	9.946e-07
14	2.246e-12	5.536e-12	1.392e-13	2.24e-12	5.541e-12	8.088e-14	6.911e-14	4.771e-14	3.102e-13
15	2.155e-12	5.505e-12	7.953e-14	2.153e-12	5.507e-12	1.38e-13	6.872e-14	4.742e-14	1.162e-13
16	5.486e-12	2.052e-12	8.27e-14	5.492e-12	2.05e-12	8.175e-14	2.901e-14	1.086e-13	2.517e-13
17	0	0	0	0	0	0	0	0	0
18	7.036e-08	7.032e-08	1.228e-09	7.028e-08	7.03e-08	1.224e-09	6.788e-10	1.021e-09	7.097e-08
19	4.607e-10	7.067e-10	9.152e-12	3.739e-10	6.261e-10	1.49e-11	6.004e-12	1.381e-11	1.518e-08
20	3.248e-09	3.25e-09	5.235e-11	3.243e-09	3.246e-09	5.22e-11	2.725e-11	4.46e-11	2.357e-10
root sum of squares	9.975e-07	9.975e-07	1e-06	9.975e-07	9.975e-07	1e-06	1e-06	1e-06	9.978e-07
full run root cov	9.975e-07	9.975e-07	1e-06	9.975e-07	9.975e-07	1e-06	1e-06	1e-06	9.973e-07
difference	-9.39e-13	-2.745e-12	-4.979e-16	-1.7e-12	-3.81e-12	-1.94e-15	-4.124e-16	-5.962e-16	-5.489e-10

Table B.6: Error budget for gyroscope misalignment, nonorthogonality, and scale factor bias states- pre-THRN

	$\mathbf{m}_{gyro,x}$	$\mathbf{m}_{gyro,y}$	$\mathbf{m}_{gyro,z}$	$\mathbf{n}_{gyro,x}$	$\mathbf{n}_{gyro,y}$	$\mathbf{n}_{gyro,z}$	$\mathbf{s}_{gyro,x}$	$\mathbf{s}_{gyro,y}$	$\mathbf{s}_{gyro,z}$
1	1.256e-12	1.392e-12	5.202e-12	1.374e-12	2.663e-12	1.134e-11	1.169e-11	2.542e-12	1.43e-13
2	2.85e-13	2.245e-13	1.691e-12	2.063e-13	2.387e-13	1.807e-12	1.572e-12	5.791e-13	2.232e-14
3	9.713e-12	6.546e-12	1.146e-11	8.96e-12	7.077e-12	1.168e-11	6.657e-12	9.328e-12	1.288e-12
4	9.457e-13	8.696e-13	6.098e-12	6.291e-13	7.298e-13	6.085e-12	5.716e-12	2.156e-12	9.136e-14
5	5.804e-11	4.397e-11	7.371e-11	5.795e-11	4.465e-11	7.347e-11	4.503e-11	5.824e-11	2.451e-12
6	0	0	0	0	0	0	0	0	0
7	4.712e-13	1.811e-13	2.159e-12	4.354e-13	4.7e-13	4.022e-12	2.377e-12	1.078e-12	4.443e-14
8	0	0	0	0	0	0	0	0	0
9	2.754e-15	2.19e-15	1.494e-14	3.253e-15	6.996e-15	2.032e-14	2.13e-14	4.765e-15	5.328e-16
10	1.904e-11	4.509e-11	5.095e-11	1.821e-11	4.317e-11	3.337e-11	1.635e-11	1.549e-11	5.878e-12
11	9.179e-13	8.362e-13	5.874e-12	6.061e-13	7e-13	5.868e-12	5.509e-12	2.066e-12	8.789e-14
12	9.191e-13	8.388e-13	5.875e-12	6.078e-13	7.04e-13	5.87e-12	5.515e-12	2.064e-12	8.785e-14
13	6.15e-14	9.99e-14	3e-13	5.89e-14	1.003e-13	1.115e-13	2.806e-13	5.423e-14	1.68e-15
14	1e-06	1e-06	1e-06	3.623e-13	8.657e-13	6.741e-13	3.352e-13	3.135e-13	1.181e-13
15	3.642e-13	8.653e-13	6.737e-13	1e-06	1e-06	1e-06	3.372e-13	3.138e-13	1.181e-13
16	3.469e-14	1.648e-13	4.43e-13	3.45e-14	1.652e-13	4.446e-13	1e-06	1e-06	1e-06
17	0	0	0	0	0	0	0	0	0
18	1.81e-11	2.141e-11	1.23e-10	1.19e-11	2.144e-11	1.232e-10	1.137e-10	4.852e-11	3.005e-12
19	9.498e-13	1.346e-12	2.611e-12	9.27e-13	1.412e-12	9.633e-12	9.704e-12	2.092e-12	3.875e-14
20	4.425e-10	9.206e-10	1.001e-09	4.423e-10	9.228e-10	1.001e-09	9.118e-10	4.171e-10	1.478e-10
root sum of squares	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06
full run root cov	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06
difference	-2.265e-17	5.872e-18	-2.692e-18	-3.207e-18	8.238e-18	-7.472e-17	-2.427e-17	-2.596e-17	-4.686e-18

Table B.7: Error budget for position, velocity, and attitude states - final descent

	$\mathbf{r}_x$	$\mathbf{r}_y$	$\mathbf{r}_z$	$\mathbf{v}_x$	$\mathbf{v}_y$	$\mathbf{v}_z$	$\boldsymbol{\alpha}_x$	$\boldsymbol{\alpha}_y$	$\boldsymbol{\alpha}_z$
1	0.08127	0.05015	0.05686	0.00326	0.001426	0.001913	5.685e-06	2.786e-06	3.052e-07
2	0.00315	0.006596	0.005682	0.000117	0.0002479	0.0002126	2.102e-07	1.383e-07	1.145e-08
3	0.0001376	0.0001125	9.108e-05	1.369e-05	1.459e-05	9.738e-06	1.608e-06	1.603e-06	1.608e-06
4	0.007148	0.008861	0.008356	0.0004342	0.0004899	0.0004685	3.062e-07	2.665e-07	1.362e-07
5	1.136e-05	1.773e-05	0.0001029	1.586e-05	9.272e-06	2.315e-06	2.064e-06	2.065e-06	2.069e-06
6	0.05921	0.09372	0.08719	2.646e-05	4.84e-06	5.362e-05	8.134e-08	3.314e-08	1.641e-09
7	0.06896	0.1046	0.09759	0.002569	0.003991	0.003714	3.071e-06	1.577e-06	1.248e-07
8	0.5907	0.9371	0.8757	7.809e-05	4.777e-05	2.144e-05	5.643e-07	2.228e-07	1.632e-08
9	0.07966	0.0341	0.04747	1.074e-05	2.239e-06	5.014e-06	7.857e-08	3.317e-08	2.783e-09
10	0.001313	0.0004099	0.001026	0.0003992	0.0002744	0.0001718	4.942e-05	4.928e-05	4.939e-05
11	0.005194	0.008182	0.007761	0.0002778	0.0004497	0.0004271	2.684e-07	2.575e-07	1.303e-07
12	0.005043	0.008258	0.007728	0.0002872	0.0004536	0.0004128	2.83e-07	2.59e-07	1.299e-07
13	0.004587	0.002294	0.002186	0.0003105	0.0001439	0.0001706	1.08e-07	1.084e-08	1.358e-08
14	1.694e-06	1.507e-05	5.716e-06	1.977e-07	1.121e-06	4.771e-07	3.69e-08	4.751e-08	5.551e-08
15	1.618e-06	1.541e-05	6.931e-06	2.004e-07	1.163e-06	5.191e-07	3.436e-08	4.833e-08	5.649e-08
16	4.571e-06	4.513e-06	1.32e-05	3.817e-07	2.487e-07	8.036e-07	4.446e-08	3.209e-08	1.561e-08
17	0.09723	0.1537	0.1438	0.00117	0.001808	0.001681	1.389e-06	8.618e-07	2.456e-07
18	0.0296	0.04385	0.04213	0.001111	0.00168	0.001614	2.408e-06	2.047e-06	1.177e-06
19	0.07108	0.03048	0.04207	0.001479	0.0006406	0.0008961	3.794e-06	1.866e-06	1.447e-07
20	0.001811	0.002255	0.0006724	0.0003223	0.0002254	0.000116	4.125e-05	4.159e-05	4.167e-05
root sum of squares	0.621	0.9635	0.9022	0.004769	0.005032	0.004938	6.493e-05	6.468e-05	6.469e-05
full run root cov	0.6161	0.9629	0.901	0.003486	0.004827	0.004554	6.468e-05	6.462e-05	6.469e-05
difference	-0.004928	-0.000613	-0.00115	-0.001284	-0.0002049	-0.0003842	-2.471e-07	-5.94e-08	-7.013e-10

Table B.8: Error budget for accelerometer and gyroscope bias states - final descent

	$\mathbf{b}_{acc,x}$	$\mathbf{b}_{acc,y}$	$\mathbf{b}_{acc,z}$	$\mathbf{b}_{gyro,x}$	$\mathbf{b}_{gyro,y}$	$\mathbf{b}_{gyro,z}$
1	5.133e-08	6.799e-09	1.054e-06	1.493e-09	2.062e-09	2.913e-09
2	5.864e-08	5.794e-08	7.081e-08	3.622e-10	3.933e-10	4.376e-10
3	3.136e-09	1.484e-09	5.116e-09	2.204e-08	2.212e-08	2.208e-08
4	9.754e-06	9.754e-06	9.723e-06	3.368e-09	2.248e-09	3.659e-09
5	3.573e-09	4.106e-09	2.226e-09	3.02e-08	3.023e-08	3.024e-08
6	7.934e-10	6.157e-09	4.016e-09	3.389e-10	1.131e-10	1.689e-11
7	7.081e-08	7.243e-08	2.572e-07	7.656e-09	4.523e-09	1.197e-09
8	8.031e-10	7.212e-10	6.314e-09	1.564e-09	6.078e-10	1.349e-10
9	1.282e-10	8.138e-11	3.594e-09	1.521e-10	5.337e-11	2.475e-11
10	3.264e-08	2.953e-08	5.125e-08	1.16e-08	1.348e-08	1.764e-08
11	2.385e-07	2.389e-07	7.478e-09	1.464e-09	1.412e-09	3.523e-09
12	2.384e-07	2.381e-07	8.116e-09	1.375e-09	1.339e-09	3.521e-09
13	4.545e-09	4.925e-09	2.672e-07	2.923e-09	1.682e-09	1.021e-10
14	4.313e-10	2.02e-10	7.007e-11	2.412e-10	2.66e-10	3.627e-10
15	4.391e-10	1.654e-10	1.834e-10	2.372e-10	2.656e-10	3.53e-10
16	1.477e-10	3.899e-10	2.414e-10	2.69e-10	2.37e-10	4.508e-11
17	1.981e-07	1.971e-07	1.219e-07	7.176e-09	6.243e-09	3.519e-09
18	1.496e-06	1.497e-06	1.521e-06	2.864e-08	2.12e-08	2.669e-08
19	1.462e-08	1.515e-08	4.432e-07	8.356e-09	4.286e-09	1.132e-09
20	5.363e-08	3.318e-08	6.012e-08	6.143e-07	6.154e-07	6.151e-07
root sum of squares	9.876e-06	9.876e-06	9.916e-06	6.164e-07	6.171e-07	6.171e-07
full run root cov	9.876e-06	9.876e-06	9.861e-06	6.164e-07	6.171e-07	6.171e-07
difference	-1.337e-10	-3.35e-12	-5.539e-08	-4.933e-12	-5.411e-12	-1.106e-11

Table B.9: Error budget for THRN and ILP bias states - final descent

	$\mathbf{b}_{hrn,x}$	$\mathbf{b}_{hrn,y}$	$\mathbf{b}_{hrn,z}$	$\mathbf{b}_{ilp,x}$	$\mathbf{b}_{ilp,y}$	$\mathbf{b}_{ilp,z}$
1	0.0004166	0.00103	9.905e-05	0.02477	0.03749	0.03533
2	5.579e-06	1.7e-05	5.619e-05	0.000334	0.0008407	0.0003274
3	4.343e-07	3.118e-06	3.629e-06	0.000332	0.0001448	0.0001967
4	6.563e-06	3.835e-05	6.284e-05	0.00051	0.0002271	0.0003141
5	2.652e-07	1.174e-06	3.009e-06	0.0001349	5.901e-05	8.078e-05
6	0.1	0.09989	0.09895	0.07651	0.03351	0.04573
7	7.232e-05	0.0001661	0.0009788	0.006989	0.003029	0.003945
8	0.0001822	2.53e-05	0.009521	0.5956	0.9347	0.8759
9	1.767e-05	5.548e-06	0.0009502	0.07653	0.03352	0.04576
10	1.021e-05	6.503e-05	9.492e-05	0.007009	0.003059	0.004161
11	4.271e-06	2.466e-06	5.94e-05	4.708e-05	5.331e-05	5.536e-05
12	3.813e-06	6.863e-06	5.905e-05	7.058e-05	6.502e-05	6.716e-05
13	4.787e-06	3.615e-05	1.149e-05	0.0004822	0.000208	0.0002929
14	6.953e-10	1.336e-08	1.054e-08	1.07e-06	4.196e-07	6.415e-07
15	4.694e-09	3.783e-08	3.235e-08	2.504e-06	1.044e-06	1.467e-06
16	6.325e-09	7.727e-08	3.687e-09	3.451e-06	1.503e-06	2.015e-06
17	0.0004441	0.00323	0.003242	0.1198	0.05248	0.07164
18	3.119e-05	7.318e-05	0.0004215	0.003026	0.00132	0.001711
19	4.855e-05	0.000426	0.0005343	0.04314	0.01891	0.02584
20	6.25e-06	3.333e-05	6.686e-05	0.003634	0.001588	0.002164
root sum of squares	0.1	0.09995	0.09948	0.6192	0.9383	0.8824
full run root cov	0.1	0.09995	0.09948	0.6192	0.9383	0.8823
difference	-7.89e-08	-4.544e-06	-7.067e-08	-3.559e-05	-5.167e-06	-1.197e-05

Table B.10: Error budget for velocimeter, altimeter, and star camera bias states - final descent

	$\mathbf{b}_{vel,x}$	$\mathbf{b}_{vel,y}$	$\mathbf{b}_{vel,z}$	b_{alt}	$\mathbf{b}_{sc,x}$	$\mathbf{b}_{sc,y}$	$\mathbf{b}_{sc,z}$
1	0.0001012	7.842e-05	0.003544	5.767e-05	4.895e-06	2.561e-06	5.245e-08
2	0.0002148	0.0002123	6.486e-05	6.228e-06	1.704e-07	1.038e-07	2.118e-08
3	3.112e-07	2.335e-07	1.257e-05	4.048e-06	2.469e-06	2.465e-06	2.467e-06
4	3.867e-05	3.976e-05	0.0001302	5.945e-06	2.995e-07	1.804e-07	8.983e-08
5	6.664e-07	4.675e-07	4.122e-06	1.694e-06	5.885e-08	5.483e-08	8.111e-08
6	6.51e-06	4.854e-05	9.131e-06	0.0009508	5.262e-08	2.371e-08	1.404e-09
7	0.00437	0.004344	0.001326	0.0001035	2.223e-06	1.186e-06	4.592e-08
8	3.047e-06	5.52e-06	4.209e-05	0.00953	4.009e-07	1.675e-07	7.674e-09
9	2.716e-07	5.122e-07	9.098e-06	0.09905	5.907e-08	2.673e-08	8.859e-10
10	4.61e-06	5.857e-06	0.0002556	8.597e-05	4.948e-05	4.936e-05	4.943e-05
11	3.4e-05	3.475e-05	8.427e-06	8.008e-07	1.545e-07	1.538e-07	8.672e-08
12	3.33e-05	3.431e-05	7.563e-06	1.128e-06	1.712e-07	1.587e-07	8.63e-08
13	1.578e-05	1.632e-05	0.0001261	5.492e-06	2.377e-07	7.676e-08	8.912e-09
14	2.836e-08	1.152e-08	4.364e-08	1.294e-08	2.151e-09	8.166e-09	9.004e-09
15	2.777e-08	5.55e-08	9.881e-08	2.834e-08	2.043e-09	7.977e-09	7.284e-09
16	8.851e-09	5.546e-08	8.468e-08	4.028e-08	5.303e-10	1.025e-09	2.719e-09
17	0.001612	0.0016	0.000268	0.001489	9.849e-07	5.435e-07	1.504e-07
18	0.0003103	0.0004813	0.001528	4.484e-05	1.464e-06	1.094e-06	5.518e-07
19	1.853e-05	6.306e-06	0.001562	0.0006085	2.914e-06	1.505e-06	3.99e-08
20	1.789e-05	9.003e-06	0.0001266	4.499e-05	1.562e-06	4.467e-06	4.679e-06
root sum of squares	0.004675	0.004661	0.004391	0.09952	4.998e-05	4.975e-05	4.971e-05
full run root cov	0.004675	0.00466	0.0026	0.09952	4.974e-05	4.968e-05	4.971e-05
difference	-4.451e-07	-5.125e-08	-0.001792	-5.78e-07	-2.382e-07	-6.54e-08	-8.17e-11

Table B.11: Error budget for accelerometer misalignment, nonorthogonality, and scale factor bias states - final descent

	$\mathbf{m}_{acc,x}$	$\mathbf{m}_{acc,y}$	$\mathbf{m}_{acc,z}$	$\mathbf{n}_{acc,x}$	$\mathbf{n}_{acc,y}$	$\mathbf{n}_{acc,z}$	$\mathbf{s}_{acc,x}$	$\mathbf{s}_{acc,y}$	$\mathbf{s}_{acc,z}$
1	5.257e-09	1.171e-09	9.866e-11	4.07e-09	8.624e-09	3.5e-10	1.571e-10	3.869e-10	9.792e-08
2	5.672e-09	5.674e-09	1.123e-10	5.656e-09	5.785e-09	1.152e-10	6.941e-11	8.94e-11	6.903e-09
3	1.327e-10	3.011e-10	7.643e-12	1.525e-10	3.043e-10	4.317e-12	6.05e-12	2.396e-12	4.914e-10
4	2.389e-08	2.386e-08	6.562e-10	2.381e-08	2.384e-08	6.539e-10	4.545e-10	4.743e-10	2.672e-08
5	3.977e-10	3.469e-10	9.778e-12	3.995e-10	3.486e-10	1.199e-11	6.1e-12	9.046e-12	2.145e-10
6	5.916e-10	7.369e-11	1.452e-11	5.901e-10	8.084e-11	2.07e-11	2.345e-12	2.617e-11	3.815e-10
7	7.138e-09	6.807e-09	3.867e-10	6.863e-09	6.81e-09	3.823e-10	2.087e-10	3.344e-10	2.496e-08
8	6.538e-11	6.367e-11	1.436e-12	7.306e-11	9.21e-11	5.206e-12	1.733e-12	7.065e-12	6.012e-10
9	2.654e-11	1.274e-12	4.763e-13	9.222e-12	2.539e-11	8.604e-13	4.552e-13	8.141e-13	3.331e-10
10	2.654e-09	3.114e-09	7.783e-11	3.037e-09	3.203e-09	8.699e-11	6.045e-11	5.767e-11	4.916e-09
11	9.768e-07	9.769e-07	1e-06	2.306e-08	2.307e-08	6.275e-10	4.369e-10	4.526e-10	6.893e-10
12	2.306e-08	2.307e-08	6.299e-10	9.77e-07	9.769e-07	1e-06	4.37e-10	4.53e-10	8.087e-10
13	6.261e-10	6.962e-10	2.053e-11	7.882e-10	6.5e-10	3.683e-11	1e-06	1e-06	9.742e-07
14	1.952e-11	4.146e-11	1.1e-12	1.943e-11	4.165e-11	4.913e-13	8.075e-13	3.178e-13	6.754e-12
15	1.576e-11	4.202e-11	5.303e-13	1.605e-11	4.261e-11	1.075e-12	8.248e-13	3.706e-13	1.768e-11
16	3.738e-11	1.416e-11	8.232e-13	3.774e-11	1.428e-11	8.4e-13	2.931e-13	8.269e-13	2.327e-11
17	1.893e-08	1.899e-08	1.039e-09	1.891e-08	1.902e-08	1.038e-09	5.662e-10	8.653e-10	1.16e-08
18	1.452e-07	1.449e-07	5.287e-09	1.447e-07	1.448e-07	5.275e-09	3.105e-09	4.281e-09	1.472e-07
19	1.489e-09	1.488e-09	5.821e-11	2.884e-09	2.396e-09	1.137e-10	4.111e-11	1.52e-10	4.198e-08
20	3.237e-09	5.192e-09	1.62e-10	3.186e-09	5.153e-09	7.105e-11	1.07e-10	6.919e-11	5.781e-09
root sum of squares	9.884e-07	9.884e-07	1e-06	9.884e-07	9.884e-07	1e-06	1e-06	1e-06	9.918e-07
full run root cov	9.883e-07	9.884e-07	1e-06	9.884e-07	9.884e-07	1e-06	1e-06	1e-06	9.87e-07
difference	-1.4e-11	-7.572e-13	-5.385e-15	-8.379e-12	-3.748e-11	-6.102e-14	-1.232e-14	-7.436e-14	-4.773e-09

Table B.12: Error budget for gyroscope misalignment, nonorthogonality, and scale factor bias states - final descent

	$\mathbf{m}_{gyro,x}$	$\mathbf{m}_{gyro,y}$	$\mathbf{m}_{gyro,z}$	$\mathbf{n}_{gyro,x}$	$\mathbf{n}_{gyro,y}$	$\mathbf{n}_{gyro,z}$	$\mathbf{s}_{gyro,x}$	$\mathbf{s}_{gyro,y}$	$\mathbf{s}_{gyro,z}$
1	2.812e-12	5.872e-12	4.824e-11	5.554e-12	1.743e-11	1.223e-10	1.35e-10	2.354e-11	9.315e-13
2	3.328e-13	1.874e-12	6.361e-12	9.449e-13	1.548e-12	7.976e-12	6.813e-12	2.59e-12	7.648e-14
3	3.782e-12	2.163e-11	2.201e-11	2.541e-12	2.168e-11	2.139e-11	2.08e-11	2.499e-12	1.988e-12
4	2.738e-12	2.387e-11	4.188e-11	8.072e-12	2.086e-11	4.533e-11	4.59e-11	1.509e-11	1.07e-12
5	2.424e-11	2.335e-11	3.314e-11	2.43e-11	2.294e-11	3.202e-11	2.256e-11	2.416e-11	4.131e-12
6	3.317e-14	3.631e-14	1.686e-13	3.152e-14	1.413e-13	4.818e-13	7.726e-13	5.211e-14	1.173e-14
7	1.126e-12	6.629e-12	7.965e-12	1.177e-12	7.229e-12	2.143e-11	1.901e-11	4.748e-12	5.13e-13
8	1.242e-13	5.91e-14	1.306e-12	8.283e-14	3.935e-13	3.062e-12	4.27e-12	4.181e-13	2.952e-14
9	2.176e-14	3.155e-14	3.206e-13	2.805e-14	1.04e-13	7.69e-13	9.414e-13	1.295e-13	6.543e-15
10	4.429e-11	1.629e-10	1.8e-10	4.206e-11	1.594e-10	1.453e-10	4.226e-11	3.899e-11	1.318e-11
11	2.542e-12	2.3e-11	3.973e-11	7.753e-12	2.006e-11	3.956e-11	3.749e-11	1.421e-11	1.033e-12
12	2.528e-12	2.295e-11	3.99e-11	7.765e-12	2.008e-11	4.03e-11	3.785e-11	1.432e-11	1.032e-12
13	7.156e-13	1.234e-12	6.685e-12	5.393e-13	1.05e-12	1.772e-11	2.318e-11	2.782e-12	1.245e-14
14	1e-06	1e-06	1e-06	8.746e-13	3.301e-12	2.937e-12	8.273e-13	7.891e-13	2.701e-13
15	8.726e-13	3.302e-12	2.937e-12	1e-06	1e-06	1e-06	8.744e-13	7.928e-13	2.7e-13
16	4.061e-14	3.979e-13	1.104e-12	4.463e-14	3.744e-13	1.151e-12	1e-06	1e-06	1e-06
17	6.072e-12	2.964e-11	2.324e-11	5.199e-12	2.906e-11	2.806e-11	3.877e-11	6.32e-12	2.713e-12
18	3.834e-11	2.241e-10	2.876e-10	5.83e-11	2.134e-10	3.133e-10	3.265e-10	1.026e-10	1.344e-11
19	2.022e-12	3.495e-12	2.362e-11	2.085e-12	5.761e-12	5.791e-11	7.464e-11	9.573e-12	2.97e-13
20	6.217e-10	1.792e-09	1.875e-09	6.207e-10	1.795e-09	1.882e-09	1.797e-09	5.897e-10	1.967e-10
root sum of squares	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06
full run root cov	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06
difference	1.04e-16	1.049e-16	-8.784e-16	2.28e-17	-1.932e-17	-7.311e-15	-8.724e-15	-2.402e-16	1.969e-18

APPENDIX C
Sensitivity Analysis Plots

The following plots show sensitivity graphs for selected error group and state variable pairs. These all illustrate pairs where the error groups have a relatively high effect on the estimation covariance for the given state variable.

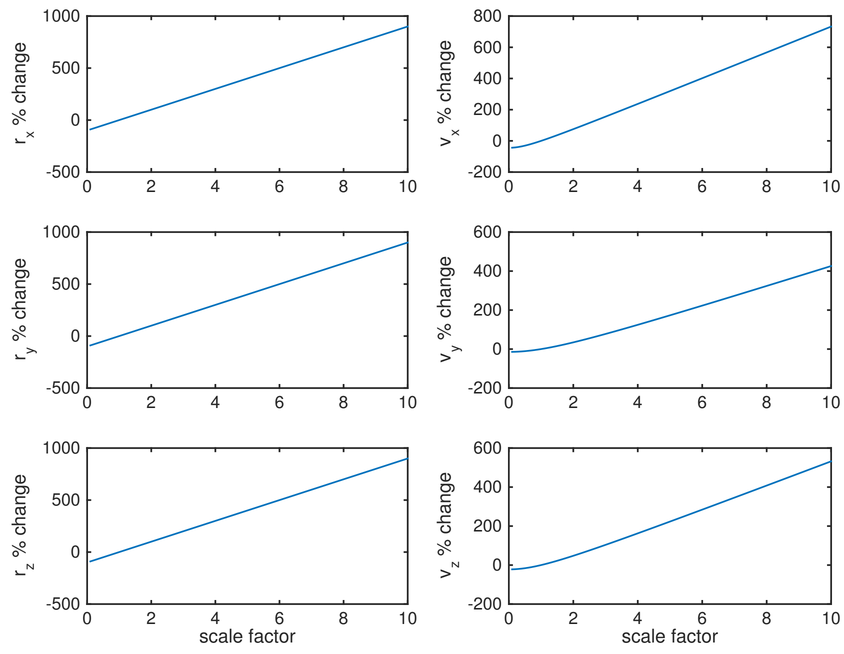

Figure C.1: Position and velocity estimation error variance sensitivity to error group 1, pre-THRN

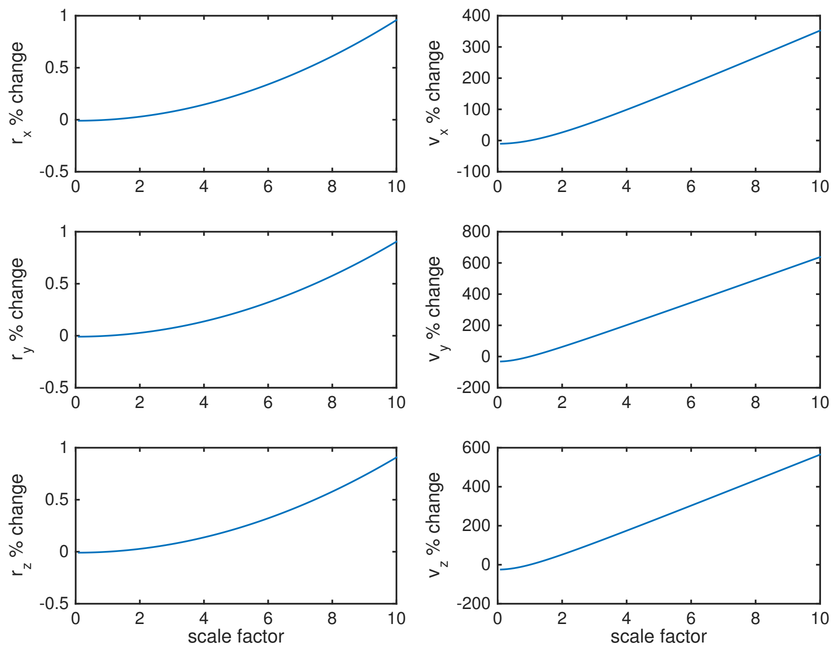

Figure C.2: Position and velocity estimation error variance sensitivity to error group 7, pre-THRN

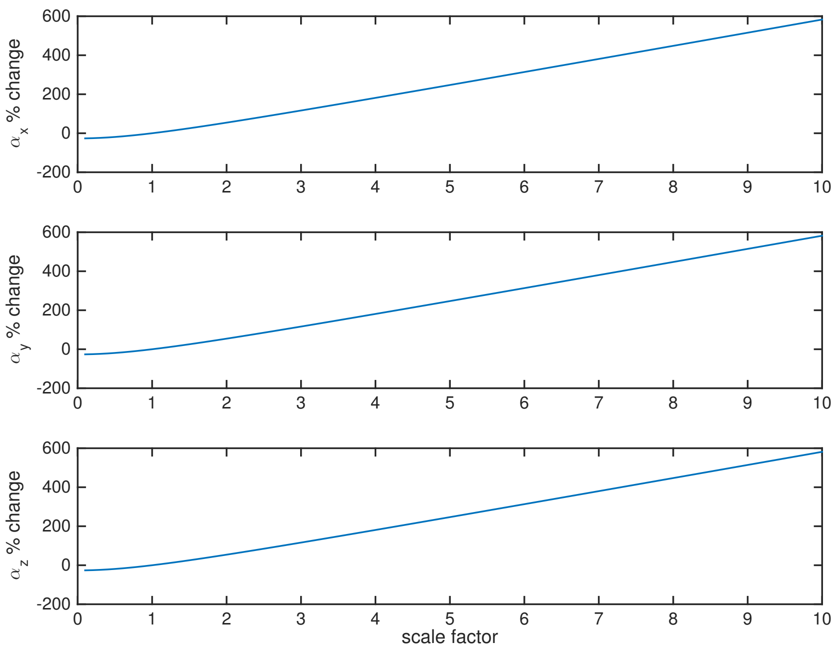

Figure C.3: Attitude estimation error variance sensitivity to error group 10, pre-THRN

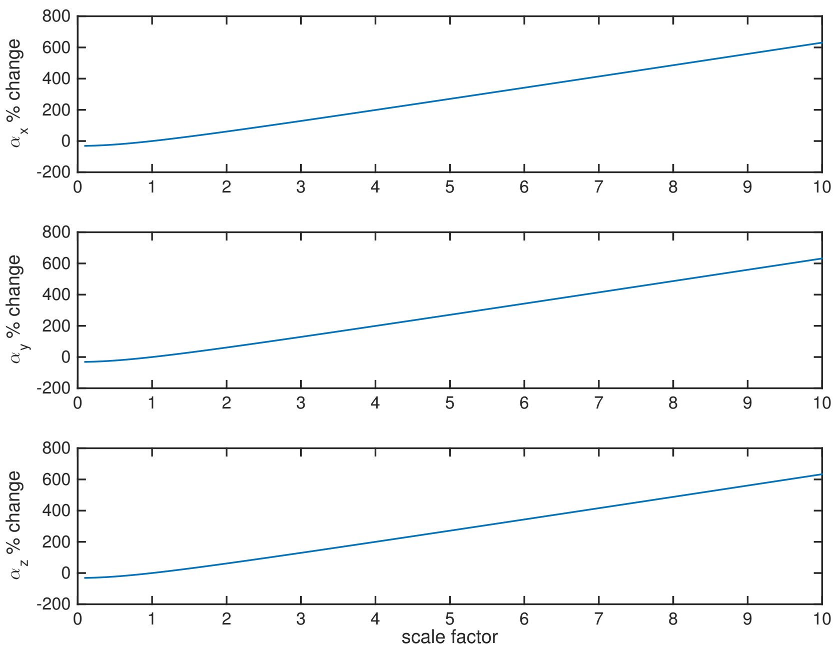

Figure C.4: Attitude estimation error variance sensitivity to error group 20, pre-THRN

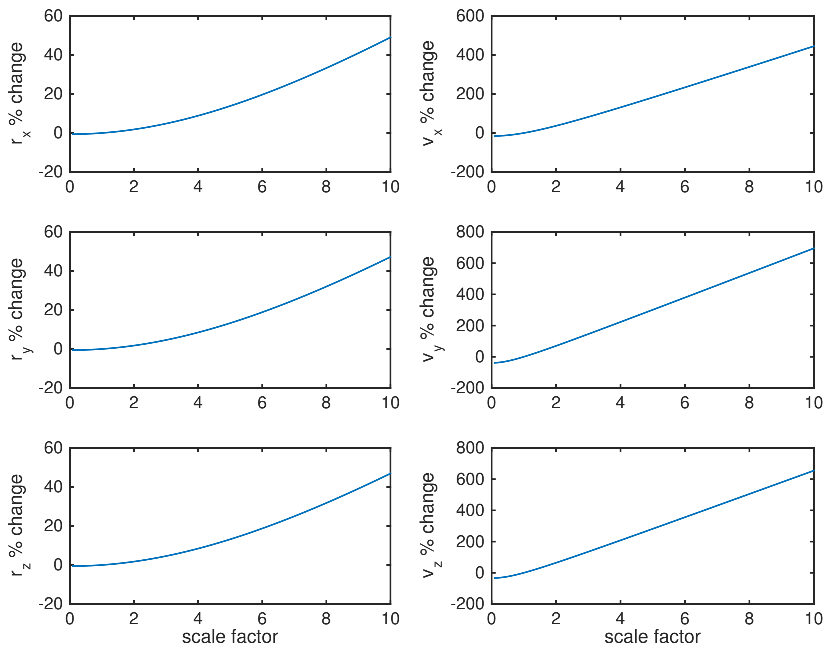

Figure C.5: Position and velocity estimation error variance sensitivity to error group 7, final descent

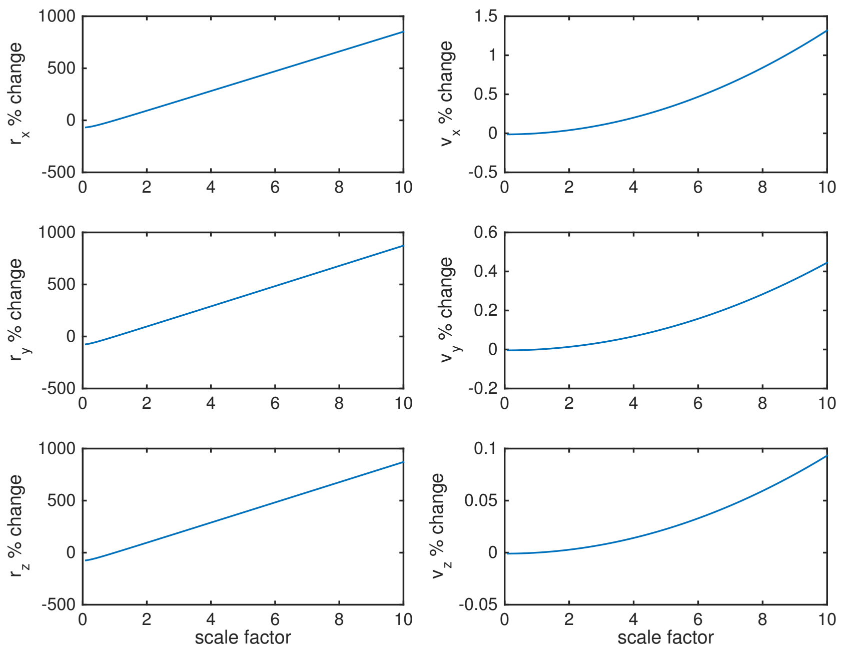

Figure C.6: Position and velocity estimation error variance sensitivity to error group 8, final descent

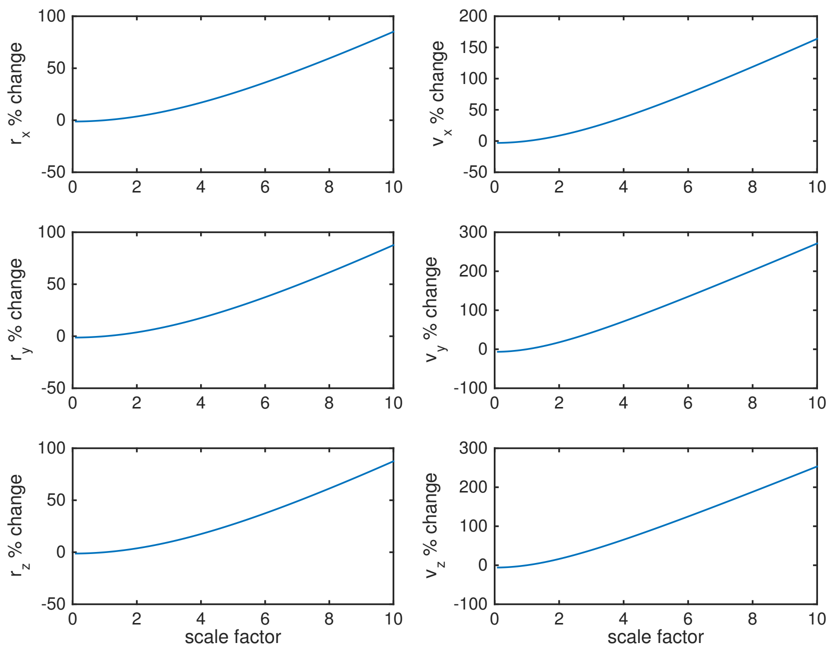

Figure C.7: Position and velocity estimation error variance sensitivity to error group 17, final descent

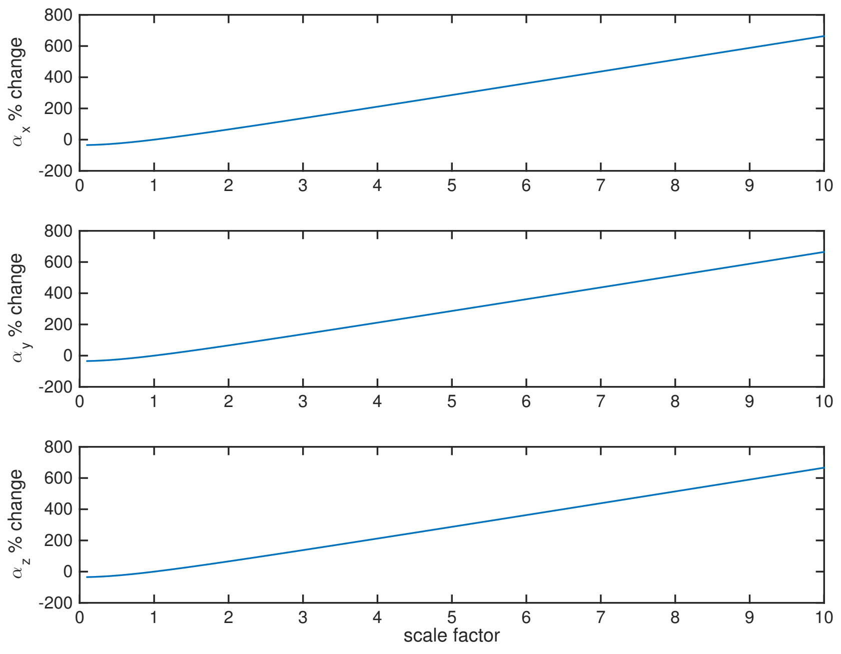

Figure C.8: Attitude estimation error variance sensitivity to error group 10, final descent

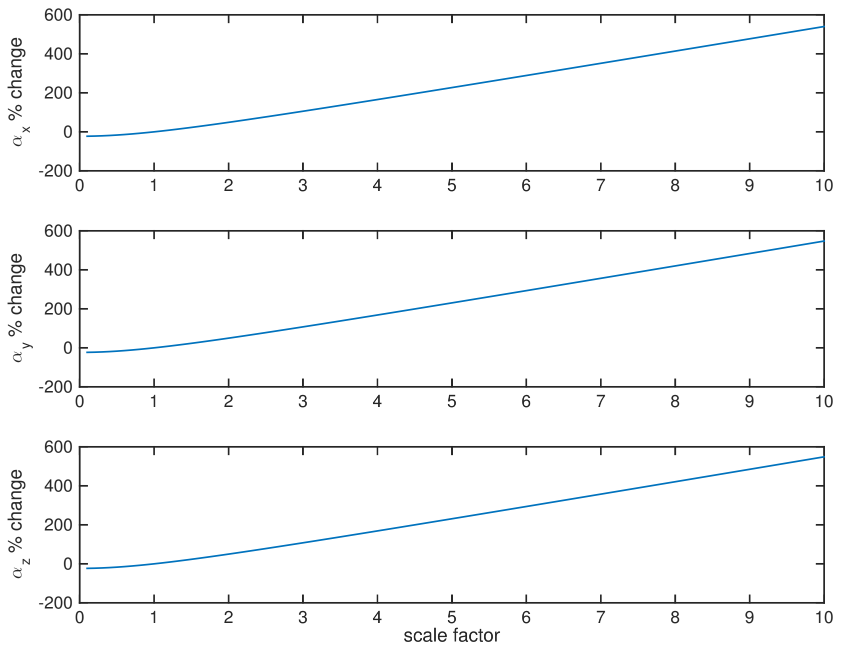

Figure C.9: Attitude estimation error variance sensitivity to error group 20, final descent

REFERENCES

[1] C.D. Epp and T.B. Smith, “Autonomous precision landing and hazard detection and avoidance technology (alhat)”, in *Aerospace Conference, 2007 IEEE*, March 2007, pp. 1–7.

[2] David W. Way, “Preliminary assessment of the mars science laboratory entry, descent, and landing simulation”, in *Aerospace Conference, 2013 IEEE*, March 2013, pp. 1–16.

[3] Otto W Dieffenbach, “Autonomous precision approach and landing system (apals)”, in *SPIE’s 1995 Symposium on OE/Aerospace Sensing and Dual Use Photonics*. International Society for Optics and Photonics, 1995, pp. 158–164.

[4] W. Mahmood and S.M.A. Shah, “Vision based hazard detection and obstacle avoidance for planetary landing”, in *Nonlinear Dynamics and Synchronization, 2009. INDS ’09. 2nd International Workshop on*, July 2009, pp. 175–181.

[5] Shengying Zhu, Pingyuan Cui, and Haijing Hu, “Hazard detection and avoidance for planetary landing based on lyapunov control method”, in *Intelligent Control and Automation (WCICA), 2012 10th World Congress on*. IEEE, 2012, pp. 2822–2826.

[6] F. Amzajerdian, L. Petway, G. Hines, B. Barnes, D. Pierrottet, and G. Lockard, “Doppler lidar sensor for precision landing on the moon and mars”, in *Aerospace Conference, 2012 IEEE*, March 2012, pp. 1–7.

[7] Chirold D Epp, Edward A Robertson, and JM Carson, “Real-time hazard detection and avoidance demonstration for a planetary lander”, in *AIAA SPACE 2014 Conference and Exposition*, 2014.

[8] E. J. Lefferts, F. L. Markley, and M. D. Shuster, “Kalman filtering for spacecraft attitude estimation”, *Journal of Guidance, Control, and Dynamics*, vol. 5, no. 5, pp. 417–429, 2015/01/31 1982.

[9] MATLAB, “version 8.4.0.150421 (r2014b)”, The Mathworks, Inc., Natick, Massachusetts, 2014.

[10] Hari B Hablani, Myron L Tapper, and David J Dana-Bashian, “Guidance and relative navigation for autonomous rendezvous in a circular orbit”, *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 3, pp. 553–562, 2002.

[11] Anastasios I Mourikis, Nikolas Trawny, Stergios I Roumeliotis, Andrew Edie Johnson, Adnan Ansar, and Larry Matthies, “Vision-aided inertial navigation for spacecraft entry, descent, and landing”, *Robotics, IEEE Transactions on*, vol. 25, no. 2, pp. 264–280, 2009.

[12] Rudolph Emil Kalman, "A new approach to linear filtering and prediction problems", *Journal of Fluids Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[13] Jason L. Speyer and Walter H. Chung, *Stochastic Processes, Estimation, and Control*, Advances in Design and Control. Society for Industrial and Applied Mathematics, 2008.

[14] Bruce P. Gibbs, *Advanced Kalman Filtering, Least-Squares and Modeling*, Wiley, 2011.

[15] Greg Welch and Gary Bishop, "An introduction to the kalman filter", 1995.

[16] Kyle J. DeMars and Robert H. Bishop, "Navigation analysis to facilitate precision descent navigation for landing at the moon", in *Spaceflight Mechanics 2008 (Advances in the Astronautical Sciences)*. AAS/AIAA, 2008, vol. 130.

[17] Arthur Gelb, Ed., *Applied Optimal Estimation*, The MIT Press, Cambridge, MA, 1974.

[18] J. L. Crassidis and J. L. Junkins, *Optimal Estimation of Dynamic Systems*, CRC Press, New York, NY, 2004.

[19] Jack B Kuipers, *Quaternions and rotation sequences*, vol. 66, Princeton university press Princeton, 1999.

[20] Evangelos A Coutsias and Louis Romero, "The quaternions with an application to rigid body dynamics", Tech. Rep., Sandia National Laboraties, 2004.

[21] Jozef C Van Der Ha and Malcolm D Shuster, "A tutorial on vectors and attitude [focus on education]", *Control Systems, IEEE*, vol. 29, no. 2, pp. 94–107, 2009.

[22] D.J. Tylavsky and G.R.L. Sohie, "Generalization of the matrix inversion lemma", *Proceedings of the IEEE*, vol. 74, no. 7, pp. 1050–1052, July 1986.

[23] J Dickey, P Bender, J Faller, X Newhall, R Ricklefs, J Ries, P Shelus, C Veillet, A Whipple, J Wiant, et al., "Lunar laser ranging: A continuing legacy of the apollo program", 1994.

[24] Jack L. Bufton, "Laser altimetry measurements from aircraft and spacecraft", *Proceedings of the IEEE*, vol. 77, no. 3, pp. 463–477, Mar 1989.

[25] Navid Serrano, "A bayesian framework for landing site selection during autonomous spacecraft descent", in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 5112–5117.

[26] Kevin Miller, Jim Masciarelli, and Reuben Rohrschneider, "Advances in multi-mission autonomous rendezvous and docking and relative navigation capabilities", in *Aerospace Conference, 2012 IEEE*. IEEE, 2012, pp. 1–9.

[27] Brandon M Jones and Ayanna Howard, "An imaging technique for safe spacecraft landing and autonomous hazard avoidance", in *Second IEEE International Conference on Space Mission Challenges for Information Technology, 2006. SMC-IT 2006*. IEEE, 2006, pp. 8–pp.

[28] Feng Junhua, Cui Pingyuan, and Cui Hutao, “Autonomous hazard detection and landing point selecting for planetary landing”, in *3rd International Symposium on Systems and Control in Aeronautics and Astronautics (ISSCAA), 2010*. IEEE, 2010, pp. 1292–1296.

[29] Farzin Amzajerdian, Diego Pierrottet, Larry Petway, Glenn Hines, and Vincent Roback, “Lidar systems for precision navigation and safe landing on planetary bodies”, in *International Symposium on Photoelectronic Detection and Imaging 2011*. International Society for Optics and Photonics, 2011, pp. 819202–819202.

[30] R.W.H. van Bezooijen, “Techniques for optimizing an autonomous star tracker”, Apr. 28 1998, US Patent 5,745,869.

[31] C.C. Liebe, E.W. Dennison, B. Hancock, R.C. Stirbl, and B. Pain, “Active pixel sensor (aps) based star tracker”, in *Aerospace Conference, 1998 IEEE*, Mar 1998, vol. 1, pp. 119–127 vol.1.

[32] R. H. Bishop, “Generating error vectors with a prescribed covariance”, September 2011.

[33] R. H. Bishop and K. J. DeMars, “Alhat navigation analysis”.

[34] Jerry LeCroy, Dean Hallmark, Peter Scott, and Richard Howard, “Comparison of navigation solutions for autonomous spacecraft from multiple sensor systems”, in *SPIE Defense and Security Symposium*. International Society for Optics and Photonics, 2008, pp. 69580D–69580D.

[35] Aaron C Brogley and Ronald Maglothin, “Lean development with the morpheus simulation software”, 2013.

[36] Robert L Hirsh, Zarrin K Chua, Todd A Heino, Al Strahan, Laura Major, and Kevin Duda, “Developing a prototype alhat human system interface for landing”, in *Aerospace Conference, 2011 IEEE*. IEEE, 2011, pp. 1–14.

[37] Timothy Crain and Robert Bishop, “Mars entry navigation: atmospheric interface through parachute deploy”, *AIAA Paper*, vol. 4501, 2002.

[38] Jerry Ginsberg, *Engineering Dynamics*, Cambridge University Press, 2008.

[39] Robert H. Bishop and Kyle DeMars, “Alhat navigation: Mathematical specifications”, Unpublished, 2007.