5-2010

Analysis of Square-Root Kalman Filters for Angles-Only Orbital Navigation and the Effects of Sensor Accuracy on State Observability

Jason Knudsen Schmidt
Utah State University

ANALYSIS OF SQUARE-ROOT KALMAN FILTERS FOR ANGLES-ONLY ORBITAL NAVIGATION AND THE EFFECTS OF SENSOR ACCURACY ON STATE OBSERVABILITY

by

Jason Knudsen Schmidt

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Aerospace Engineering

Approved:

Dr. David K. Geller Major Professor	Dr. Charles M. Swenson Committee Member
Dr. Stephen A. Whitmore Committee Member	Dr. Byron R. Burnham Dean of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2010

Abstract

Analysis of Square-Root Kalman Filters for
Angles-Only Orbital Navigation and
the Effects of Sensor Accuracy
on State Observability

by

Jason Knudsen Schmidt, Master of Science
Utah State University, 2010

Major Professor: Dr. David K. Geller
Department: Mechanical and Aerospace Engineering

Angles-only navigation is simple, robust, and well proven in many applications. However, it is sometimes ill-conditioned for orbital rendezvous and proximity operations because, without a direct range measurement, the distance to approaching satellites must be estimated by firing thrusters and observing the change in the target's bearing. Nevertheless, the simplicity of angles-only navigation gives it great appeal. The viability of this technique for relative navigation is examined by building a high-fidelity simulation and evaluating the sensitivity of the system to sensor errors. The relative performances of square-root filtering methods, including Potter, Carlson, and UD factorization filters, are compared to the conventional and Joseph formulations. Filter performance is evaluated during closed-loop "station keeping" operations in simulation.

(165 pages)

Acknowledgments

I feel it is critical to thank friends, fellow students, and professors that I have depended on for the last few years. I'd like to thank my roommates who have given me a home to go home to after long nights on campus. They broke the monotony and added excitement to the life of a single grad student! When I think of the mountains I've climbed with fellow students Shane Robinson, Ben Timmins, Nicholas Maughan and many others, I quickly realize that I couldn't have learned everything on my own. Without them I would have run out of steam, hit dead-ends, and settled for something less. With them at my side, nothing was impossible and fatigue was never an excuse. I owe much to their brilliant minds, endless motivation, and supportive friendship.

I cannot fail to recognize the incredible debt I owe to Dr. David Geller. He chose to hire an undergraduate engineering student and show the way to the exciting and fulfilling world of aerospace engineering. I wouldn't be anywhere near the place I am now without his gentle yet incisive guidance and generous patronage. Finally, I would like to thank my parents. My mother who taught me to be organized and balanced in a busy world and my father who inspires me to new heights whenever I talk with him. Under their loving instruction I have learned to cherish friendship, goodness, hard work, service, honesty, and my indispensable Heavenly Father.

Jason Schmidt

Contents

List of Tables

List of Figures

Notation

Decorations

$\bar{o}$	True Value (vector)
$\hat{o}$	Estimated Value (vector)
$\widetilde{o}$	Measured Value (vector)
$\dot{o}$	Time derivative

Superscripts

o^-	Previous Time step Estimate
o^+	Updated Estimate
o^{cam}	Chaser camera frame
o^b	Chaser body frame

Subscripts

o_c	Chaser values
o_t	Target values

Errors

ν	Measurement Error $(\widetilde{o} - \bar{o})$
e	Residual Error $(\widetilde{o} - \hat{o})$
ϵ	Misalignment
β	Bias

Noise

w	Process Noise
η	Measurement Noise

Other

$[o\times]$	Cross Product Matrix Form
$\boldsymbol{A}$	Bold capital is a matrix
f(x)	Function of x

Acronyms

AON	angles-only navigation
ARCSS	Autonomous Rendezvous and Capture Sensor System
AVGS	Advanced Video Guidance Sensor
DOF	degree of freedom
ECI	earth-centered inertial frame
ECRV	exponentially correlated random variable
EKF	extended Kalman filter
GN&C	guidance, navigation, and control
GPS	Global Positioning System
IMU	inertial measurement unit
LEO	low-earth orbit
LIDAR	light detection and ranging
LVLH	local vertical local horizontal frame
LOS	line-of-sight
MGS	Modified Gram-Schmidt
PD	position-derivative (controller)
PID	position-integral-derivative (controller)
PPM	parts per million
RPO	rendezvous and proximity operations
RSS	root-sum-square
TPI	terminal phase initiation
UD	upper-diagonal
WMGS	Weighted Modified Gram-Schmidt

Chapter 1

Introduction

The intent of this study is to understand the feasibility of angles-only navigation for orbital rendezvous operations and to see if square-root Kalman filters will do angles-only navigation better than conventional filters. This study is motivated by a growing interest in autonomous rendezvous and docking operations. This interest has resulted from a shift in political priorities and improvements in small satellite capabilities.

Since the close of the Cold War, the United States has had a more difficult time justifying monolithic satellite missions like Hubble. These massive satellites require billions of dollars of one-time funding and are susceptible to the failure of a single subsystem. While Hubble has benefited from manned repair missions, the majority of missions cannot expect such support. In addition, budgeting would be simplified if the the cost of these missions could be spread over several years.

In order to address these issues, NASA adopted the "smaller, faster, cheaper" slogan. However, the reduced capability of these small satellites was inadequate for many missions. The new buzzword in small satellite circles is "fractionated space." Primarily implemented by DARPA's F6 program, the goal is to replace extremely capable monolithic satellites with clusters of small wirelessly-networked satellites. The hope is to improve the reliability, survivability, and serviceability of large, complex systems without losing important capabilities [1].

Programs such as F6 will require small spacecraft to rendezvous and coordinate with each other on a regular basis. For F6 this challenge is simplified because each satellite will cooperate by beaming range, attitude, and other important information to the other spacecraft. However, it is probable that even more advanced missions will require small satellites to rendezvous with uncooperative satellites that may have lost power, or belong

to other parties. In order to keep spacecraft small, sufficient navigation information for rendezvous and proximity operations must be extracted from small, low-powered sensors.

One of the most simple yet useful sensors is a camera, whether infrared or optical. With a camera, a satellite can track where a second satellite is located within its field-of-view. While such measurements can be obtained from many types of sensors (e.g. Lidar or Radar), the camera has the additional advantage of being entirely passive.

The line-of-sight (LOS) measurement is the foundation of angles-only navigation (AON). Navigation by measuring angles is not new. Sailors take angle measurements to the north-star to determine latitude. Hikers take bearings to mountains and other known objects on the horizon to determine their own position. Astronauts on Gemini and Apollo would trigger key maneuvers required for orbital rendezvous by measuring the relative elevation angle to the target spacecraft [2, pg 42, 47]. The Deep Space One spacecraft autonomously estimated its position and velocity on its way to rendezvous with asteroids and comets by performing optical data triangulation to known planets, asteroids, and other bodies [3, pg 2-3].

It is clear that angles-only navigation is simple, robust, and well proven in many applications. However, angles-only-navigation can be ill-conditioned for orbital rendezvous and proximity operations because, without a direct range measurement, the distance to approaching satellites must be estimated in other ways. Nevertheless, the simplicity of angles-only navigation gives it great appeal.

Various strategies have been developed to overcome the limitations of AON. Two techniques are 1) taking "apparent diameter" measurements to the target object, and 2) performing translational maneuvers so the range of the target may be estimated [4]. Apparent diameter measurements have been explored in detail in [2, pg 119]. Both methods have their advantages. Apparent diameter measurements are range limited based on the resolution of the camera and require prior knowledge of the satellite being observed. Translational maneuvers consume fuel and lower the life of the satellite.

Using translation maneuvers to estimate range can be readily compared to orienteering

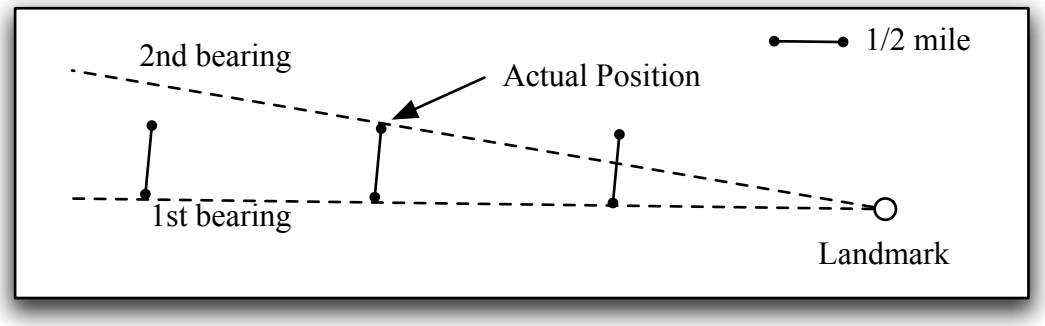


Fig. 1.1: Hiker analogy for observing range.

with a compass (see figure 1.1). Hikers can take a bearing to a distant landmark and draw a line on the map, emanating from the landmark, representing all of their possible locations. If there was a second landmark, a second measurement would result in a second line. The hikers would be where the two lines cross. In the absence of a second landmark, the hikers must get more creative. If they are carrying a pedometer or know the length of their stride, they can walk a half mile in a direction normal to the first line. Taking a second bearing measurement, only one location will satisfy the half mile distance they walked. In this way one could estimate position using a single landmark.

Translational maneuvers to estimate range have been explored for land and sea based operations [5–8], but is not as well understood for space rendezvous applications. In concept, a satellite would track it's target using a camera. These angles-only measurements would be processed by a state estimator like a Kalman filter. Assuming the satellite knows where it is, it can narrow the possible locations of the target satellite to a narrow cone projecting out from the camera. Of course, the range is unknown at this point. Thrusters can then be fired in a direction that will cause the angle to the target spacecraft to change. This change is a function of the magnitude and direction of the ΔV imparted by the thrusters, and the range to the target. The ΔV can be measured with accelerometers, leaving the range as the free variable to be solved for (see section 3.4 for a more detailed description of this concept).

Real world implementation of angles-only navigation for orbital rendezvous has some complications. First, the measurements will not be exact, so how will line-of-sight camera and accelerometer errors effect the ability to estimate relative position, especially in the range component? Second, numerical word length is limited in a digital computer. Under what conditions will finite word length cause problems? Can square-root filtering methods alleviate these problems? It is the purpose of this research to answer these questions and to demonstrate the feasibility of using AON for orbital rendezvous through high-fidelity simulation.

To provide background and context for this research, a more thorough discussion of space-based angles-only-navigation missions and research is contained in Chapter 3. The Kalman filter algorithm and its more numerically stable relatives are covered in Chapter 4. A more intuitive understanding of numerical roundoff error and its effects on filter performance may be obtained by studying the "Toy Cases" explored in Chapter 5.

To aid in this research, a high fidelity, six degree-of-freedom simulation of two spacecraft (a "chaser" and "target") in low-Earth orbit was developed. The simulation includes sensor, actuator, and dynamic models that include noise, bias and other errors. Appropriate attitude and translational controllers were built for the "chaser" spacecraft. The simulation models are covered in detail in Chapter 6.

Chapter 7 develops an extended Kalman filter for Angles-Only-Navigation applications. The filter design model, linearization of these models and other supporting derivations are contained here.

The effects of LOS camera and accelerometer errors as well as finite numerical word length was explored by running the simulation open-loop and processing the sensor data in the Kalman filters after-the-fact. Results of the open-loop analysis are found in Chapter 8. To obtain more realistic performance results of AON for orbital rendezvous and proximity operations, the filters were also implemented in a closed-loop manner. Chapter 9 details their performance during "station-keeping" scenarios. Conclusions and possible future work are discussed in Chapter 10.

Chapter 2

Thesis Statement

The thesis of this research is to show that angles-only navigation (AON) is viable for short duration orbital rendezvous and proximity operations (RPO) and that square-root formulations of the Kalman filter will exhibit better numerical stability than the standard extended Kalman filter under realistic conditions.

Chapter 3

Literature Survey

3.1 Rendezvous Missions that Used Angles based Navigation

Line-of-sight or angles based navigation has been implemented during proximity operations as early as Gemini. During rendezvous operations, Gemini astronauts would trigger the terminal phase initiation (TPI) burn when the relative elevation angle to the target was at 27.5 degrees [9]. Apollo adopted a similar strategy for initiating orbital rendezvous [10]. In both cases, an angle measurement was chosen to initiate the TPI burn to reduce approach trajectory dispersions and because it lent itself to backup techniques dependent on the crew [9, pg 1024]. To keep things simple, relative attitude determination and docking relied on the human eyeball.

The simple, fault tolerant systems on Gemini and Apollo were replaced by much more complex systems on later spacecraft. The American Space Shuttle uses Radar as the primary sensor for proximity operations, but also has a laser ranging device and a centerline camera [11]. The Russian Soyuz Spacecraft employs multiple directional and omnidirectional radio antennas and receivers on both vehicles to acquire range, range-rate, line-of-site angles, and relative attitude measurements [12]. The Shuttle and Soyuz systems both consume a lot of power, which would be unacceptable on a small spacecraft.

Recent attempts at autonomous orbital rendezvous have tried to reduce the size and power-consumption of onboard sensors. XSS-11 employed an IMU, a sun sensor, a camera, and a scanning LIDAR [13].

The Demonstration of Autonomous Rendezvous Technology (DART) spacecraft had only GPS and the Advanced Video Guidance Sensor (AVGS) which had one camera and a laser to illuminate special reflectors on the target. At long ranges, GPS provided the required relative position data. At closer ranges the AVGS system allowed DART to determine

range through apparent diameter measurements [14]. Unfortunately, errors in computer software design and filter implementation caused the DART mission to fail [15].

Orbital Express was equipped with two imaging systems, the Autonomous Rendezvous and Capture Sensor System (ARCSS), which had three cameras, and AVGS, the same system used on DART. Orbital Express was also equipped with a laser rangefinder [16].

It is important to note that none of these spacecraft relied entirely on a non-illuminating (passive), angles-only camera. All of them were equipped with an active sensor to determine range, whether it was GPS, a laser rangefinder, or LIDAR.

3.2 The Kalman Filter

The Kalman Filter was first developed by R.E. Kalman [17] and has been improved and expanded over the years to process all sorts of measurements. A Kalman filter propagates an estimated state ($\hat{x}$) and the covariance (P_x) (or uncertainty) of that state in real time. When measurements are made, the estimated state is improved and the state covariance decreases.

The covariance of position can be thought of as a ellipsoid enclosing most of the possible solutions. The ellipsoid is represented by a 3x3 matrix. The eigenvalues of this matrix correspond to the length, width, and height of the ellipsoid.

The behavior of Kalman Filters when processing LOS measurements has been analyzed in [4, 18]. It was shown that during free motion the component of the covariance ellipsoid parallel to the LOS vector will grow almost without bound. As a result, extremely large and very small values will be contained in the same covariance matrix. These ill-conditioned covariance matrices can lead to numerical errors that can cause the filter to fail.

This problem can be alleviated by normalizing the matrix in some manner. Several filters that operate on the square-root of the covariance matrix have been developed and are well summarized by Maybeck [19, Chapter 7]. Taking the square-root of the covariance matrix makes large values smaller and small values larger, making the matrix better conditioned. This research will make use of the conventional, Joseph, Potter, Carlson, and UD factorization filters. These filter algorithms are covered in detail in Chapter 4.

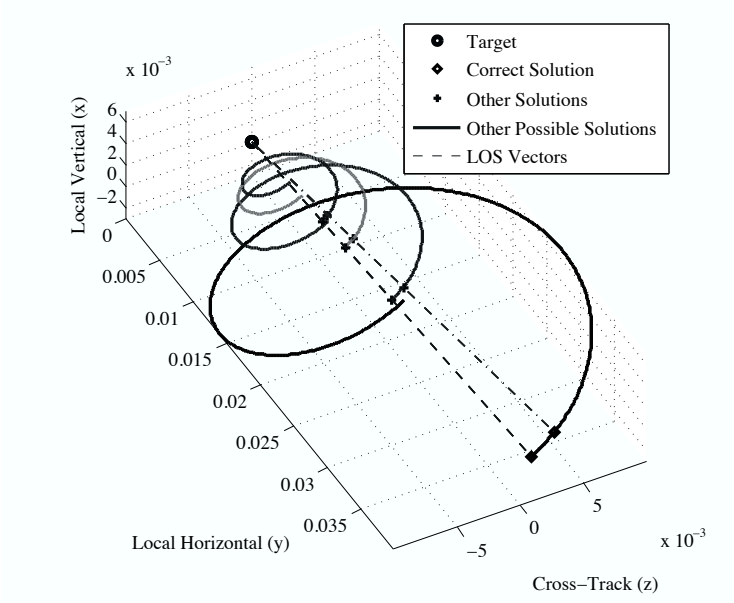


Fig. 3.1: A family of relative-motion trajectories exhibiting identical line-of-sight measurement histories.

3.3 Range Estimation with Angles-only Navigation

There are still unanswered questions about how effective angles-only navigation will be for proximity operations. For non-maneuvering satellites the problem is especially vexing. If the satellites are far apart enough to observe the curvature of their orbit, Gauss' method for orbit determination can be used to determine range [20]. However, at closer distances, Gauss' method breaks down. This is because whole families of trajectories will exhibit nearly identical Line-of-sight (LOS) measurement histories as seen in figure 3.1. They only differ in their range component. If this motion is linearized using CW equations, then the LOS measurement histories will truly be identical. Without a unique solution for the given measurements, the state of the chaser will remain unobservable.

When these angle measurements are processed in a Kalman filter, no information can be gleaned along the line of sight. Thus, the component of the covariance ellipsoid parallel to the LOS vector will grow almost without bound.

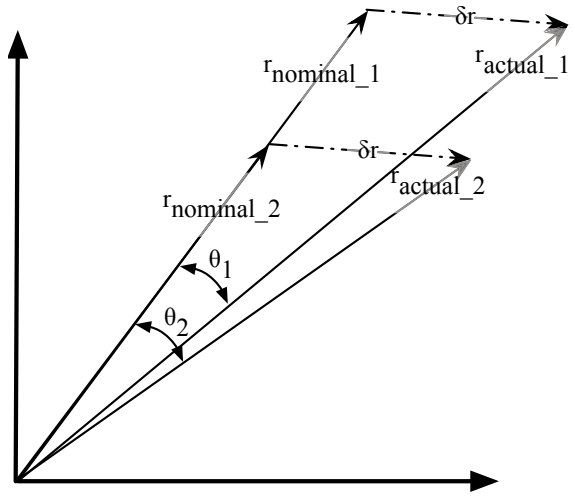


Fig. 3.2: Predictable change in observability angle (δr) results from known acceleration.

3.4 Observability Burns

In order to estimate range, satellites using AON during proximity operations can make a calculated maneuver to improve observability. As figure 3.2 shows, only one range will satisfy the known change in position, δr, resulting from a known acceleration delivered by the spacecraft's thrusters. It is important to note that δr cannot be parallel to $r_{nominal}$. The change in the LOS vector for a given δr is known as the observability angle (θ) [2] . These LOS measurements are processed in an EKF for state estimation.

The difference between the actual θ and the calculated θ is a function of accelerometer and centroiding errors. As shown in figure 3.3, if the measured observability angle is too small, then the Kalman filter will ignore it, because it falls within bounds that might be due to process noise. Only when the differences are statistically significant will the range estimate be improved. Thus, if accelerometers or camera measurements are poor, either the satellite must wait longer for the trajectories to diverge or larger burns will be required.

3.5 Summary of Related Work

Essentially, the proposed thesis will expand on previous work done by Raja Chari, Dave Woffinden, and Nathan Stastny. As summarized in table 3.1, Chari explored the navigation performance of an angles-only navigation Kalman filter during proximity operations. This

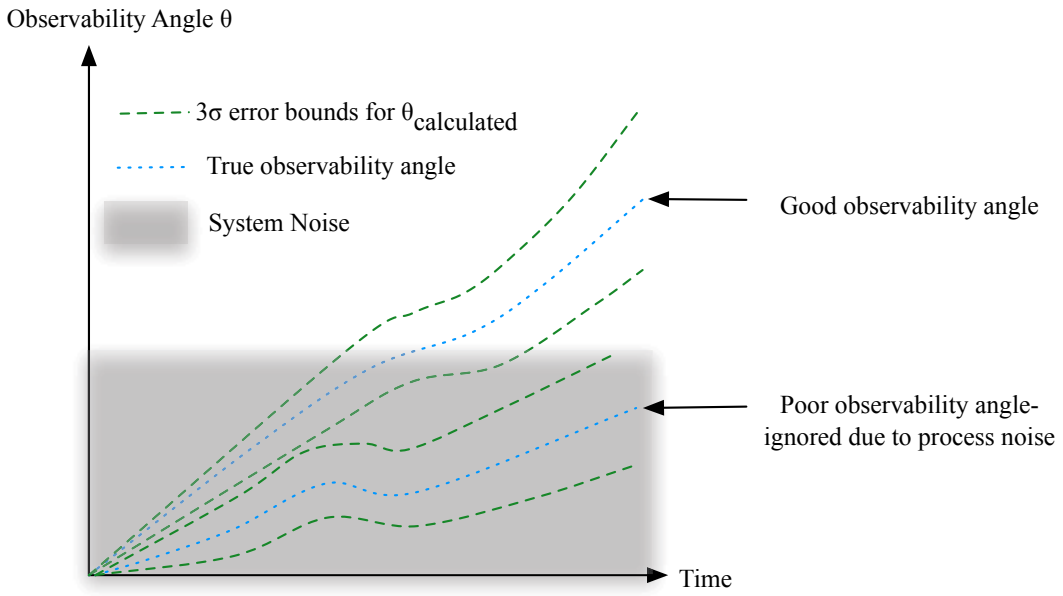


Fig. 3.3: Observability angle as a function of time.

analysis was done by way of linear covariance analysis, which produced highly useful, but idealized results. Dave Woffinden expanded on the concept with a six-degree-of-freedom simulation to prove the concepts applicability in a more realistic scenario. Nathan Stastny expanded on Chari's work by applying angles-only navigation techniques to deep space planetary rendezvous. The proposed thesis will expand on Dave Woffinden's work by comparing the performance of various square-root formulations of the EKF to the Conventional EKF (with and without the Joseph form of the update equations) under the more realistic conditions that a six-DOF simulation provides, and without the benefit of apparent diameter measurements.

Table 3.1: Summary of Related Research

	Chari [4]	Woffinden [21]	Stastny [3]	Schmidt
Year	2001	2004	2006	2010
Angles Only Nav	√	√	√	√
Standard Kalman Filter	√		√	
Extended Kalman Filter		√		√
Square-Root EKFs				√
Proximity Operations	√	√		√
LinCov	√		√	
Simulation		√		√
Apparent Diameter Measurements		√		
Line-of-Sight Errors	√	√	√	√
Accelerometer Errors				√
Closed-Loop		√		√
Open Loop	√		√	√

Chapter 4

Conventional and Square-Root Kalman Filters

4.1 Introduction to the Kalman Filter

The Kalman filter is a two-step process as shown in figure 4.1. The first step is to propagate the state and state covariance matrix. The second step is the "update" step, where the state and state covariance matrix are updated based on new measurements and a priori information.

There are a number of variations on the conventional Kalman filter. Some maintain the inverse of the covariance matrix, others maintain the square-root of the covariance matrix, and one uses the square-root of the inverse of the covariance matrix! These variations have been developed to improve the performance of a Kalman filter when implemented on a computer that has memory and word length constraints. Four of these variations are covered in sections 4.2.1 through 4.2.4. A study comparing their numerical performance is found in Chapter 5.

The conventional Kalman filter is shown in figure 4.2a. In addition to the variations mentioned above, the conventional Kalman filter may be derived in continuous time, discrete time, and combinations thereof. In the continuous case, the state estimate and state covariance are defined in term of their time derivatives (i.e. $\dot{x} = f(x)$), and they must be continuously integrated to get the actual value at any instant. There is no separate update step. In the discrete case, the state estimate and state covariance are propagated forward discretely (i.e. $x_{k+1} = \mathbf{\Phi}_k x_k$), and there is a separate update step whenever a measurement is processed.

This study will implement a hybrid of the two. The state estimate will be numerically integrated, while the state covariance will be propagated discretely. Both the state estimate and state covariance will be updated when a measurement is processed.

All five forms of the Kalman filter discussed in this chapter are applied in future chapters. Chapter 5 compares their numerical performance during application to two simple linear scenarios. Chapter 7 provides all of the groundwork necessary to create Extended forms of these filters for the non-linear scenario of rendezvousing satellites implementing Angles-only navigation.

It is important to understand key elements of the Kalman filter algorithm that apply to the conventional filter as well as its numerical variations discussed in sections 4.2.1 through 4.2.4. A brief summary of these elements, following the order of presentation in figure 4.2a, is included here.

First, every filter must be initialized with an initial state ($\hat{x}_0$) and state covariance ($\boldsymbol{P}_0$). The state may be position, velocity, bias, or any other value of interest. The initial covariance must be large enough that the estimated state will be free to move to the true value. If the initial covariance is too small, the filter will be locked and unresponsive to the measurements processed.

Next comes the propagation step. The dynamics of the state are modeled by the function $f(x)$. If $f(x)$ is a linear function then it may be described by a matrix $\boldsymbol{F}$ such that $\dot{x} = \boldsymbol{F}x$, and the state transition matrix ($\boldsymbol{\Phi}$) can be calculated ahead of time as $\boldsymbol{\Phi} = e^{\boldsymbol{F}dt}$ (where dt is the propagation stepsize) such that $x_{k+1} = \boldsymbol{\Phi}x_k$. If $f(x)$ is not a linear function, it may be linearized about either a nominal state or the current state estimate such that $\boldsymbol{F} = \frac{df(x)}{x}|_{\hat{x}}$. Thus, in the nonlinear case, the value of $\boldsymbol{F}$ and $\boldsymbol{\Phi}$ must be recalculated for every times-step. A filter that operates this way is known as an Extended Kalman filter.

It goes without saying that the mathematical model $f(x)$ will fail to capture all the dynamics present in a real world case. The Kalman filter assumes that these unmodeled effects can be accounted for by adding a Gaussian white or colored noise process with zero mean to the dynamic model. This noise causes the covariance to grow with time. The strength of this noise is captured in the matrix $\boldsymbol{Q}$. Thus, the covariance is propagated forward in time with the state transition matrix and grows larger due to $\boldsymbol{Q}$, the noise strength, and $\boldsymbol{B}$, the noise input matrix.

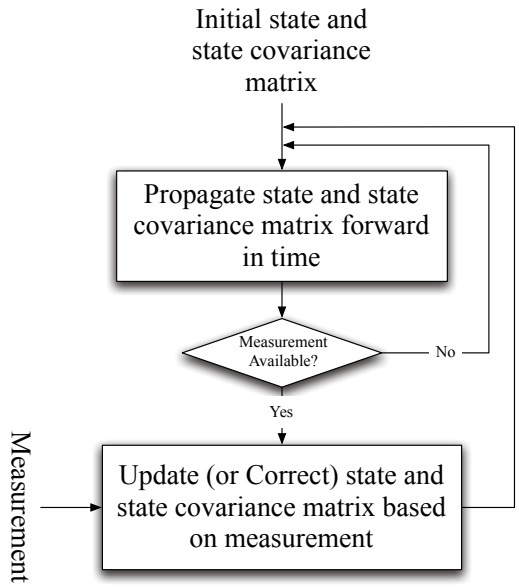


Fig. 4.1: Basic Kalman filter algorithm.

The update step introduces the sensor model $h(x)$. This function relates the state to the measurement. Like the dynamic model, if $h(x)$ is a nonlinear function, it can be linearized by $\boldsymbol{H} = \frac{h(x)}{x}|_{\hat{x}}$, where $\boldsymbol{H}$ is known as the measurement sensitivity matrix. The sensor model is used to predict what the measurement should be. When a measurement is made, the error between the estimated measurement ($\hat{y}$) and actual measurement ($\widetilde{y}$) is used to improve the estimate of the state, and to modify the shape and size of the covariance matrix. However, the actual measurement is not trusted completely. It is given a weight based on a priori knowledge of the sensor measurement covariance ($\boldsymbol{R}$), the current state covariance, and the linearized sensor model. This weight ($\boldsymbol{K}$) is known as the Kalman gain.

The initial state and state covariance matirx ($\hat{x}_0$ and $\boldsymbol{P}_0$), linearized dynamic model ($\boldsymbol{F}$), state transition matrix ($\boldsymbol{\Phi}$), noise strength and noise input matrices ($\boldsymbol{Q}$ and $\boldsymbol{B}$), measurement sensitivity matrix ($\boldsymbol{H}$), measurement covariance ($\boldsymbol{R}$), and Kalman gain ($\boldsymbol{K}$) are important elements common to all of the numerical variations of the Kalman filter discussed below, though they may take different forms.

4.2 Variations of the Kalman Filter

The variations of the conventional Kalman filter that will be discussed in this section include:

1. Joseph form Kalman filter [19, pg 237]
2. Potter covariance square root filter [19, pg 384]
3. Carlson covariance square root filter [19, pg 385]
4. UD covariance factorization filter [19, pg 392]

4.2.1 Joseph Form Kalman Filter

The Joseph form of the Kalman Filter addresses issues with the update portion of the Kalman filter algorithm. The symmetric and positive definite (positive, non-zero eigenvalues) nature of the covariance matrix can be lost in a finite word length computer when the covariance update equation is defined by $P^+ = (I - KH)P^-$. As seen in figure 4.2b, the Joseph form replaces this formulation with one better conditioned for numerical evaluation. This is the only change from the conventional formulation and, though it exhibits better numerical conditioning, is not classified as a square-root filter.

4.2.2 Potter Covariance Square-root Kalman Filter

The fully implemented Potter covariance square-root Kalman filter (see figure 4.3a) modifies both the propagation and update steps for the covariance matrix. The square-root of the state covariance matrix is maintained in place of the conventional covariance matrix. While the square-root of a matrix is not unique, the Cholesky factorization algorithm will find a lower triangular square-root in a numerically well conditioned manner. This algorithm is easily implemented in Matlab with $\boldsymbol{S}_0 = chol(\boldsymbol{P}_0, 'lower')$ where $\boldsymbol{S}\boldsymbol{S}^T = \boldsymbol{P}$. The square root of the noise strength is similarly derived with $\boldsymbol{W}_d = chol(\boldsymbol{Q}\,dt, 'lower')$, where $\boldsymbol{W}_d$ is the square root of the discrete process noise strength. The Potter formulation of the Kalman filter doubles the effective precision of the conventional filter in ill-conditioned problems.

There are three options for the propagation step. They are as follows.

- Matrix RSS (root-sum-square) method [19, pg 377]
- Modified Gram-Schmidt (MGS) method [19, pg 380]
- Householder transformation method [19, pg 382]

The first method is quick and easy but has the same numerical precision as the conventional Kalman filter propagation step. The MGS method requires more calculations than the Householder transformation method, but is slightly more precise when processing large residuals. See Appendix E.8 for implementation of the MGS method in code.

The update step for the square-root of the state covariance matrix was first developed by Potter for the case of scalar measurements. While vector forms of the Potter square-root update exist, they are unnecessary. Non-scalar measurement updates may be performed by using the Potter update multiple times. If the measurements are correlated (i.e. R is not diagonal), then a transformation must be performed before processing the measurements [19, pg 375]. In addition, while the state covariance matrix may start out lower triangular, the Potter update equation will not maintain the lower triangular form.

4.2.3 Carlson Covariance Square-root Kalman Filter

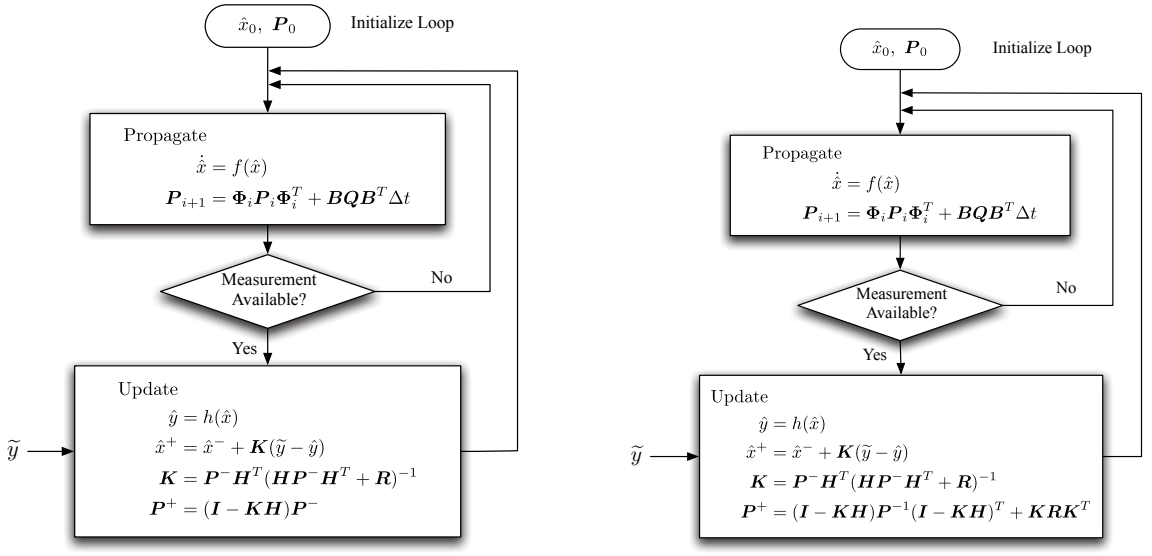
The Carlson covariance square-root Kalman filter (figure 4.3b) improves on the Potter filter by maintaining the covariance square-root matrix in triangular form. This reduces required computations and memory storage. Theoretical performance improvements are discussed in section 4.3. The propagation step is unchanged from that of the Potter filter. Like the Potter filter, the update step processes vector measurements sequentially as scalars.

4.2.4 UD Covariance Factorization Filter

Though the UD covariance factorization filter is not a true, square-root filter, it exhibits the same numerical performance as the Carlson and Potter formulations. The UD covariance factorization filter replaces the state covariance matrix P with upper triangular and diagonal

matrices such that $\boldsymbol{P} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T$. The upper and diagonal matrices are calculated with an algorithm outline in [19, section 7.7 pg 392] referenced here as $[\boldsymbol{U}, \boldsymbol{D}] = udu(\boldsymbol{P})$. This algorithm may be replicated in Matlab with $[\boldsymbol{u}, \boldsymbol{d}, \boldsymbol{p}] = ldl(\boldsymbol{P}, 'upper')$ where $\boldsymbol{D} = \boldsymbol{d}$ and $\boldsymbol{U} = \boldsymbol{p}^{-T}\boldsymbol{u}^T = \boldsymbol{p}' \backslash \boldsymbol{u}'$. The UD filter also requires use of the Weighted Modified Gram-Schmidt (WMGS) method detailed in [19, pg 397].

Like the Carlson filter, these triangular and diagonal forms reduce computation and memory requirements as compared to the Potter filter. In addition, the UD filter does not require any of the scalar square-root operations that the Carlson filter mandates. Thus, it has the best theoretical performance of any of the square-root filters, but considerable work must be done in order to take advantage of the symmetric and upper diagonal matrices. The UD covariance factorization filter is outlined in figure 4.3c.



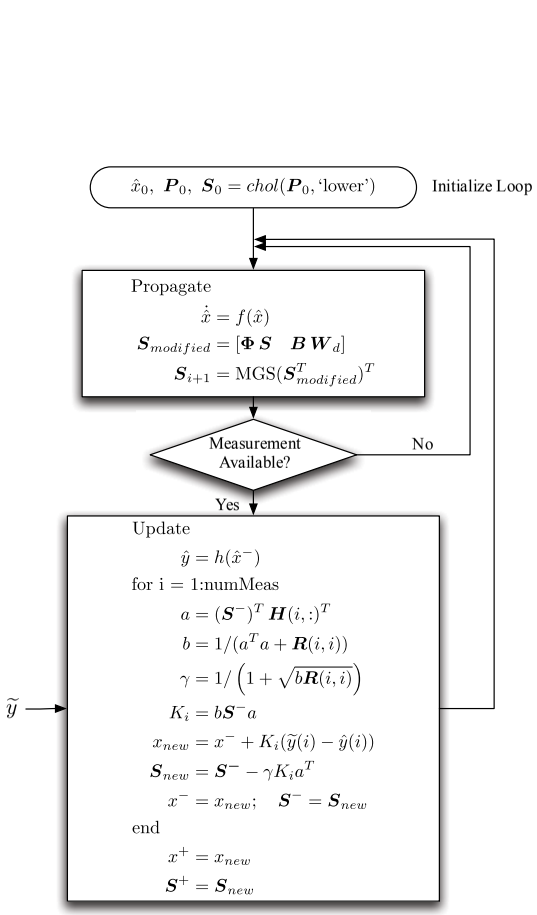
(a) Conventional Kalman Filter

(b) Joseph Form Kalman Filter

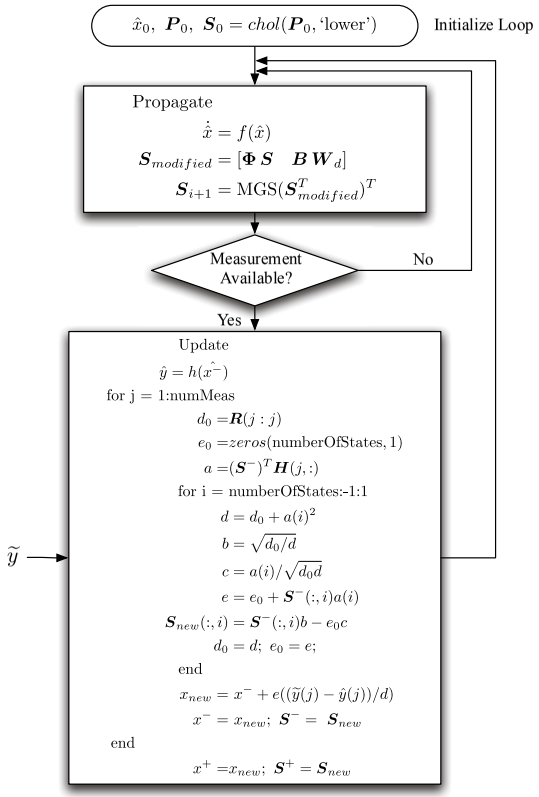
Fig. 4.2: Conventional and Joseph Kalman filter algorithms.

4.3 Comparison of Run-Times

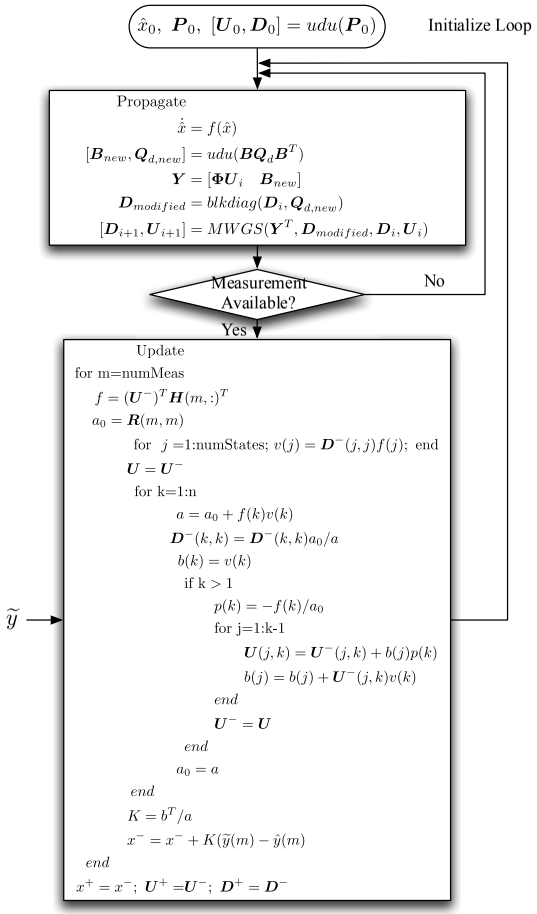
The numerical requirements for each of these filters is shown in figure 4.4. These bar plots are graphical representations of Maybeck's numerical analysis summarized in [19,



(a) Potter State Covariance Square-root Filter



(b) Carlson State Covariance Square-root Filter

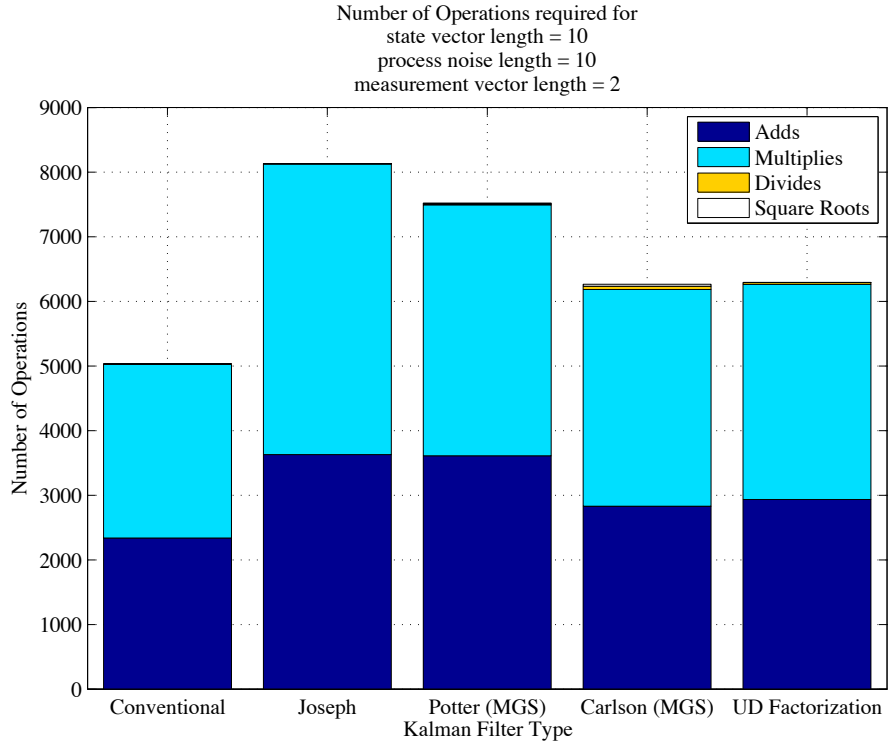


(c) UD Covariance Factorization Filter

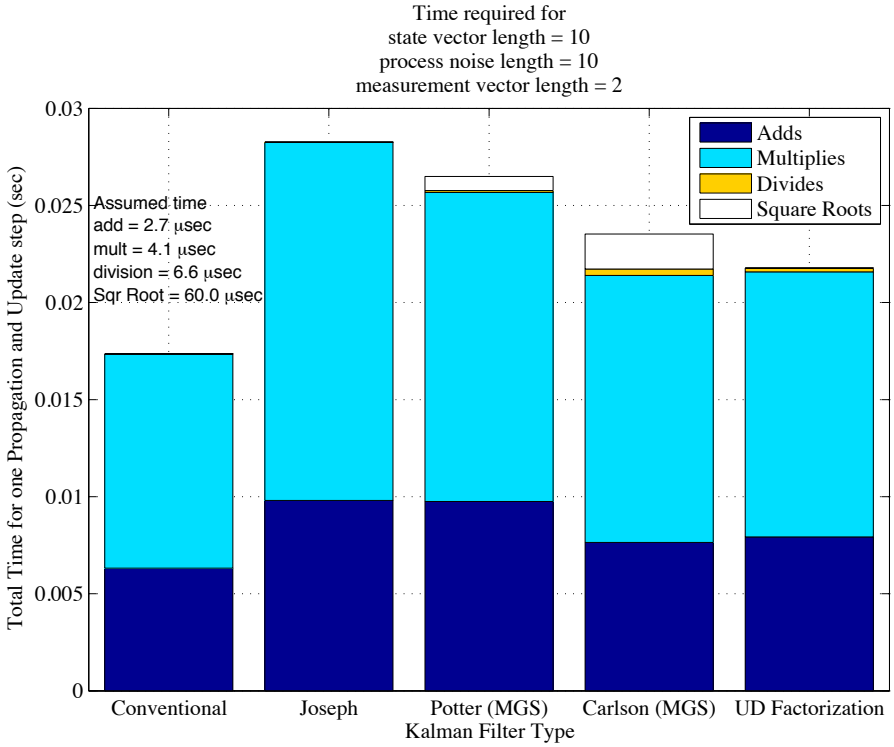
Fig. 4.3: Square-root filter algorithms.

Tables 7.1 and 7.2]. The run times calculated in figure 4.4b are based on instruction times on an IBM 360, a 1960's era machine, and do not reflect actual runtimes on any modern chip. Nevertheless, the advantages of the UD factorization filter over the Carlson filter are illustrated due to the lack of square-roots in the UD formulation. It should be noted that on modern machines the square-root operation is only slightly more expensive then division.

These calculations assume that $\mathbf{R}$ and $\boldsymbol{Q}_d$ (Where $\boldsymbol{Q}_d = \boldsymbol{Q}\Delta t$) are diagonal and that all implementations take advantage of symmetry and zeros as they appear in general forms. This last assumption is key. While a sparse matrix implementation of these filters in Matlab would take advantage of zeros, additional work would be necessary to take advantage of symmetry. Sparse matrices are not supported in Simulink®, so the theoretical speed advantages of the Carlson and UD factor filters over the Potter formulation would not be realized if implemented in Simulink without further modifications. In fact, without special implementation, the UD factorization filter takes over twice as long to run as the Potter filter.



(a) Number of operations required for each filter type.



(b) Runtime required for each filter type.

Fig. 4.4: Numerical requirements for Kalman filters.

Chapter 5

Numerical Errors in Kalman Filters

Numerical errors due to finite world length can be tricky to track down, especially in an algorithm as complex as a Kalman filter. Nevertheless, it is important that these effects be detected when they occur. In order to better understand the nature of these failures this chapter will examine the performance of the Conventional, Joseph, Potter, Carlson, and UD factorization filters in extreme detail for a couple of very simple "Toy Cases." The first toy case involves a single rotating shaft with absolute position measurements in degrees. The second toy case involves two rotating shafts with relative position measurements in degrees. The toy case linear Kalman filter codes are found in Appendix D.

5.1 Toy Case 1: Rotating Shaft with Absolute Measurements

Description of Toy Case 1

Cases 1A-1E involve filters estimating the state of a rotating shaft. Thus, the state is a two dimensional vector containing angular position (r) and angular rate (v). The only measurement comes from an extremely accurate position sensor with measurement noise standard deviation on the same order as machine epsilon ("eps" in the figures, and ϵ in the text), though measurement noise is varied from case to case. The initial condition is zero for position and velocity, however, there is process noise on both acceleration and velocity. By varying the amount of process noise and the size of the initial covariance values, the effect of numerical errors on filter performance may be observed and characterized. While these models were implemented in degrees and degrees per second, with a one second stepsize, only the numerical values themselves are relevant to the results emphasized in this chapter. Thus, units have been dropped in all tables and figures.

5.1.1 Case 1A: Working Case

Following the steps in figure 5.1, it can be seen that all five formulations of the Kalman filter are numerically stable and match the symbolic solution very well (though they are not identical to it). While the state covariance matrix is stored in a different form in the Potter, Carlson, and UD filters, the equivalent state covariance matrix is shown in these figures for comparative purposes. Walking through the steps sequentially reveals the following:

Step 0: Initially the covariance is an identity matrix.

Step 0^+: After the first update step the highly accurate position measurement results in a position covariance of R. The symbolic solution is slightly smaller. Thus, the numeric filters are slightly conservative. If R was sufficiently large, then the numeric filters would match the symbolic result exactly.

Step 1: After the first propagation step the velocity covariance has been fed into the position covariance. This is a very ill-conditioned situation, with a very small number (R) being added to one.

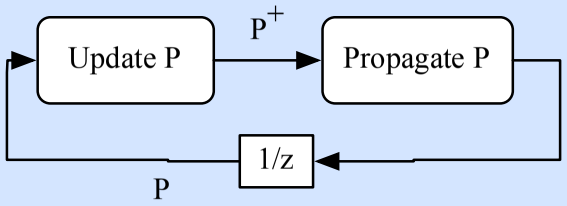
Step 1^+: After the second update step all the values in the covariance matrix have become very small because the velocity and position states are almost perfectly correlated. The 1+R position covariance from the previous step has resulted in a 2R velocity covariance on this step. The symbolic solution is not shown (due to its complexity) but the numerical solution continue to be conservative as seen in step 0^+.

Step 2: After the second propagation step the values have all grown as expected.

Step 2^+: After a third update step covariance shrinks as expected.

The Conventional and Joseph filters can fail as the problem becomes more ill-conditioned. Ill-conditioning can be produced a number of ways. Making the measurement more accurate (shrinking R) or increasing the values in the initial covariance matrix (P_0) can both cause one or more of these filters to fail. These two cases are examined in detail in sections 5.1.2 and 5.1.3.

Side-by-side comparison of Conventional, Joseph, Potter, Carlson, UD and Symbolic Filter Covariance



Initial Covariance Matrix $P_0 =$	$\begin{bmatrix} position = 1 & 0 \\ 0 & velocity = 1 \end{bmatrix}$
State Transition Matrix $\Phi =$	$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$
Measurement Sensitivity Matrix $H =$	$\begin{bmatrix} 1 & 0 \end{bmatrix}$
Position measurement Covariance $R = 4\,eps$ (eps = machine epsilon)	
Process noise $\omega \ll R$ and will be neglected unless it is the only term	

Step	Conv.	Joseph	Potter	Carl	UD	Symbolic
0	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
0^+	$\begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1-\frac{1}{1+R} & 0 \\ 0 & 1 \end{bmatrix}$
1	$\begin{bmatrix} 1+R & 1 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1+R & 1 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1+R & 1 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1+R & 1 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1+R & 1 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 2-\frac{1}{1+R} & 1 \\ 1 & 1 \end{bmatrix}$
1^+	$\begin{bmatrix} R & R \\ R & 2R \end{bmatrix}$	$\begin{bmatrix} R & R \\ R & 2R \end{bmatrix}$	$\begin{bmatrix} R & R \\ R & 2R \end{bmatrix}$	$\begin{bmatrix} R & R \\ R & 2R \end{bmatrix}$	$\begin{bmatrix} R & R \\ R & 2R \end{bmatrix}$	not calculated
2	$\begin{bmatrix} 5R & 3R \\ 3R & 2R \end{bmatrix}$	$\begin{bmatrix} 5R & 3R \\ 3R & 2R \end{bmatrix}$	$\begin{bmatrix} 5R & 3R \\ 3R & 2R \end{bmatrix}$	$\begin{bmatrix} 5R & 3R \\ 3R & 2R \end{bmatrix}$	$\begin{bmatrix} 5R & 3R \\ 3R & 2R \end{bmatrix}$	not calculated
2^+	$\begin{bmatrix} 5R/6 & R/2 \\ R/2 & R/2 \end{bmatrix}$	$\begin{bmatrix} 5R/6 & R/2 \\ R/2 & R/2 \end{bmatrix}$	$\begin{bmatrix} 5R/6 & R/2 \\ R/2 & R/2 \end{bmatrix}$	$\begin{bmatrix} 5R/6 & R/2 \\ R/2 & R/2 \end{bmatrix}$	$\begin{bmatrix} 5R/6 & R/2 \\ R/2 & R/2 \end{bmatrix}$	not calculated

Fig. 5.1: Case 1A: Filters operating nominally.

5.1.2 Case 1B: Numerical Failure Due to More Accurate Measurements

In this case (figure 5.2), the magnitude of R has been reduced from 4ϵ to $\epsilon/4$. Such an accurate measurement causes the conventional filter to encounter filter-lock, where one or more of the eigenvalues of the state covariance matrix have gone to zero.

Step 0: Initially, the covariance matrix P is the identity matrix.

Step 0^+: After the first update step the Conventional filter truncates the actual position covariance to zero, while the Joseph, Potter, Carlson and UD filters round to R. This is because the inverse required to solve for K in the Conventional and Joseph filters failed. Instead of outputting 1/(1+R) the matrix inverse algorithm outputted 1. This is easily overcome in the scalar case by replacing the matrix inverse algorithm with a scalar inverse. However, this is is common point of failure when matrix inverse methods are employed.

The symbolic solution for the position covariance is slightly smaller than R, thus the Joseph, Potter, Carlson and UD filters are conservative.

Step 1: After the first propagation step the covariance propagated by the Joseph form does not properly represent the effect of R. Because R$<\epsilon$, 1+R becomes 1. It should be noted, if R was equal to epsilon and not $\epsilon/4$, the Joseph filter would not fail at this point. Potter, Carlson and UD filters maintain R. Again, Potter, Carlson and UD filters are conservative (larger than necessary) compared to the symbolic solution.

Step 1^+: After the second update step the Conventional filter covariance goes to zero because states are perfectly correlated and R is rounded to zero. The Joseph filter covariance is also perfectly correlated, but R is better represented. The Potter, Carlson and UD 1+R term in step 1 results in 2R velocity covariance in step 1^+. While the symbolic solution is not shown, Potter, Carlson and UD results continue to be conservative.

Step 2: After the second propagation step process noise slowly starts to make the conventional covariance grow. Joseph covariance is smaller than it should be due to losing R during step 1.

Step 2$^+$: After a third update step the Joseph filter covariance is still smaller than it should be.

5.1.3 Case 1C: Numerical Failure Due to Larger Initial Covariance

Another way to create an ill-conditioned problem that causes the lower fidelity filters to fail is to increase the magnitude of the initial covariance matrix P. This is because machine epsilon is always with respect to a given value. By convention machine epsilon is usually given with respect to one, but each floating point number has a machine epsilon. For example, on a certain machine, eps(1)=222.0446e-18 while eps(10)=177.63568e-17. A larger initial covariance decreases a filters ability to correctly process an accurate measurement.

In this example (figure 5.3), the measurement covariance has been returned to 4ϵ (like Case 1A), but the P matrix has been increased to ten times the identity matrix.

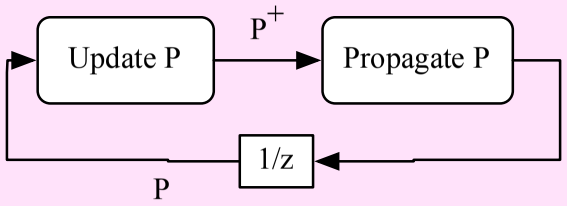
Step 0: Initially, the covariance matrix P is ten times the identity matrix.

Step 0$^+$: After the first update step there are already signs that the conventional filter is not working correctly, with the position covariance larger than the other solutions. This is a result of the inverse operation failing. Instead of rounding $\frac{1}{10+R}$ to 1/10, it has been rounded to $\frac{1}{10+R/64}$. This is a result of using the Cholesky inverse algorithm ($M^{-1} = (LL^*)^{-1}$ where L^* is the complex conjugate). If the LDL inverse algorithm ($M^{-1} = (LDL^*)^{-1}$) is used, then the inverse is rounded to 1/10, in the same way as case B.

The symbolic solution for the position covariance is slightly smaller than R, thus the Joseph, Potter, Carlson and UD filters are conservative.

Step 1: After the first propagation step both the Conventional and Joseph filters are unable to represent the effect of R, even though R>ϵ. 10+R becomes 10 because ϵ is with

Side-by-side comparison of Conventional, Joseph, Potter, Carlson, UD and Symbolic Filter Covariance



Initial Covariance Matrix $P_0 =$	$\begin{bmatrix} position = 1 & 0 \\ 0 & velocity = 1 \end{bmatrix}$
State Transition Matrix $\Phi =$	$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$
Measurement Sensitivity Matrix $H =$	$\begin{bmatrix} 1 & 0 \end{bmatrix}$
Position measurement Covariance $R = eps/4$ (eps = machine epsilon)	
Process noise $\omega \ll R$ and will be neglected unless it is the only term	

Step	Conv.	Joseph	Potter	Carl	UD	Symbolic
0	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
0^+	$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1-\frac{1}{1+R} & 0 \\ 0 & 1 \end{bmatrix}$
1	$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1+R & 1 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1+R & 1 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1+R & 1 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 2-\frac{1}{1+R} & 1 \\ 1 & 1 \end{bmatrix}$
1^+	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} R & R \\ R & R \end{bmatrix}$	$\begin{bmatrix} R & R \\ R & 2R \end{bmatrix}$	$\begin{bmatrix} R & R \\ R & 2R \end{bmatrix}$	$\begin{bmatrix} R & R \\ R & 2R \end{bmatrix}$	not calculated
2	$\begin{bmatrix} \omega^2 & 0 \\ 0 & \omega^2 \end{bmatrix}$	$\begin{bmatrix} 4R & 2R \\ 2R & R \end{bmatrix}$	$\begin{bmatrix} 5R & 3R \\ 3R & 2R \end{bmatrix}$	$\begin{bmatrix} 5R & 3R \\ 3R & 2R \end{bmatrix}$	$\begin{bmatrix} 5R & 3R \\ 3R & 2R \end{bmatrix}$	not calculated
2^+	$\begin{bmatrix} \omega^2 & 0 \\ 0 & \omega^2 \end{bmatrix}$	$\begin{bmatrix} 4R/5 & 2R/5 \\ 2R/5 & R/5 \end{bmatrix}$	$\begin{bmatrix} 5R/6 & R/2 \\ R/2 & R/2 \end{bmatrix}$	$\begin{bmatrix} 5R/6 & R/2 \\ R/2 & R/2 \end{bmatrix}$	$\begin{bmatrix} 5R/6 & R/2 \\ R/2 & R/2 \end{bmatrix}$	not calculated

Fig. 5.2: Case 1B: Filter failure due to increased measurement accuracy.

respect to 1, not 10. Potter, Carlson and UD filters maintain the effect of R on the state covariance. Again, Potter, Carlson and UD filters are conservative (larger than necessary) compared to the symbolic solution.

Step 1^+: After the second update step the conventional filter covariance is in error and is no longer symmetric! This is a direct result of numerical errors like those seen in step 0^+. With the inverse being incorrectly calculated by the Cholesky inverse algorithm. Use of a LDL inverse algorithm would have maintained symmetry, but would have resulted in filter-lock. The Joseph filter has incorrectly calculated the velocity covariance because it unable to represent the effect of R during step 1. While the symbolic solution is not shown, Potter, Carlson and UD results continue to be conservative.

Step 2: After the second propagation step, errors in the Conventional and Joseph filters continue to exist. Potter, Carlson and UD filters continue to perform correctly.

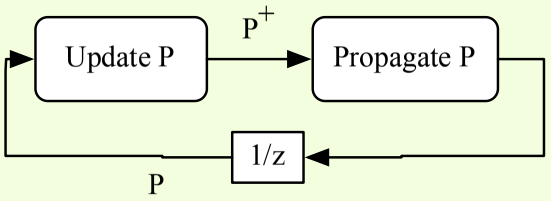
Step 2^+: After a third update step the Conventional and Joseph filters continue to have errors. Potter, Carlson and UD filters continue to perform correctly.

If the initial covariance matrix was 100 times the identity matrix, then the behavior of the filters would exactly match case B.

5.1.4 Case 1D: Filter Lock with Virtually No Process-Noise

In this case, as seen in table 5.1, the process noise (w) has been set to machine epsilon squared, thus, the noise strength (Q) is ϵ^4 which is virtually zero. The measurements noise (ν) has been set to ϵ, thus the measurement covariance (R) is ϵ^2. As shown in Figures 5.4 and 5.6, the highly accurate measurement causes the conventional filter to completely ignore future measurements, while the Potter filter continues to operate correctly.

Side-by-side comparison of Conventional, Joseph, Potter, Carlson, UD and Symbolic Filter Covariance



Initial Covariance Matrix $P_0 =$	$\begin{bmatrix} position = 10 & 0 \\ 0 & velocity = 10 \end{bmatrix}$
State Transition Matrix $\Phi =$	$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$
Measurement Sensitivity Matrix $H = \begin{bmatrix} 1 & 0 \end{bmatrix}$	
Position measurement Covariance $R = 4\,eps$ (eps = machine epsilon)	
Process noise $\omega \ll R$ and will be neglected unless it is the only term	

Step	Conv.	Joseph	Potter	Carl	UD	Symbolic
0	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$
0^+	$\begin{bmatrix} 5R/4 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} R & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} R & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} R & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} R & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 10 - \frac{100}{10+R} & 0 \\ 0 & 10 \end{bmatrix}$
1	$\begin{bmatrix} 10 & 10 \\ 10 & 10 \end{bmatrix}$	$\begin{bmatrix} 10 & 10 \\ 10 & 10 \end{bmatrix}$	$\begin{bmatrix} 10+R & 10 \\ 10 & 10 \end{bmatrix}$	$\begin{bmatrix} 10+R & 10 \\ 10 & 10 \end{bmatrix}$	$\begin{bmatrix} 10+R & 10 \\ 10 & 10 \end{bmatrix}$	$\begin{bmatrix} 20 - \frac{100}{10+R} & 10 \\ 10 & 10 \end{bmatrix}$
1^+	$\begin{bmatrix} 2.5R & 2.5R \\ 2R & 4R \end{bmatrix}$	$\begin{bmatrix} R & R \\ R & R \end{bmatrix}$	$\begin{bmatrix} R & R \\ R & 2R \end{bmatrix}$	$\begin{bmatrix} R & R \\ R & 2R \end{bmatrix}$	$\begin{bmatrix} R & R \\ R & 2R \end{bmatrix}$	not calculated
2	$\begin{bmatrix} 11 & 6.5 \\ 6 & 4 \end{bmatrix}$	$\begin{bmatrix} 4R & 2R \\ 2R & R \end{bmatrix}$	$\begin{bmatrix} 5R & 3R \\ 3R & 2R \end{bmatrix}$	$\begin{bmatrix} 5R & 3R \\ 3R & 2R \end{bmatrix}$	$\begin{bmatrix} 5R & 3R \\ 3R & 2R \end{bmatrix}$	not calculated
2^+	$\begin{bmatrix} \frac{11R}{12} & \frac{13R}{24} \\ \frac{R}{2} & \frac{3R}{4} \end{bmatrix}$	$\begin{bmatrix} 4R/5 & 2R/5 \\ 2R/5 & R/5 \end{bmatrix}$	$\begin{bmatrix} 5R/6 & R/2 \\ R/2 & R/2 \end{bmatrix}$	$\begin{bmatrix} 5R/6 & R/2 \\ R/2 & R/2 \end{bmatrix}$	$\begin{bmatrix} 5R/6 & R/2 \\ R/2 & R/2 \end{bmatrix}$	not calculated

Fig. 5.3: Case 1C: Filter failure due to increased initial covariance magnitude.

Table 5.1: Model Information for Case 1D: Filter Lock When Process Noise 1σ is ϵ^2

Initial Conditions	Symbol	Truth Model	Filter Model
state	$\begin{bmatrix} r_0 & v_0 \end{bmatrix}^T$	$\begin{bmatrix} 0 & 0 \end{bmatrix}^T$	$\begin{bmatrix} 0 & 0 \end{bmatrix}^T$
state covariance	P_0		eye(2)
Model Dynamics	**Symbol**	**Truth Model**	**Filter Model**
state dynamics	$\dot{x}$	$\dot{x} = Fx + Bw$	$\dot{x} = Fx$
state covariance dynamics	P_{t+dt}		$P_{t+dt} = function(P_t, F, B, \omega)$
Linear Model	F	[0 1; 0 0]	[0 1; 0 0]
Noise input matrix	B	eye(2)	eye(2)
Process Noise standard deviation	w	$\begin{bmatrix} \epsilon^2 & \epsilon^2 \end{bmatrix}^T$	$\begin{bmatrix} \epsilon^2 & \epsilon^2 \end{bmatrix}^T$
Sensor Info	**Symbol**	**Truth Model**	**Filter Model**
position sensor	$\tilde{y}$	$\widetilde{y} = H\,\bar{x} + \nu$	$\hat{y} = H\hat{x}$
measurement sensitivity matrix	H	[1 0]	[1 0]
measurement noise standard deviation	v	ϵ	ϵ

5.1.5 Case 1E: Filter Lock with Low Process-Noise

If the magnitude of the process noise is increased, as seen in table 5.2, the initial filter-lock will be overcome because process noise increases the magnitude of the eigenvalues of the covariance matrix (see Figures 5.5 and 5.7).

Table 5.2: Model Information for Case 1E: Filter Lock When Process Noise 1σ is $\epsilon/100$

Initial Conditions	Symbol	Truth Model	Filter Model
state	$\begin{bmatrix} r_0 & v_0 \end{bmatrix}^T$	$\begin{bmatrix} 0 & 0 \end{bmatrix}^T$	$\begin{bmatrix} 0 & 0 \end{bmatrix}^T$
state covariance	P_0		eye(2)
Model Dynamics	**Symbol**	**Truth Model**	**Filter Model**
state dynamics	$\dot{x}$	$\dot{x} = Fx + Bw$	$\dot{x} = Fx$
state covariance dynamics	P_{t+dt}		$P_{t+dt} = function(P_t, F, B, \omega)$
Linear Model	F	[0 1; 0 0]	[0 1; 0 0]
Noise input matrix	B	eye(2)	eye(2)
Process Noise standard deviation	w	$\frac{\begin{bmatrix} \epsilon & \epsilon \end{bmatrix}^T}{100}$	$\frac{\begin{bmatrix} \epsilon & \epsilon \end{bmatrix}^T}{100}$
Sensor Info	**Symbol**	**Truth Model**	**Filter Model**
position sensor	$\tilde{y}$	$\widetilde{y} = H\,\bar{x} + \nu$	$\hat{y} = H\hat{x}$
measurement sensitivity matrix	H	[1 0]	[1 0]
measurement noise	v	ϵ	ϵ

5.1.6 Other Failure Modes

In addition to these failure modes, the Joseph formulation can be as susceptible to numerical errors as the Conventional filter if a measurement is correlated to more then one state (e.g. H=[1 1]). See [19, example 7.11] for more details.

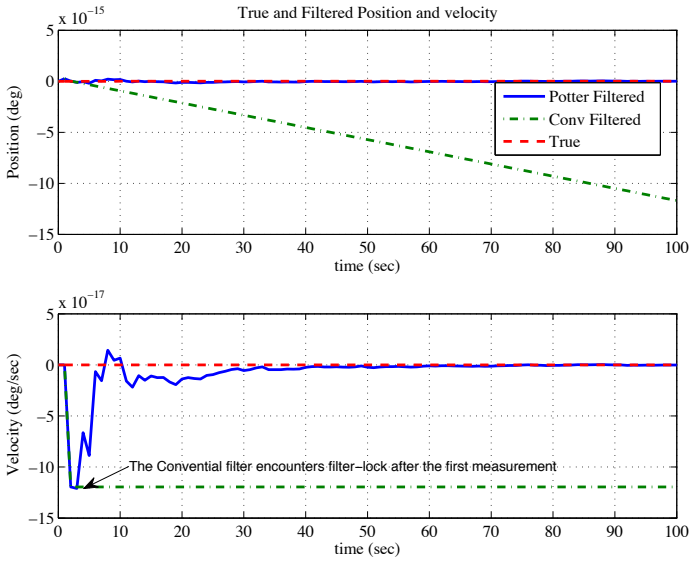


Fig. 5.4: Case 1D: Filter estimates of position and velocity when process noise 1σ is ϵ^2.

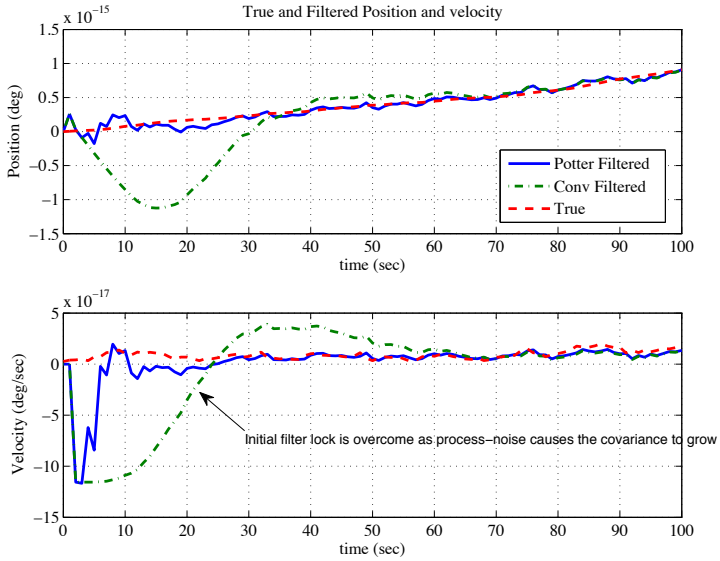


Fig. 5.5: Case 1E: Filter estimates of position and velocity when process noise 1σ is $\epsilon/100$.

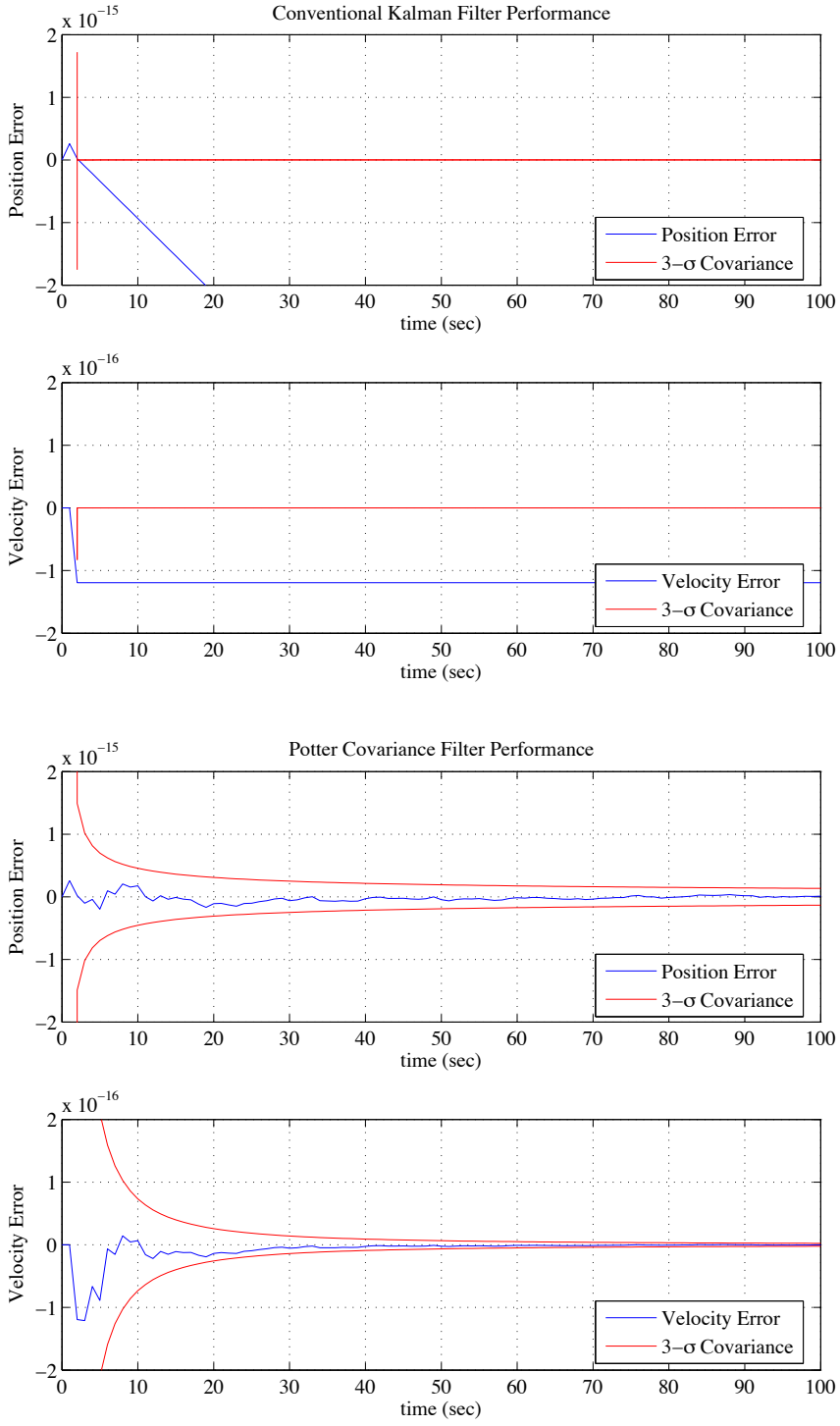


Fig. 5.6: Case 1D: Conventional vs Potter filter performance when process noise 1σ is ϵ^2.

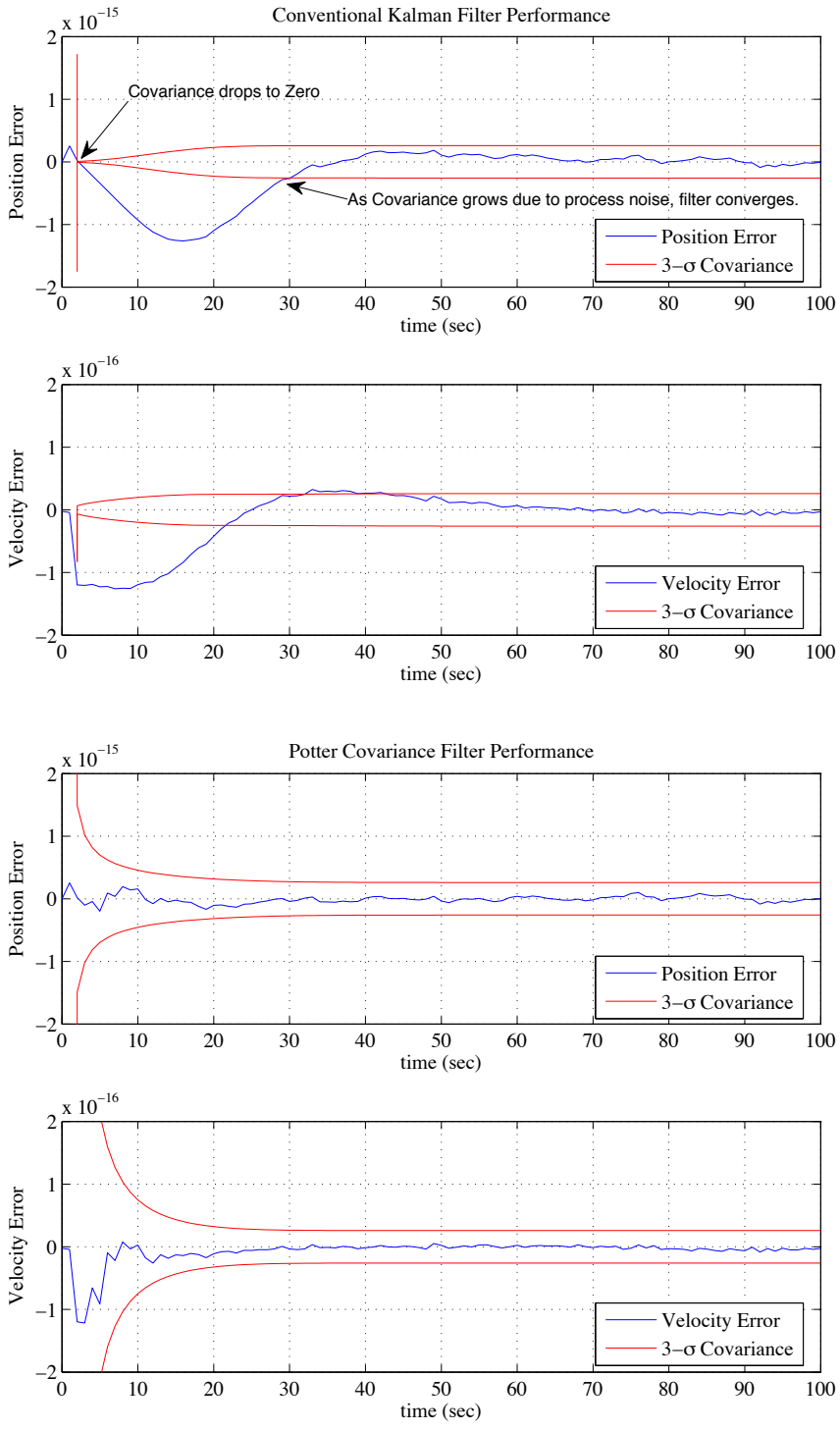


Fig. 5.7: Case 1E: Conventional vs Potter filter performance when process noise 1σ is $\epsilon/100$.

5.2 Toy Case 2: Two Rotating Shafts with Relative Measurement

The following case involves a filter estimating the absolute state (position and velocity) of two "rotating" shafts (see table 5.3). However, only a relative position measurement is to be processed. In this scenario, the absolute state covariance continues to grow without bound, while the correlation between the two rotating shaft's positions and velocities becomes higher and higher.

Under these conditions the high correlation between the positions and velocities results in an ill-conditioned matrix. This is most easily seen by looking at the relative state error and covariance. The relative state is defined by:

$$\begin{bmatrix} r_{rel} \\ v_{rel} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} r2_{inertial} \\ v1_{inertial} \\ r2_{inertail} \\ v2_{inertial} \end{bmatrix} \tag{5.1}$$

The relative covariance matrix is defined by:

$$\begin{bmatrix} Pr_{rel} & 0 \\ 0 & Pv_{rel} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} P_{inertial} \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.2}$$

In this case, the Conventional and Joseph filters encounter filter lock (in terms of the relative covariance) within the first 250 seconds as seen in Figure 5.8. The consequence of this filter lock is devastating, as seen in figure 5.9. The Potter, Carlson, and UD filters continue to estimate the relative state well (see figure 5.10) and their relative state estimates differ only on the order of machine precision (see figure 5.11).

Table 5.3: Model Information for Case 2: Relative Measurements

Initial Conditions	Symbol	Truth and/or Filter Model
state	$\begin{bmatrix} r1_0 & v1_0 & r2_0 & v2_0 \end{bmatrix}^T$	$\begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^T$
state covariance	P_0	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & .01 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & .01 \end{bmatrix}$
Model Dynamics	**Symbol**	**Truth and/or Filter Model**
state dynamics	$\dot{x}$	$\dot{x} = Fx + Bw$ / $\dot{x} = Fx$
state covariance dynamics	P_{t+dt}	$P_{t+dt} = function(P_t, F, B, \omega)$
Linear Model	F	$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$
Noise input matrix	B	eye(4)
Process Noise standard deviation	w	$[eps;\ eps;\ eps;\ eps] \times 1e4$
Sensor Info	**Symbol**	**Truth and/or Filter Model**
position sensor	$\tilde{y}$	$\widetilde{y} = H\,\bar{x} + \nu$ / $\hat{y} = H\hat{x}$
measurement sensitivity matrix	H	[-1 0 1 0]
measurement noise standard deviation	v	$\sqrt{eps} \times 100$

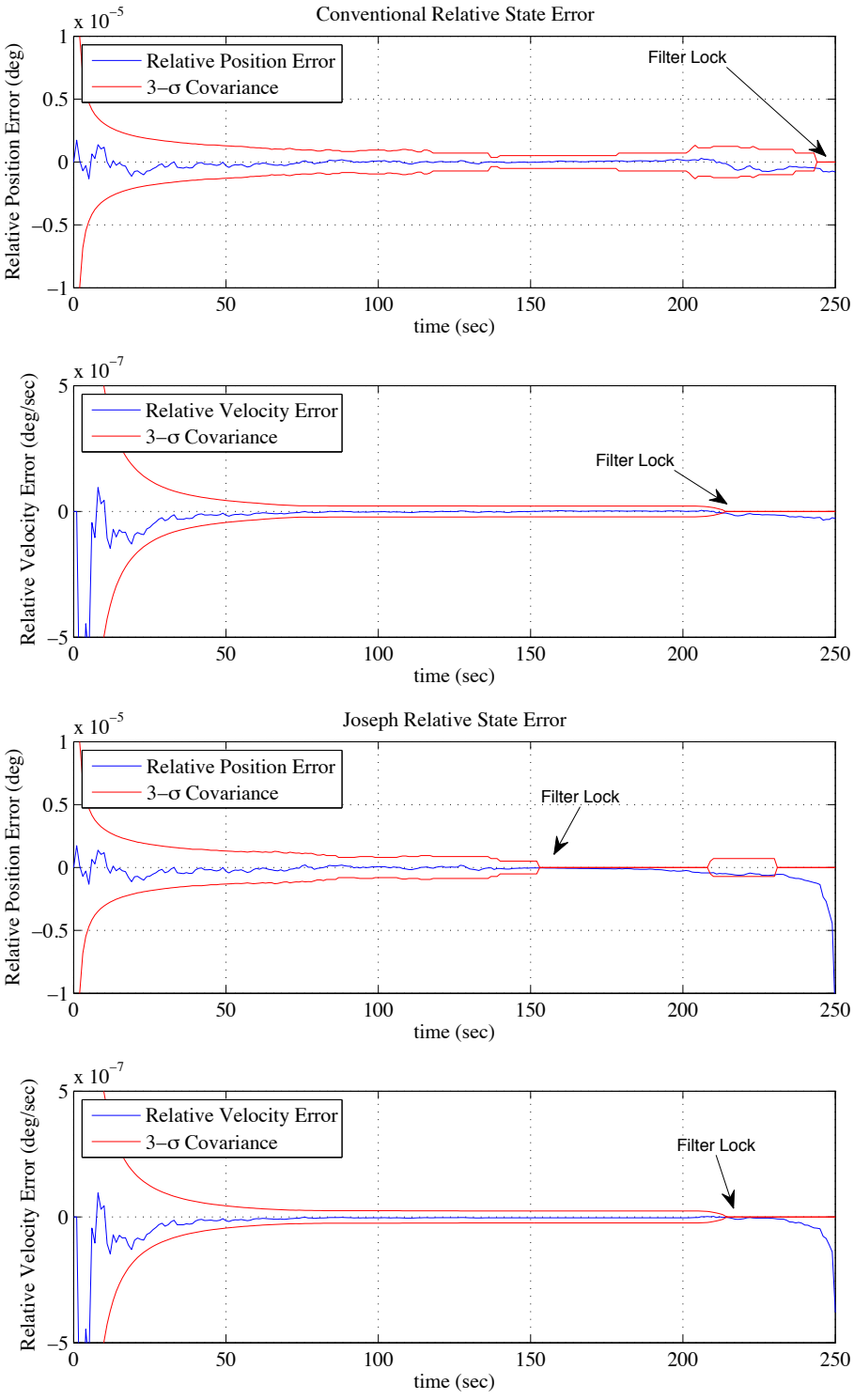


Fig. 5.8: Case 2: Filter lock in Conventional and Joseph filters.

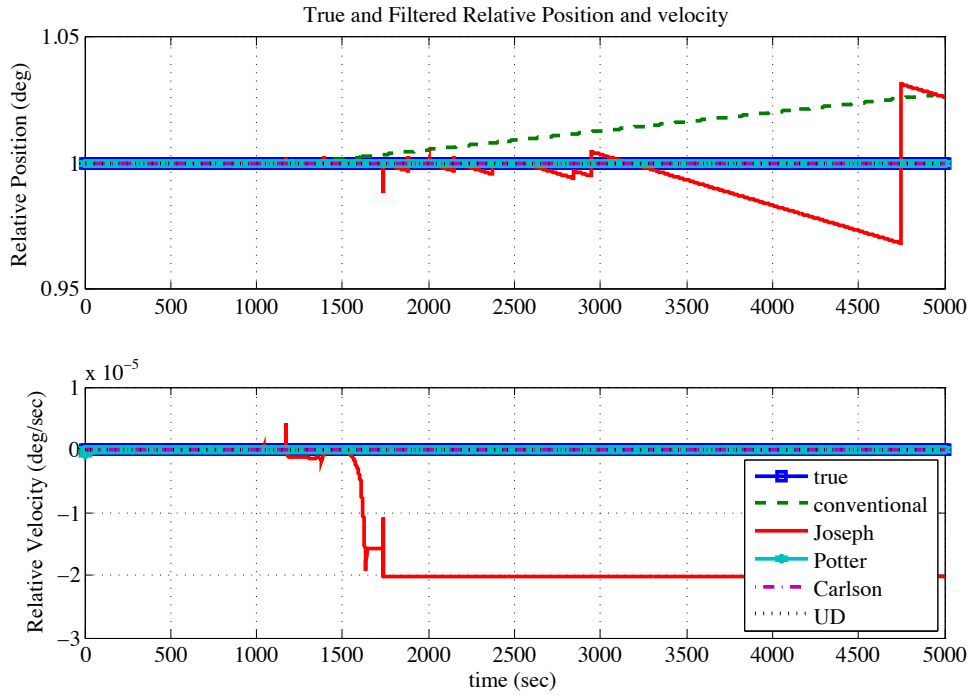


Fig. 5.9: Case 2: Effect of filter lock on estimation error.

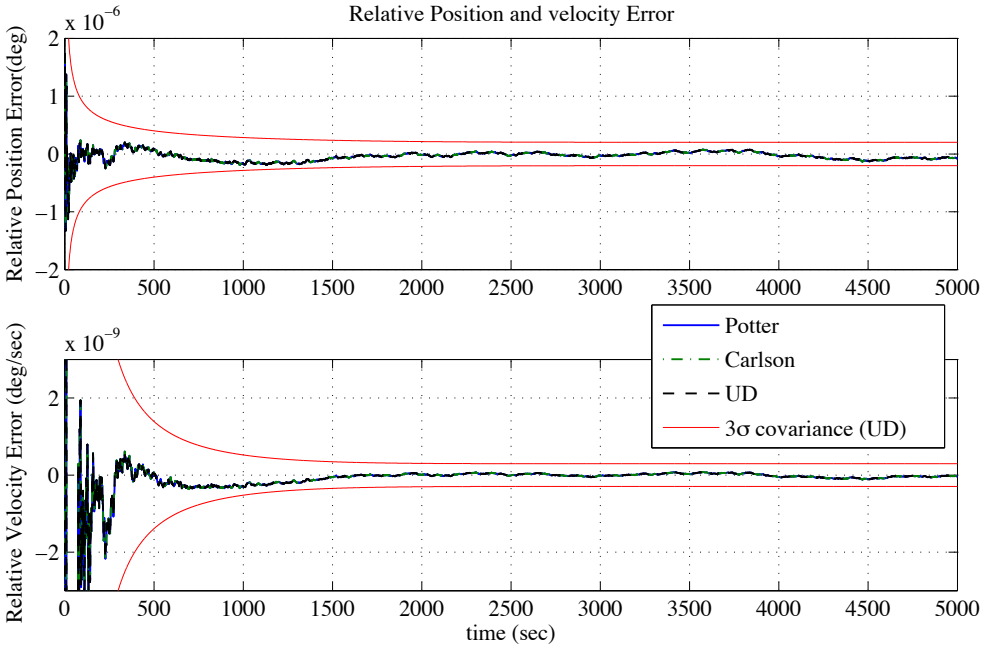


Fig. 5.10: Case 2: No filter lock in Potter, Carlson, and UD filters.

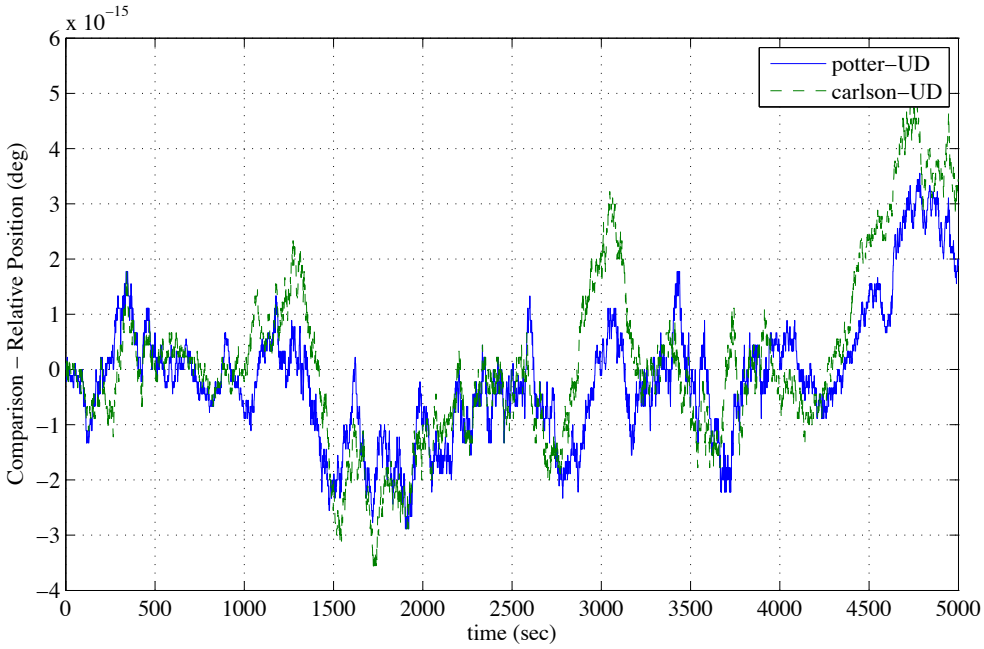


Fig. 5.11: Case 2: Difference between Potter, Carlson, and UD filter relative state estimates during a 5,000 sec run. Note that differences are on the order of machine epsilon.

Performance Comparison: Potter/Carlson and the UD filter

Up to this point, no significant difference has been detected between the Potter, Carlson, and UD factorization filters. With a relative state measurement, however, numerical differences between the filters do show up. Figure 5.12 plots both the Potter, Carlson, and UD absolute states and the true absolute states. In this case, because the initial angular position and velocity are zero, all the states should stay at zero. It is clear that the UD factorization filter performs better than either the Potter or Carlson filters. The Potter and Carlson filters are corrupting the initial state when they process relative state measurements. Comparing figure 5.12 with figure 5.13, shows that the UD filter is outperforming both the Potter and the Carlson filters by a couple orders of magnitude.

This difference can probably be attributed to how the Kalman gain (K) is calculated. As seen in the flowcharts in figure 4.3, the Potter filter Kalman gain $K = bSa$, where b is a scalar, S is a matrix, and a is a vector. In the Carlson filter, $K = e_n/d_n$ where e_n is an

element of a vector calculated by $e_k = e_{k-1} + S_k^- a_k$. S_k^- is a matrix and a_k is a vector. Both of these formulations seem to be more susceptible to numerical errors due to matrix multiplication. The UD factorization filter stays with scalar calculations with $K = b/a_n$, where b is a vector calculated by iterating through $b_j = b_{j\,old} + U_{jk} v_k$.

These errors, however, are much, much smaller than the covariance of these states. Thus, real world gains of the UD filter over the Potter and Carlson filters due to these differences are most likely undetectable.

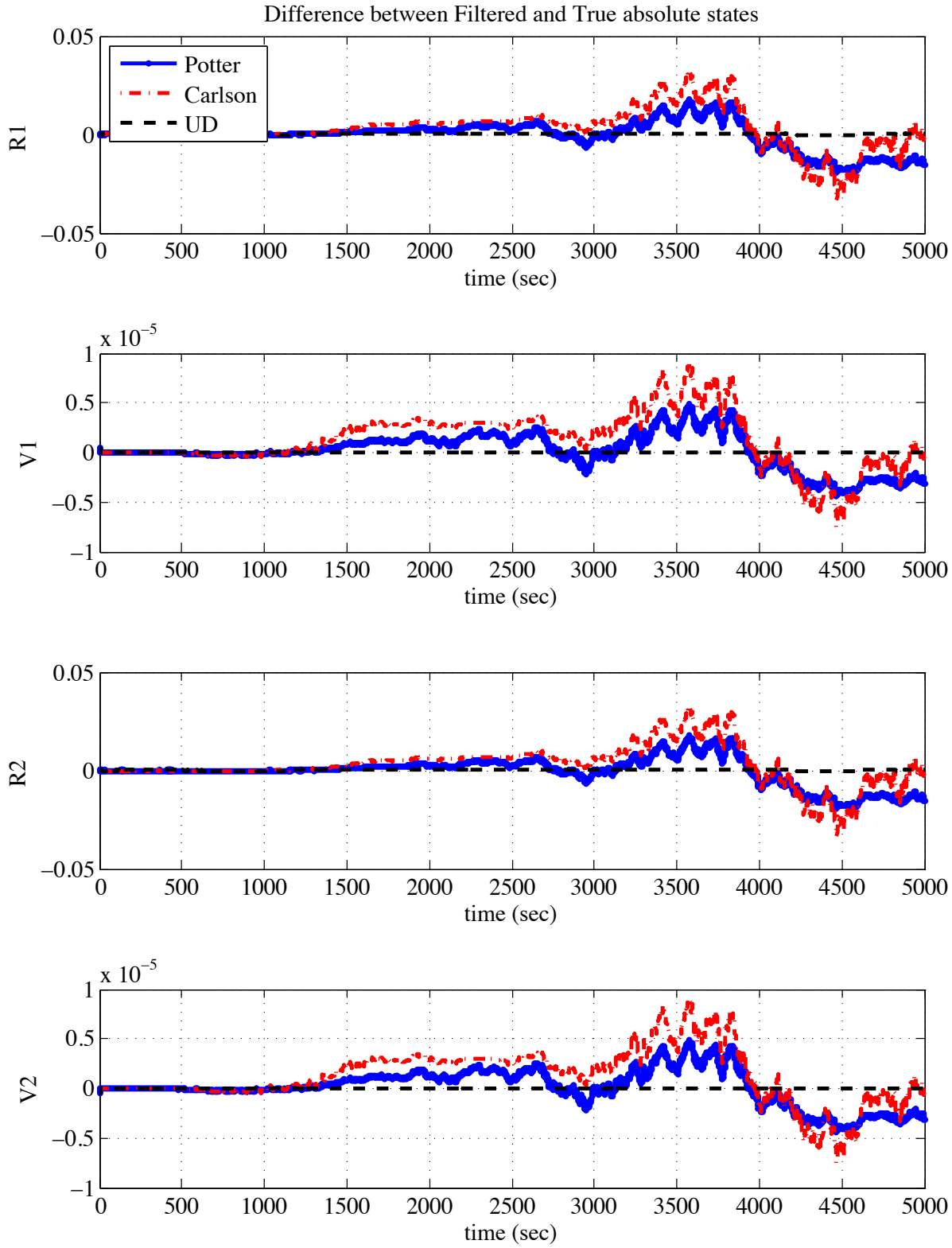


Fig. 5.12: Filtered absolute states minus true absolute states.

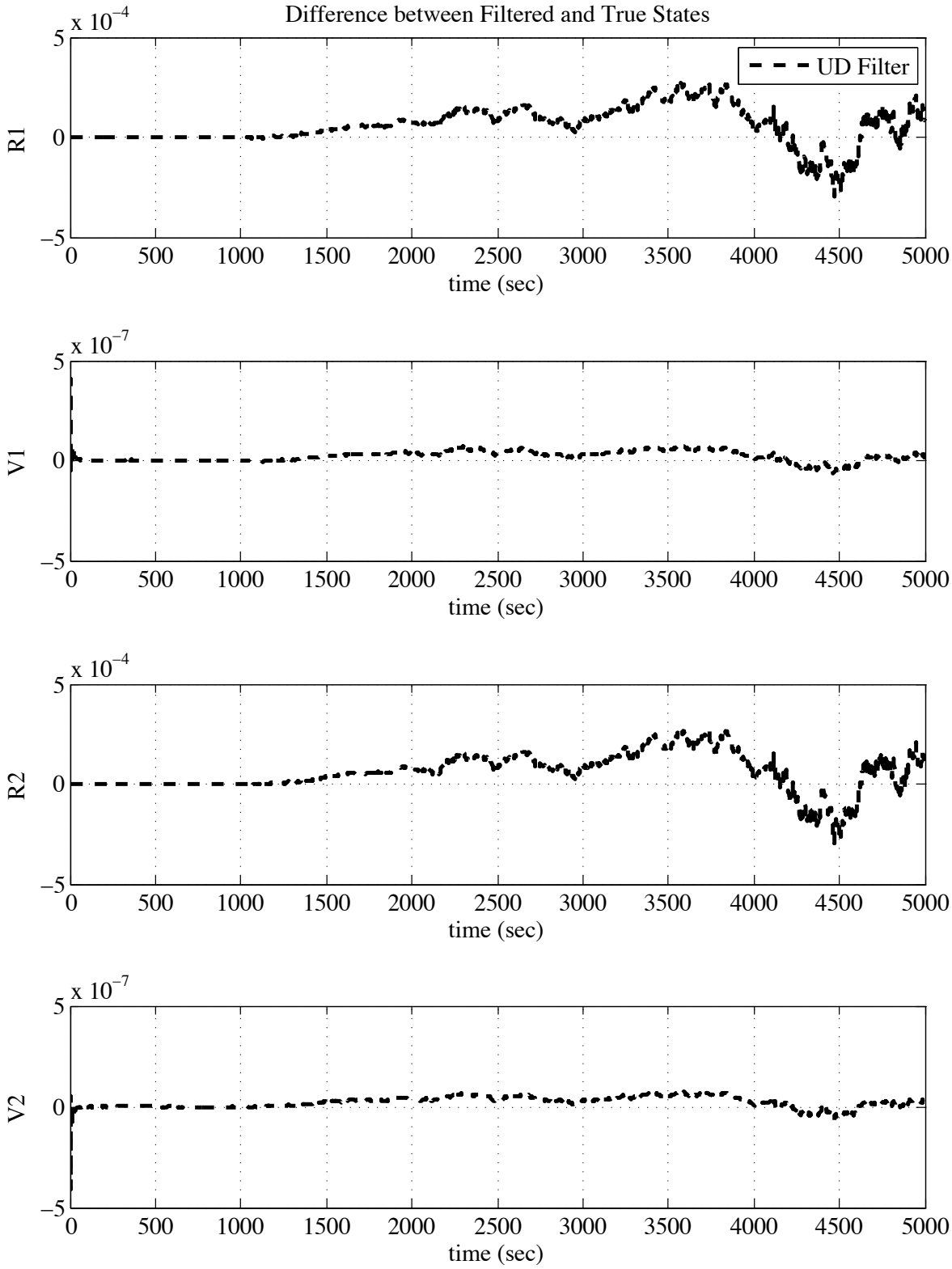


Fig. 5.13: UD filter absolute states minus true absolute states.

Chapter 6

Orbital Rendezvous Simulation Model Development

The high fidelity, six degree-of-freedom simulation consists of two spacecraft (a "chaser" and "target") in low-Earth orbit. The simulation includes sensor, actuator, and dynamic models that include noise, bias and other errors. Appropriate attitude and translational controllers were built for the "chaser" spacecraft. Since this simulation is not the main focus of the thesis, only a basic overview of the key models and equations is included here. Unusual symbols unique to this chapter are summarized in Appendix A. Initial conditions, constants, and sensor noise and bias specifications are covered in section 8.2.

6.1 Simulation Overview

A quick overview of the satellite simulation may be seen in figure 6.1. The labeled blocks are implementations of models covered in this chapter.

6.2 Vehicle Dynamics

The chaser spacecraft consists of a 1x1x1 meter cube with a mass of 15 kg and and inertia matrix:

$$\mathbb{I} = \begin{bmatrix} 0.8 & 0.05 & 0.05 \\ 0.05 & 0.8 & 0.05 \\ 0.05 & 0.05 & 0.8 \end{bmatrix} kg\,m^2 \tag{6.1}$$

6.2.1 Translation Dynamics

While translational dynamics is covered in more detail in Chapter 7, J_2 gravity, thruster accelerations, and process noise were summed and integrated to obtain the spacecraft's position and velocity in the ECI frame. Thus, the acceleration of the spacecraft was calculated

A Sections 6.2 and 6.3

B Section 6.5

C Section 6.4

D Section 6.7 and Chapters 4 and 7

E Section 6.6

F Section 6.8

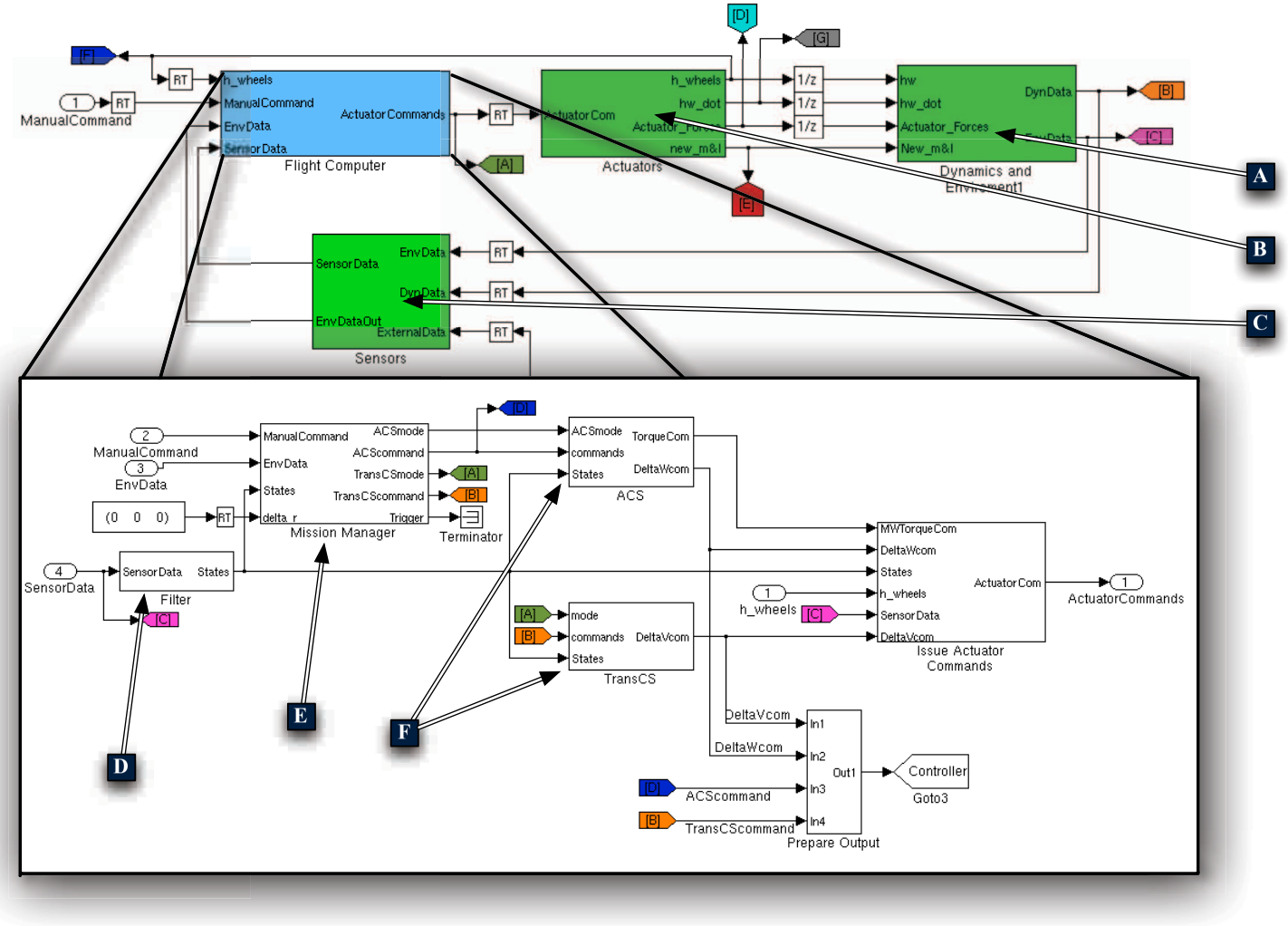


Fig. 6.1: Overview of satellite simulation.

by:

$$\begin{aligned} \bar{a} &= -\mu\frac{\bar{r}}{|\bar{r}|^3} - \mu\frac{J_2 R_e^2}{2|\bar{r}|^5}\left\{6(\bar{r}\cdot\bar{n})\bar{n} + 3\bar{r} - 15(\bar{i}_r\cdot\bar{n})^2\bar{r}\right\} + \frac{\bar{F}_{thrust}}{mass} + \bar{w}_{acc} \qquad (6.2)\\ \bar{r} &= \text{ ECI position, } \bar{i}_r = \text{unit vector form of } r,\ \bar{F}_{thrust} = \text{thrust vector}\\ mass &= mass_o - (\dot{mass})t,\ \bar{w}_{acc} = \text{process noise with strength of } 1\times 10^{-12} m^2/s^3 \end{aligned}$$

$\mu,\ J_2,\ R_e,$ and $\bar{n}$ are defined in table 8.7.

6.2.2 Attitude Dynamics

Traditional Euler equations and quaternion propagation in conjunction with the Simulink integrators was used to propagate the attitude of the spacecraft. Thus the attitude acceleration was calculated by

$$\dot{\bar{\omega}} = \mathbb{I}^{-1}\left\{-\bar{\omega}\times\left(\mathbb{I}\bar{\omega} + \bar{h}\right) + \bar{\mathcal{T}} + \dot{\bar{h}}\right\} \qquad (6.3)$$

$$\omega = \text{ angular rate, } \bar{h} = \text{angular momentum of momentum wheels} \qquad (6.4)$$

The torque ($\bar{\mathcal{T}}$) is a function of the thrusters being fired, the actuation of the momentum wheels, and process noise with a strength of $1\times 10^{-12}{}^{(N\,m)^2}/_{sec}$.

Integrating angular acceleration yields angular velocity $\bar{\omega}$ which can be written in terms of a quaternion derivative:

$$\dot{\bar{q}} = \frac{1}{2}\mathbf{\Omega}(\bar{\omega})\bar{q} \qquad (6.5)$$

where

$$\mathbf{\Omega}(\bar{\omega}) = \begin{bmatrix} -[\bar{\omega}\times] & \bar{\omega} \\ -\bar{\omega}^T & 0 \end{bmatrix} \qquad (6.6)$$

$$[\bar{\omega}\times] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \qquad (6.7)$$

This quaternion derivative is also integrated to yield the attitude in terms of a quaternion.

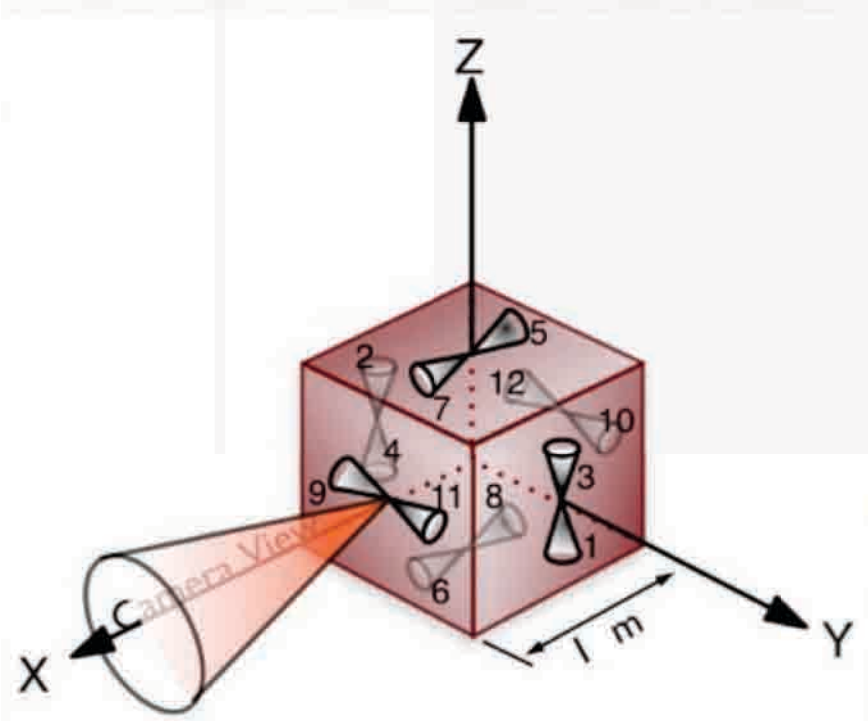


Fig. 6.2: Chaser satellite thruster configuration.

6.3 Environment Models

No atmospheric drag, or solar pressure models were implemented in this simulation. However, random acceleration and torque effects are included as noted in equations 6.2 and 6.3.

6.4 Sensor Models

The onboard sensors include three-axis accelerometers, a line-of-sight camera for observing the target satellite, and a star-camera for determining attitude. The specifications for the accelerometers and LOS camera are found in tables 8.2 and 8.3 on pages 68 and 68.

6.4.1 Accelerometers

The accelerometers provide a measurement of all non-gravitational forces corrupted by misalignment (ϵ), noise (η_{acc}) and bias (β_{acc}) and quantization effects with $1 \times 10^{-9} m/s^2$ resolution. The actual values for misalignment and noise are dependent on the specific run.

$$\tilde{a} = quant(\boldsymbol{E}_{ortho}\bar{a}_{non-grav} + \bar{\eta}_{acc} + \bar{\beta_{acc}}) \tag{6.8}$$

$$\boldsymbol{E}_{ortho} = \begin{bmatrix} 0 & \epsilon_z & -\epsilon_y \\ -\epsilon_z & 0 & \epsilon_x \\ \epsilon_y & -\epsilon_x & 0 \end{bmatrix} \tag{6.9}$$

6.4.2 Line-of-sight Camera

The line-of-sight camera provided the tangent of the azimuth and elevation angles corrupted by misalignment and noise. The actual values for misalignment (β_{LOS}) and noise (η_{LOS}) are dependent on the specific run. R^{cam} is the same as R_{rel} defined in figure 7.2.

$$tan(el) = tan\left(asin\left(\frac{R_z^{cam}}{|R^{cam}|}\right) + \eta_{LOS} + \beta_{LOS}\right) \tag{6.10}$$

$$tan(az) = tan\left(atan2\left(\frac{R_y^{cam}}{|R^{cam}|}, \frac{R_x^{cam}}{|R^{cam}|}\right) + \eta_{LOS} + \beta_{LOS}\right) \tag{6.11}$$

6.4.3 Star-Camera

For this simulation, a perfect Star-Camera was used to define orientation of the body frame. The effect of any non-zero camera errors is easily modeled as LOS-camera error and accelerometer misalignment. Thus, the star-camera measurement defines the true attitude of the chaser.

6.5 Actuator Models

Actuators for the chaser satellite include twelve thrusters for translation and attitude control, and four momentum wheels for more precise attitude control.

6.5.1 Thrusters

The 12 thrusters on the chaser satellite are located a half meter from the center of mass and are configured as seen in figure 6.2. The force of each thruster depends on the scenario being run (see section 8.7). The orientation and position of the thrusters are described by the matrices below. The $\boldsymbol{F}^c$ matrix contains unit vectors pointing in the direction of the

force from each thruster. The $\boldsymbol{R}^c$ matrix defines the position of each thruster in the chaser body frame.

$$\boldsymbol{F}^c = -\begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix} \tag{6.12}$$

$$\boldsymbol{R}^c = \begin{bmatrix} 0 & 0.5 & 0 \\ 0 & -0.5 & 0 \\ 0 & 0.5 & 0 \\ 0 & -0.5 & 0 \\ 0 & 0 & 0.5 \\ 0 & 0 & -0.5 \\ 0 & 0 & 0.5 \\ 0 & 0 & -0.5 \\ 0.5 & 0 & 0 \\ -0.5 & 0 & 0 \\ 0.5 & 0 & 0 \\ -0.5 & 0 & 0 \end{bmatrix} m \tag{6.13}$$

Note that the thrusters are in pairs and can be used to generate torque and force effects. The thrusters are either on or off, and the magnitude and direction of the force and torque on the spacecraft due to the thrusters is modeled by:

$$\bar{F}_{thrust} = (\boldsymbol{I} + \boldsymbol{E}_{ortho})(\boldsymbol{I} + \boldsymbol{S})\bar{F}_{nominal} + \bar{w}_{force} + \bar{\beta}_{force} \tag{6.14}$$

$$\bar{\mathcal{T}}_{thrust} = (\boldsymbol{I} + \boldsymbol{E}_{ortho})(\boldsymbol{I} + \boldsymbol{S})\bar{\mathcal{T}}_{nominal} + \bar{w}_{torque} + \bar{\beta}_{torque} \tag{6.15}$$

$\boldsymbol{E}_{ortho}$ is defined in equation 6.8, $\boldsymbol{S}$ is the identity matrix multiplied by the scale factor. Pertinent values in the above equations are defined in tables 6.1 and 6.2.

Table 6.1: Thruster Force Model Specifications

Thruster Force model Specification		
Spec	Value	Units
Noise Strength	1×10^{-8}	$(N\,s)^2/s$
Bias Variance	1×10^{-8}	$(N)^2$
Bias Time Constant	60	sec
Misalignment Standard Deviation	20 arcseconds	rad
Scale factor	100×10^{-6}	PPM

Table 6.2: Thruster Torque Model Specifications

Thruster Torque model Specification		
Spec	Value	Units
Noise Strength	1×10^{-6}	${}^{(N\,m\,s)^2}/_{s}$
Bias Variance	1×10^{-6}	$(N\,m)^2$
Bias Time Constant	60	sec
Misalignment Standard Deviation	20 arcseconds	rad
Scale factor	100×10^{-6}	PPM

6.5.2 Momentum Wheels

The moment wheel system is actually four wheels that work together to generate torques around the three primary axes. These momentum wheels are arranged according to figure 6.3. The controller and model is based on work found in [22, Chapter 7 section 3]. For a given torque command $[T_{cx}\ T_{cy}\ T_{cz}]^T$, the individual wheel torque commands are calculated according to

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & \frac{1}{2} & -\frac{1}{2} \\ -1 & 0 & \frac{1}{2} & -\frac{1}{2} \\ 0 & -1 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} T_{cx} \\ T_{cy} \\ T_{cz} \\ 0 \end{bmatrix} - [h_1 - h_2 + h_3 - h_4] \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} K_{momMan} \tag{6.16}$$

where the momentum management gain (K_{momMan}) was set to 0.05 sec^{-1}.

The resultant torque around each axis is calculated by

$$\begin{bmatrix} T_{cx} \\ T_{cy} \\ T_{cz} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \tag{6.17}$$

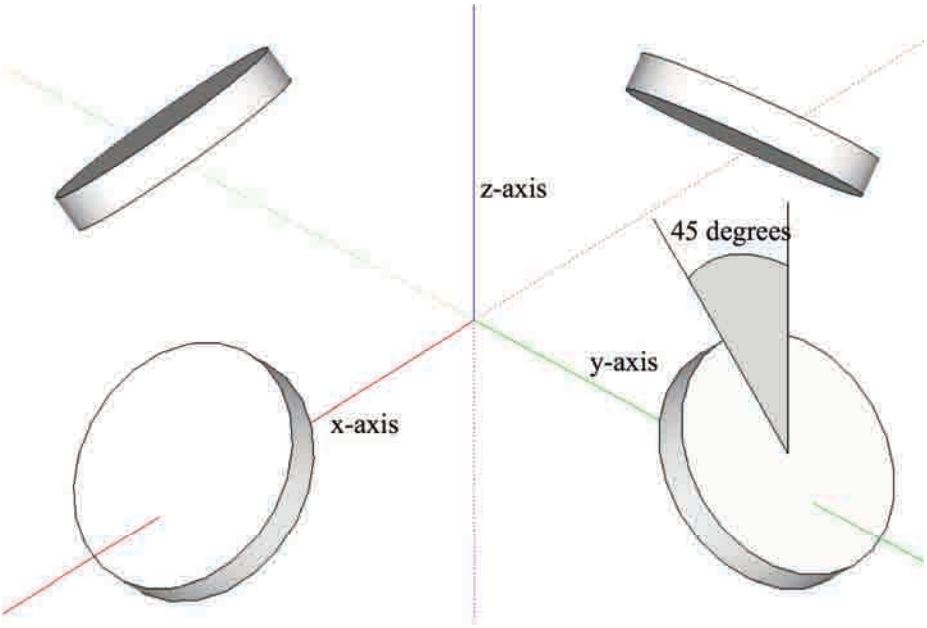


Fig. 6.3: Momentum wheel orientation.

6.6 Translation and Attitude Guidance

Translational guidance consisted of a simple stationkeeping command. The chaser is commanded to hold a desired position in the LVLH frame. For this application, the LVLH frame is defined as follows. The origin is at the target spacecraft. The x axis is the local horizontal axis, with the positive direction aligned with the velocity bar (V-bar) of the target spacecraft. The y-axis is the cross track axis, with the positive direction aligned with the positive of the angular momentum. The z-axis is the local vertical axis, with the positive direction aligned with the radius bar (r-bar).

$$\begin{aligned} R_{com} &= R_{des} \\ V_{com} &= [0\ 0\ 0]^T \end{aligned} \tag{6.18}$$

The attitude guidance consisted of a simple target tracking command. The command issued by the algorithm would orient the chaser such that the LOS camera would point at the target. The commanded quaternion is derived from the commanded direction cosine

matrix by way of equation 6.25. The commanded direction cosine matrix is calculated by equation 6.19. The commanded angular rate is calculated by equation 6.20.

$$
\begin{aligned}
R_{com} &= \begin{bmatrix} \bar{e}_x^T & \bar{e}_y^T & \bar{e}_z^T \end{bmatrix} \\
\bar{e}_x &= \frac{(\bar{r}_t - \bar{r}_c)}{|\bar{r}_t - \bar{r}_c|} \\
\bar{e}_y &= \frac{(\bar{r}_c \times \bar{v}_c) \times e_x}{|(\bar{r}_c \times \bar{v}_c) \times e_x|} \\
\bar{e}_z &= \bar{e}_x \times \bar{e}_y
\end{aligned}
\tag{6.19}
$$

$$
\omega_{com} = \frac{(\bar{r}_t - \bar{r}_c) \times (\bar{v}_t - \bar{v}_c)}{|\bar{r}_t - \bar{r}_c|^2} \tag{6.20}
$$

6.7 Navigation

Navigation was accomplished by the Kalman filters developed in Chapters 4 and 7.

6.8 Position, Velocity, and Attitude Controllers

Primary controllers consist of a station keeping position-derivative (PD) controller, and two attitude controllers: a phase-plane controller for the thrusters, and a position-integral-derivative (PID) controller for the momentum wheels.

6.8.1 Translational Control: Station Keeping PD Controller

The station keeping PD computed a change in velocity command $\bar{dv}_{com}$.

$$
\bar{dv}_{com} = K_p(\bar{r}_{com} - \hat{r}_c) + K_D(\bar{v}_{com} - \hat{v}_c) \tag{6.21}
$$

The position gain $K_p = 0.010966227\, sec^{-1}$. The derivative gain $K_D = 0.18849556$. Because the thrusters are either on or off, $\bar{dv}_{com}$ was rounded to zero if less then half of

the thruster's minimum ΔV. The K_P and K_D gains were computed based on a natural frequency (ω_n) and damping ratio (ξ) of $2\pi/60$ and 0.9 respectively.

6.8.2 Attitude Controllers

Two controllers exist to control attitude. When the errors are large, the thrusters are used to correct attitude by way of a phase-plane controller. When the errors are small enough to be handled by the momentum wheels, the phase-plane controller is deactivated and a PID controller uses the momentum wheels for precision pointing.

6.8.2.1 Phase-Plane Controller for Thrusters

Phase Plane Controllers are exceptional at maintaining stable and predictable behavior when using bang-bang actuators like thrusters. A Phase Plane Controller has been implemented successfully on the Space Shuttle for many years. The phase plane controller implemented in the simulation is based on the work found in [23].

The geometry of the deadband and a typical control path is represented in figure 6.4. The initial state is to the bottom left and represents a large error in attitude and attitude rate. A PD control law drives the rate to the deadband region. The satellite then coasts until it approaches the desired position. The PD control law commands minimum impulse burns whenever the state leaves the deadband region, slowing the angular rate until it slightly overshoots the desired rate and reverses direction. A steady-state oscillation then occurs about the desired position/rate with a predictable frequency. If the rate is ever too fast, then a rate control law will slow the spacecraft down to the deadband rate.

The parameters required to develop the deadband geometry were:

- User Selected Parameters
  - $\delta\theta$ = permissible error in attitude = 6°
  - Convergence rate = $4°/sec$
  - ζ = damping ratio of the controller =0.707

- Satellite Specification Parameters
  - I_n =Satellite Inertia about respective axis
  - T = Available torque about respective axis
  - Δt = minimum thruster ontime
- Derived Parameters
  - $K = controller\ gain = \frac{2\Delta t}{\delta\theta}$
  - $\omega_n = natural\ frequency = \sqrt{\frac{KT}{I_n \Delta t}}$
  - $\tau = time\ constant\ of\ the\ system = \frac{2\zeta}{\omega_n}$
  - $\delta\dot{\theta}$ = permissible error in attitude rate =$\sqrt{\frac{2\Delta t}{K\tau}}$
  - Minimum time between thrusts during steady state operation = $\frac{2I_n}{KT}$

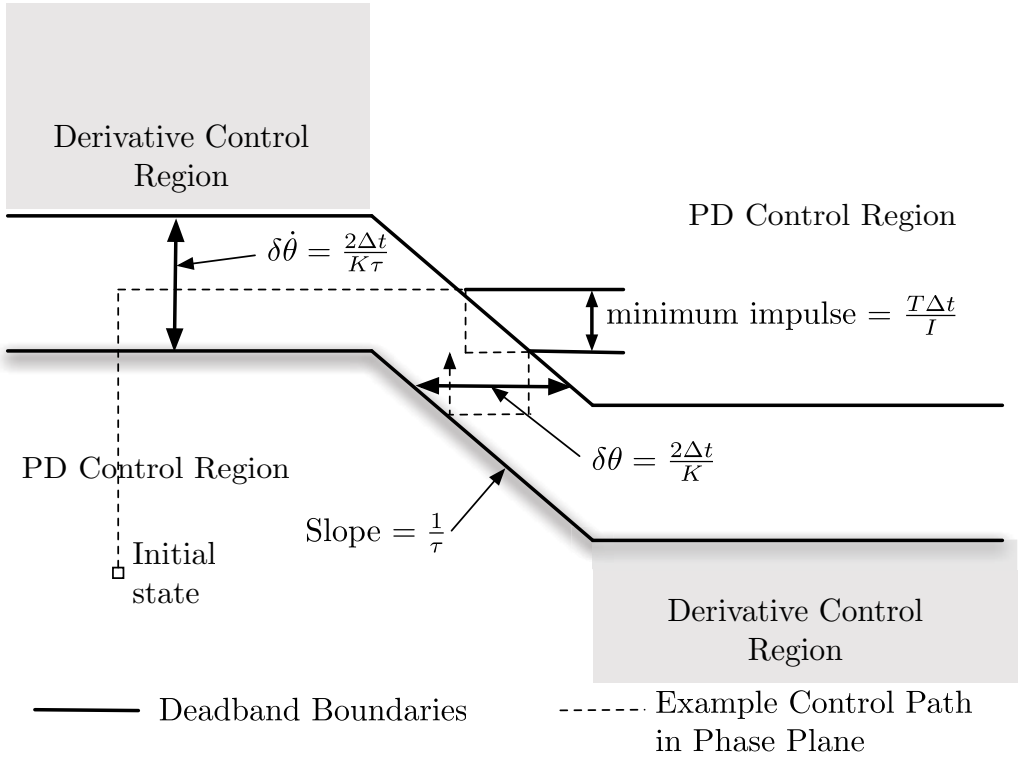


Fig. 6.4: Phase-plane attitude controller for thrusters.

6.8.2.2 PID Controller for Momentum Wheels

The PID controller for the momentum wheels is an extension of the quaternion error PD controller found in [22, Chapter 7 section 2]. Implementation involves just a few steps. Sidi develops a control law using the elements of the direction cosine error matrix defined by

$$\boldsymbol{A}_E = \boldsymbol{A}_{des}\boldsymbol{A}_c \tag{6.22}$$

This law is shown in equation 6.23, where a_{xxE} denotes an element of $\boldsymbol{A}_E$.

$$\begin{aligned} T_{cx} &= -\frac{1}{2}K_x(a_{32E} - a_{23E}) + K_{xd}\omega_x \\ T_{cy} &= -\frac{1}{2}K_y(a_{13E} - a_{31E}) + K_{yd}\omega_y \\ T_{cz} &= -\frac{1}{2}K_z(a_{21E} - a_{12E}) + K_{zd}\omega_z \end{aligned} \tag{6.23}$$

This control law has the advantage of always commanding a rotation about the Euler axis of rotation, minimizing the angular path to be covered. This control law can be implemented in terms of quaternions by calculating an error quaternion, which captures the difference between the desired and estimate quaternions (q_{des} *and* q_c).

$$q_E = \begin{bmatrix} q_{des4} & q_{des3} & -q_{des2} & q_{des1} \\ -q_{des3} & q_{des4} & q_{des1} & q_{des2} \\ q_{des2} & -q_{des1} & q_{des4} & q_{des3} \\ -q_{des1} & -q_{des2} & -q_{des3} & q_{des4} \end{bmatrix} \begin{bmatrix} -q_{c1} \\ -q_{c2} \\ -q_{c3} \\ q_{c4} \end{bmatrix} \tag{6.24}$$

Then, noting the relationship between direction cosine matrix elements and the elements of the quaternion vector noted in equation 6.25, equation 6.23 can be rewritten as equation 6.26.

$$\begin{aligned} q_4 &= \pm\, 0.5\sqrt{1 + a_{11} + a_{22} + a_{33}} \\ q_1 &= 0.25(a_{23} - a_{32})/q_4 \\ q_2 &= 0.25(a_{31} - a_{13})/q_4 \\ q_3 &= 0.25(a_{12} - a_{21})/q_4 \end{aligned} \tag{6.25}$$

$$\begin{aligned} T_{cx} &= 2K_x q_{1E} q_{4E} + K_{xd}\omega_x \\ T_{cy} &= 2K_y q_{2E} q_{4E} + K_{yd}\omega_y \\ T_{cz} &= 2K_z q_{3E} q_{4E} + K_{zd}\omega_z \end{aligned} \tag{6.26}$$

This result was modified into a PID controller by adding an integral and associated gain as seen in equation 6.27.

$$\begin{aligned} T_{cx} &= \int 2K_{xI}q_{1E}q_{4E}dt + 2K_x q_{1E}q_{4E} + K_{xd}\omega_x \\ T_{cy} &= \int 2K_{yI}q_{2E}q_{4E}dt + 2K_y q_{2E}q_{4E} + K_{yd}\omega_y \\ T_{cz} &= \int 2K_{zI}q_{3E}q_{4E}dt + 2K_z q_{3E}q_{4E} + K_{zd}\omega_z \end{aligned} \tag{6.27}$$

For implementation a deadbeat setup was used such that the deadbeat coefficients a_{db} = 1.90 and b_{db} = 2.20. The desired natural frequency was w_n = .707 rad/sec. This resulted in position, derivative, and integral gains of:

$$\begin{aligned} K_p &= \mathbb{I} b_{db} w_n^2 = [0.87973424\ 0.87973424\ 1.0996678]\, sec^{-1} \\ K_d &= a_{db} w_n \mathbb{I} = [1.07464\ 1.07464\ 1.3433] \\ K_I &= \mathbb{I} w_n^3 = [0.02827145944\ 0.02827145944\ 0.0353393243]\, sec^{-2} \end{aligned}$$

Chapter 7

Extended Kalman Filter for Angles-only Navigation

Developing an Extended Kalman Filter for orbital rendezvous angles-only navigation involves several steps, which are summarized in figure 7.1. First the filter design model must be developed. This is the navigation system designer's "working model." This nonlinear model accounts for the position and velocity of the chaser and target vehicles, misalignment and measurement noise on the accelerometers and the optical camera, and bias on the accelerometers. It uses process noise in the vehicle acceleration channel to account for unmodeled effects like drag and solar pressure.

Once the filter design model has been established, a different model without process noise effects, and possibly of lower fidelity, must be developed. This new model is still nonlinear and its integral with respect to time will be the state propagation equation inside the Kalman filter. Next, the state covariance propagation equation must be developed. The state covariance propagation equation requires a linearization of the state filter model as well as noise strength estimates based on the process noise values developed during the filter design model step.

Once the filter propagation equation has been established, the filter update step must be developed. The specifics of the update step will vary depending on whether it is a Conventional, Joseph, Carlson, or some other type of filter. However, each of these filters require a measurement sensitivity matrix, which is a linearized version of the measurement equation.

Once these five steps have been completed, these equations can be implemented into a conventional, Joseph, Potter, or some other Kalman filter formulation.

1. Develop filter design model.

$$\begin{aligned} \dot{\bar{x}} &= f(\bar{x}) + \boldsymbol{B}\bar{\mathrm{w}} \\ \bar{y} &= h(\bar{x}) + \eta_{cam} \end{aligned}$$

2. Develop state propagation equation.

$$\hat{x}_{i+1} = \int f(\hat{x}_i)\, dt$$

3. Linearize state filter model.

$$\boldsymbol{F} = \left.\frac{\partial f(\hat{x})}{\partial \hat{x}}\right|_{\hat{x}}$$

4. Develop the state covariance propagation equation.

$$\boldsymbol{P}_{i+1} = \boldsymbol{\Phi}_i \boldsymbol{P}_i \boldsymbol{\Phi}_i^T + \boldsymbol{B}\boldsymbol{Q}\boldsymbol{B}^T \delta t \qquad \text{where} \quad \boldsymbol{\Phi} = e^{\boldsymbol{F}dt}$$

5. Linearize measurement equation (a.k.a. measurement sensitivity matrix).

$$\boldsymbol{H} = \left.\frac{\partial h(\hat{x})}{\partial \hat{x}}\right|_{\hat{x}}$$

6. Implement into a conventional, Joseph, Potter, or other Kalman filter formulation.

Fig. 7.1: Extended Kalman filter development summary.

7.1 Filter Design Model

This is the navigation system designer's "working model." This model accounts for the position and velocity of the chaser and target vehicles in the Earth Centered Inertial (ECI) frame, misalignment and measurement noise on the accelerometers and the optical camera, and bias on the accelerometers. Note that the misalignment terms $\bar{\epsilon}_{acc}$ and $\bar{\epsilon}_{cam}$ and the bias term β are modeled as exponentially correlated random variables (ECRV's) with very long time constants. Also, the filter design model uses process noise in the vehicle acceleration channel to account for unmodeled effects like drag and solar pressure.

The only measurement that will be processed in the Kalman filter directly will be the LOS camera measurements. Accelerometers will be used to propagate position and attitude states directly. As a result, the accelerometer measurement noise (η_c) will be treated like a

process noise. The star-camera measurements will be used to transform between the chaser body frame and the ECI frame (i.e. $T_{b\rightarrow I}$).

The filter design model is represented by equations 7.1 and 7.2. Note that the $\bar{n}$ symbol is a vector defined as $[0\ 0\ 0]^T$.

$$\boxed{\begin{aligned}\dot{\bar{x}} &= f(\bar{x}) + \boldsymbol{B}\bar{\mathrm{w}} \\ \bar{y} &= h(\bar{x}) + \eta_{cam}\end{aligned}} \tag{7.1}$$

where

$$\dot{\bar{x}} = \begin{bmatrix} \dot{\bar{r}}_c \\ \dot{\bar{v}}_c \\ \dot{\bar{r}}_t \\ \dot{\bar{v}}_t \\ \dot{\bar{\beta}} \\ \dot{\bar{\epsilon}}_{acc} \\ \dot{\bar{\epsilon}}_{cam} \end{bmatrix}_{21x1} \qquad f(\bar{x}) = \begin{bmatrix} \bar{v}_c \\ g(\bar{r}_c) + \boldsymbol{T}_{b\rightarrow I}[\boldsymbol{I}_{3x3} + [\bar{\epsilon}_{acc}\times]][\widetilde{a} - \bar{\beta}] \\ \bar{v}_t \\ g(\bar{r}_t) \\ -\bar{\beta}/\tau_{acc\beta} \\ -\bar{\epsilon}_{acc}/\tau_{acc} \\ -\bar{\epsilon}_{cam}/\tau_{cam} \end{bmatrix}_{21x1}$$

$$g(\bar{r}) = -\mu\frac{\bar{r}}{|\bar{r}|^3} - \mu\frac{J_2 R_e^2}{2|\bar{r}|^5}\left\{6(\bar{r}\cdot\bar{n})\bar{n} + 3\bar{r} - 15(\bar{i}_r\cdot\bar{n})^2\bar{r}\right\} \tag{7.2}$$

$$\bar{\mathrm{w}} = \begin{bmatrix} \bar{w}_c \\ \bar{\eta}_c \\ \bar{w}_t \\ \bar{w}_{acc\beta} \\ \bar{w}_{acc} \\ \bar{w}_{cam} \end{bmatrix}_{18x1} \qquad \boldsymbol{B} = \begin{bmatrix} 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ \boldsymbol{I}_{3x3} & -\boldsymbol{T}_{b\rightarrow I}[\boldsymbol{I}_{3x3} + [\bar{\epsilon}_{acc}\times]] & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & \boldsymbol{I}_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & \boldsymbol{I}_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & \boldsymbol{I}_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & \boldsymbol{I}_{3x3} \end{bmatrix}_{21x18}$$

$$h(\bar{x}) = \begin{bmatrix} tan(az) \\ tan(el) \end{bmatrix} = \begin{bmatrix} R_z^{cam}/R_x^{cam} \\ R_y^{cam}/R_x^{cam} \end{bmatrix} \tag{7.3}$$

$$\bar{R}_{rel}^{cam} = [\boldsymbol{I}_{3x3} - [\bar{\epsilon}_{cam}\times]](\boldsymbol{T}_{I\rightarrow b}(\bar{r}_t - \bar{r}_c) - r_{cam}^b) = [R_x^{cam}\ R_y^{cam}\ R_z^{cam}]^T \tag{7.4}$$

Transformation Matrix

The Transformation Matrix comes directly from the star camera, which returns a quaternion measurement ($\widetilde{q}$), as seen below:

$$\boldsymbol{T}_{I\rightarrow b}(\bar{q}) = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}_{3x3} \tag{7.5}$$

where

$$\bar{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}_{4x1} = \begin{bmatrix} cos(\theta/2) \\ \hat{u}sin(\frac{\theta}{2}) \end{bmatrix} \tag{7.6}$$

$\hat{u}$ = unit vector defining axis of rotation

θ = angle of rotation in radians

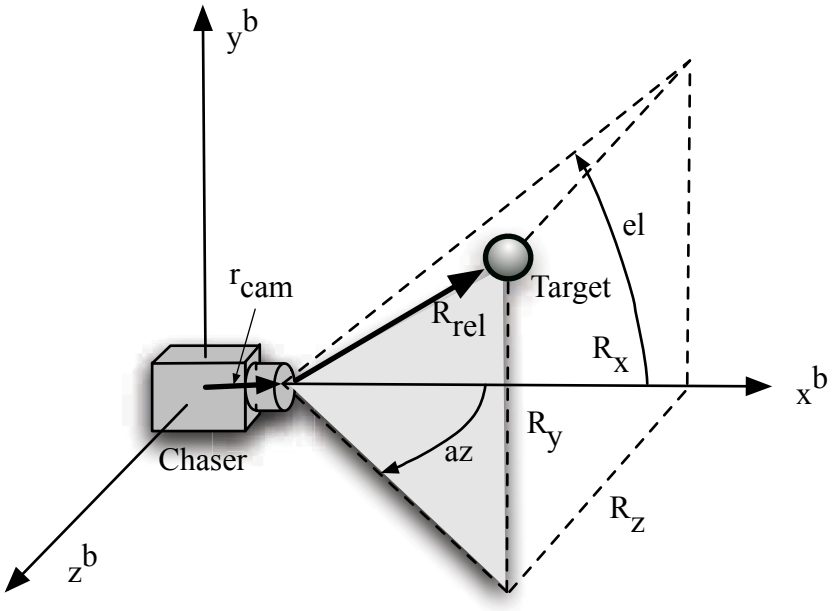


Fig. 7.2: Azimuth and elevation measurements in chaser body frame.

7.2 State Propagation Equation

The state propagation equation is simply the integral of $f(\hat{x})$ over time.

$$\hat{x}_{i+1} = \int f(\hat{x})\, dt \tag{7.7}$$

7.3 Linearized State Filter Model

The linearization of the dynamics equation is a precondition for solving for the transition matrix Φ.

The system dynamics equation ($f(\bar{x})$) is linearized as follows:

$$\boldsymbol{F} = \left.\frac{\partial f(\hat{x})}{\partial \hat{x}}\right|_{\hat{x}} = \left[\begin{array}{ccccccc} \frac{\partial f(\hat{x})}{\partial \hat{r}_c} & \frac{\partial f(\hat{x})}{\partial \hat{v}_c} & \frac{\partial f(\hat{x})}{\partial \hat{r}_t} & \frac{\partial f(\hat{x})}{\partial \hat{v}_t} & \frac{\partial f(\hat{x})}{\partial \hat{\beta}} & \frac{\partial f(\hat{x})}{\partial \hat{\epsilon}_{acc}} & \frac{\partial f(\hat{x})}{\partial \hat{\epsilon}_{cam}} \end{array}\right]_{21x21} \tag{7.8}$$

Each element of the above matrix is a column of partials as seen below.

$$\boldsymbol{F} = \left[\begin{array}{c|c|c|c|c|c|c} \partial\dot{\hat{r}}_c/\partial\hat{r}_c & \partial\dot{\hat{r}}_c/\partial\hat{v}_c & \partial\dot{\hat{r}}_c/\partial\hat{r}_t & \partial\dot{\hat{r}}_c/\partial\hat{v}_t & \partial\dot{\hat{r}}_c/\partial\hat{\beta} & \partial\dot{\hat{r}}_c/\partial\hat{\epsilon}_{acc} & \partial\dot{\hat{r}}_c/\partial\hat{\epsilon}_{cam} \\ \partial\dot{\hat{v}}_c/\partial\hat{r}_c & \partial\dot{\hat{v}}_c/\partial\hat{v}_c & \partial\dot{\hat{v}}_c/\partial\hat{r}_t & \partial\dot{\hat{v}}_c/\partial\hat{v}_t & \partial\dot{\hat{v}}_c/\partial\hat{\beta} & \partial\dot{\hat{v}}_c/\partial\hat{\epsilon}_{acc} & \partial\dot{\hat{v}}_c/\partial\hat{\epsilon}_{cam} \\ \partial\dot{\hat{r}}_t/\partial\hat{r}_c & \partial\dot{\hat{r}}_t/\partial\hat{v}_c & \partial\dot{\hat{r}}_t/\partial\hat{r}_t & \partial\dot{\hat{r}}_t/\partial\hat{v}_t & \partial\dot{\hat{r}}_t/\partial\hat{\beta} & \partial\dot{\hat{r}}_t/\partial\hat{\epsilon}_{acc} & \partial\dot{\hat{r}}_t/\partial\hat{\epsilon}_{cam} \\ \partial\dot{\hat{v}}_t/\partial\hat{r}_c & \partial\dot{\hat{v}}_t/\partial\hat{v}_c & \partial\dot{\hat{v}}_t/\partial\hat{r}_t & \partial\dot{\hat{v}}_t/\partial\hat{v}_t & \partial\dot{\hat{v}}_t/\partial\hat{\beta} & \partial\dot{\hat{v}}_t/\partial\hat{\epsilon}_{acc} & \partial\dot{\hat{v}}_t/\partial\hat{\epsilon}_{cam} \\ \partial\dot{\hat{\beta}}/\partial\hat{r}_c & \partial\dot{\hat{\beta}}/\partial\hat{v}_c & \partial\dot{\hat{\beta}}/\partial\hat{r}_t & \partial\dot{\hat{\beta}}/\partial\hat{v}_t & \partial\dot{\hat{\beta}}/\partial\hat{\beta} & \partial\dot{\hat{\beta}}/\partial\hat{\epsilon}_{acc} & \partial\dot{\hat{\beta}}/\partial\hat{\epsilon}_{cam} \\ \partial\dot{\hat{\epsilon}}_{acc}/\partial\hat{r}_c & \partial\dot{\hat{\epsilon}}_{acc}/\partial\hat{v}_c & \partial\dot{\hat{\epsilon}}_{acc}/\partial\hat{r}_t & \partial\dot{\hat{\epsilon}}_{acc}/\partial\hat{v}_t & \partial\dot{\hat{\epsilon}}_{acc}/\partial\hat{\beta} & \partial\dot{\hat{\epsilon}}_{acc}/\partial\hat{\epsilon}_{acc} & \partial\dot{\hat{\epsilon}}_{acc}/\partial\hat{\epsilon}_{cam} \\ \partial\dot{\hat{\epsilon}}_{cam}/\partial\hat{r}_c & \partial\dot{\hat{\epsilon}}_{cam}/\partial\hat{v}_c & \partial\dot{\hat{\epsilon}}_{cam}/\partial\hat{r}_t & \partial\dot{\hat{\epsilon}}_{cam}/\partial\hat{v}_t & \partial\dot{\hat{\epsilon}}_{cam}/\partial\hat{\beta} & \partial\dot{\hat{\epsilon}}_{cam}/\partial\hat{\epsilon}_{acc} & \partial\dot{\hat{\epsilon}}_{cam}/\partial\hat{\epsilon}_{cam} \end{array}\right]_{21x21} \tag{7.9}$$

Many of these partial derivatives are zero, resulting in the following.

$$\boldsymbol{F} = \left[\begin{array}{c|c|c|c|c|c|c} 0_{3x3} & \partial\dot{\hat{r}}_c/\partial\hat{v}_c & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ \partial\dot{\hat{v}}_c/\partial\hat{r}_c & 0_{3x3} & 0_{3x3} & 0_{3x3} & \partial\dot{\hat{v}}_c/\partial\hat{\beta} & \partial\dot{\hat{v}}_c/\partial\hat{\epsilon}_{acc} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & \partial\dot{\hat{r}}_t/\partial\hat{v}_t & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & \partial\dot{\hat{v}}_t/\partial\hat{r}_t & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & \partial\dot{\hat{\beta}}/\partial\hat{\beta} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & \partial\dot{\hat{\epsilon}}_{acc}/\partial\hat{\epsilon}_{acc} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & \partial\dot{\hat{\epsilon}}_{cam}/\partial\hat{\epsilon}_{cam} \end{array}\right]_{21x21} \tag{7.10}$$

The individual partial derivatives are evaluated as follows:

$$\begin{aligned} \partial\dot{\hat{r}}_c/\partial\hat{v}_c &= \boldsymbol{I}_{3x3} \\ \partial\dot{\hat{v}}_c/\partial\hat{r}_c &= A_1(\hat{r}_c) + A_2(\hat{r}_c) + A_3(\hat{r}_c) \\ \partial\dot{\hat{v}}_c/\partial\hat{\beta} &= -\boldsymbol{T}_{b\rightarrow I}[\boldsymbol{I}_{3x3} + [\hat{\epsilon}_{acc}\times]] \\ \partial\dot{\hat{v}}_c/\partial\hat{\epsilon}_{acc} &= \boldsymbol{T}_{b\rightarrow I}[[\hat{\beta}\times] - [\widetilde{a}\times]] \\ \partial\dot{\hat{r}}_t/\partial\hat{v}_t &= \boldsymbol{I}_{3x3} \\ \partial\dot{\hat{v}}_t/\partial\hat{r}_t &= A_1(\hat{r}_t) + A_2(\hat{r}_t) + A_3(\hat{r}_t) \\ \partial\dot{\hat{\beta}}/\partial\hat{\beta} &= \frac{-1}{\tau_{acc\beta}}\boldsymbol{I}_{3x3} \\ \partial\dot{\hat{\epsilon}}_{acc}/\partial\hat{\epsilon}_{acc} &= \frac{-1}{\tau_{acc}}\boldsymbol{I}_{3x3} \\ \partial\dot{\hat{\epsilon}}_{cam}/\partial\hat{\epsilon}_{cam} &= \frac{-1}{\tau_{cam}}\boldsymbol{I}_{3x3} \end{aligned} \tag{7.11}$$

where

$$A_1(\hat{r}) = \frac{-\mu J_2 R_e^2}{|\hat{r}|^3}\left(\boldsymbol{I}_{3x3} - 3\hat{i}_r\hat{i}_r^T\right)$$

$$A_2(\hat{r}) = \frac{-3\mu J_2 R_e^2 \bar{n}}{|\hat{r}|^5}\left(\bar{n}^T\boldsymbol{I}_{3x3} - 5(\hat{i}_r^T\bar{n})\hat{i}_r^T\right) - \frac{3\mu J_2 R_e^2}{2|\hat{r}|^5}\left(\boldsymbol{I}_{3x3} - 5\hat{i}_r\hat{i}_r^T\right)$$

$$A_3(\hat{r}) = \frac{7.5\mu J_2 R_e^2}{|\hat{r}|^5}\left[(\hat{i}_r^T\bar{n})^2\boldsymbol{I}_{3x3} - 5(\hat{i}_r^T\bar{n})^2\hat{i}_r\hat{i}_r^T + \frac{\hat{r}\hat{i}_r^T\bar{n}\bar{n}^T\boldsymbol{I}_{3x3}}{|\hat{r}|}\left(\boldsymbol{I}_{3x3} - \hat{i}_r\hat{i}_r^T\right)\right]$$

7.4 State Covariance Propagation Equation

The covariance matrix is propagated discretely with the state transition matrix.

$$\boldsymbol{P}_{i+1} = \boldsymbol{\Phi}_i\boldsymbol{P}_i\boldsymbol{\Phi}_i^{\boldsymbol{T}} + \boldsymbol{B}\boldsymbol{Q}\boldsymbol{B}^{\boldsymbol{T}}\delta t \tag{7.12}$$

Transition Matrix

$$\boldsymbol{\Phi} = e^{\boldsymbol{F}dt} = \boldsymbol{I} + \boldsymbol{F}dt + \frac{\boldsymbol{F}^2dt^2}{2!} + \frac{\boldsymbol{F}^3dt^3}{3!} + \cdots \tag{7.13}$$

Strength of the Process Noise

The strength of the process noise $\boldsymbol{Q}$ is related to the process noise $\bar{w}$ by the relationship:

$$\boldsymbol{Q}\delta(t'-t) = E(\hat{\mathrm{w}}'\hat{\mathrm{w}}^{\mathrm{T}}) = E[\hat{w}(t')\hat{w}(t)] \tag{7.14}$$

where the E operator is the expected value, and $\delta(t'-t)$ is the Dirac delta function.

$\boldsymbol{Q}$ may be represented as a matrix as shown below.

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_{w_c} & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & \boldsymbol{Q}_{\eta_c} & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & \boldsymbol{Q}_{w_t} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & \boldsymbol{Q}_{w_{acc\beta}} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & \boldsymbol{Q}_{w_{acc}} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & \boldsymbol{Q}_{w_{cam}} \end{bmatrix}_{18x18} \tag{7.15}$$

The values in these submatrices usually come from the hardware specification, while $\boldsymbol{Q}_{\omega_c}$ and $\boldsymbol{Q}_{\omega_t}$ are functions of unmodeled accelerations of the chaser and target, these would

include J_{3+} gravity effects and solar radiation forces.

7.5 Linearized Measurement Equation

The linearization of the measurement equation results in the measurement sensitivity matrix ($\boldsymbol{H}$), which is required to solve for the Kalman gain ($\boldsymbol{K}$).

Note that the $\boldsymbol{H}$ matrix will take the general form:

$$\begin{aligned}\boldsymbol{H} &= \left.\frac{\partial h(\hat{x})}{\partial \hat{x}}\right|_{\hat{x}} = \left.\frac{\partial h(\hat{x})}{\partial \hat{R}^b_{rel}}\frac{\partial \hat{R}^b_{rel}}{\partial \hat{x}}\right|_{\hat{x}} \\ &= \frac{\partial h(\hat{x})}{\partial \hat{R}^b_{rel}}\left[\begin{array}{ccccccc} \frac{\partial \hat{R}^b_{rel}}{\partial \hat{r}_c} & \frac{\partial \hat{R}^b_{rel}}{\partial \hat{v}_c} & \frac{\partial \hat{R}^b_{rel}}{\partial \hat{r}_t} & \frac{\partial \hat{R}^b_{rel}}{\partial \hat{v}_t} & \frac{\partial \hat{R}^b_{rel}}{\partial \hat{\beta}} & \frac{\partial \hat{R}^b_{rel}}{\partial \hat{\epsilon}_{acc}} & \frac{\partial \hat{R}^b_{rel}}{\partial \hat{\epsilon}_{cam}} \end{array}\right]_{2x21}\end{aligned} \tag{7.16}$$

The angle measurement is not a function of acceleration bias (β), accelerometer misalignment ($\hat{\epsilon}_{acc}$), or vehicle velocities. So the measurement sensitivity matrix becomes:

$$\boldsymbol{H} = \frac{\partial h(\hat{x})}{\partial \hat{R}^b_{rel}}\left[\begin{array}{ccccccc} \frac{\partial \hat{R}^b_{rel}}{\partial \hat{r}_c} & 0 & \frac{\partial \hat{R}^b_{rel}}{\partial \hat{r}_t} & 0 & 0 & 0 & \frac{\partial \hat{R}^b_{rel}}{\partial \hat{\epsilon}_{cam}} \end{array}\right]_{2x21} \tag{7.17}$$

The individual partials are:

$$\begin{aligned}\frac{\partial h(\hat{x})}{\partial \hat{R}^b_{rel}} &= \left[\begin{array}{ccc} \frac{-R_z}{R_x^2} & 0 & \frac{1}{R_x} \\ \\ \frac{-R_y}{R_x^2} & \frac{1}{R_x} & 0 \end{array}\right]_{2x3} \\ \frac{\partial \hat{R}^b_{rel}}{\partial \hat{r}_c} &= -[I_{3x3} - [\hat{\epsilon}_{cam}\times]]T_{I\to b} \\ \frac{\partial \hat{R}^b_{rel}}{\partial \hat{r}_t} &= [I_{3x3} - [\hat{\epsilon}_{cam}\times]]T_{I\to b} \\ \frac{\partial \hat{R}^b_{rel}}{\partial \hat{\epsilon}_{cam}} &= [\Big(T_{I\to b}(\hat{r}_t - \hat{r}_c) - r^b_{cam}\Big)\times]\end{aligned} \tag{7.18}$$

7.6 Implementation of the Extended Kalman Filter

Actual implementation of the extended Kalman filter combines the filter algorithms discussed in Chapter 4 (figures 4.2 and 4.3) with the equations developed so far in this chapter. Figure 7.3 shows how the equations covered in this chapter may be implemented

in a conventional Kalman filter. The labeled blocks are implementations of the following equations:

A Equations 7.17 and 7.3

B Equation 7.5

C Equation 7.7 with $f(\hat{x})$ defined in equation 7.2 (Also see propagate step in figure 4.2a)

D Equations 7.13 and 7.10

E Exponentially decaying value for measurement covariance ($\boldsymbol{R}$) (see section 8.3)

F State and state covariance update (see figure 4.2a)

G See definition of $\boldsymbol{B}$ in equation 7.2

H Equation 7.12 (Also see the propagate step in figure 4.2a)

Of course, there are many constants and parameters that must be defined before this code will run. However, these values depend on the scenario to be run. Thus they are defined in sections 8.2 and 8.3. Actual code can be found in Appendix E.

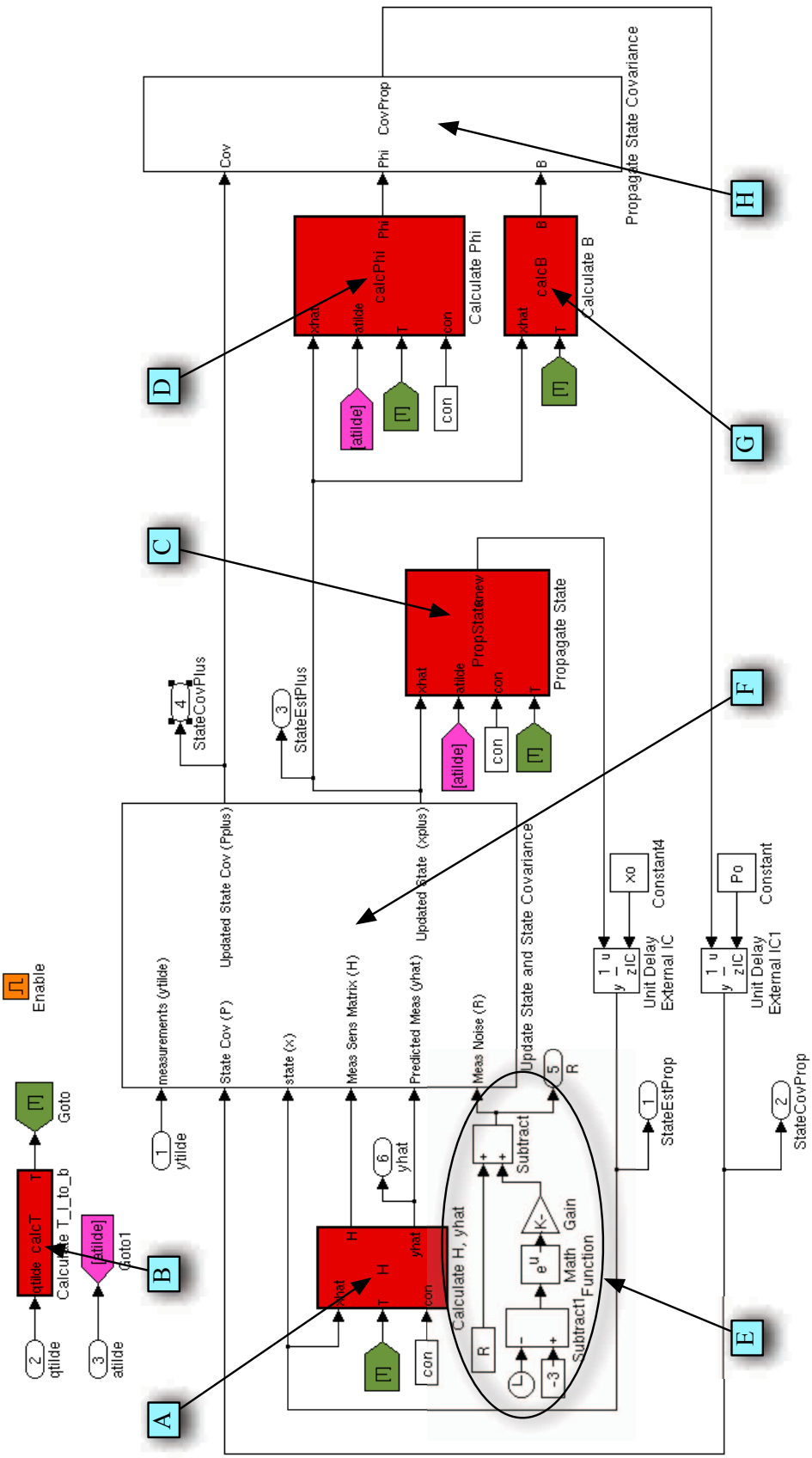


Fig. 7.3: Implementation of an extended Kalman filter.

Chapter 8

Open-Loop Relative Navigation Performance Analysis

This chapter examines the sensitivity of navigation performance and range observability to sensor accuracy, thrust acceleration levels, initial state uncertainties, and the type of navigation filter that is employed. To ensure an accurate comparison of the navigation performance data, all maneuvers are executed in an open-loop manner. Chapter 9 examines the performance of the navigation filter in a closed-loop environment.

The effects of accelerometer and LOS camera errors on filter performance is examined in sections 8.4 through 8.6. Section 8.7 covers a couple of approaches for improving filter performance without changing sensor performance. Section 8.8 quantitatively demonstrates that square-root filters are less susceptible to numerical errors brought on by large initial uncertainties or extremely accurate measurements.

8.1 Performance Metrics

The performance metrics for this study are the true relative navigation position errors and the associated relative navigation position error covariance. Particular attention will be given to the relative range error and variance in order to better understand when and under what conditions the relative range is observable. In general, a smaller error is better, but only if the filter covariance reflects the true estimation error statistics. In the absence of Monte-Carlo analysis, performance evaluation is often a pass/fail approach. If the true errors are reliably within the 3σ bounds, the filter is performing well.

8.2 Nominal Trajectory

The nominal flightpath consists of a chaser satellite that is following a non-maneuvering

target satellite. The chaser is located slightly off the v-bar and 406 m behind with zero relative velocity.

In order to make the range observable, the Chaser satellite fires thrusters every 75 seconds, alternatively in the positive and negative cross-track directions for 15 and 16.5 seconds, respectively. The resulting thrust acceleration is shown in figure 8.1. The resulting flightpath is shown in figures 8.2 and 8.3. This thruster firing pattern continues until the end of the simulation.

The LOS camera measurements are shown in figure 8.4. Initially, the chaser is pointing three degrees above and two and a half degrees to the right of the target, but the attitude controller quickly brings that pointing error down to almost zero.

The initial position and velocity of the target and chaser are shown in table 8.1.

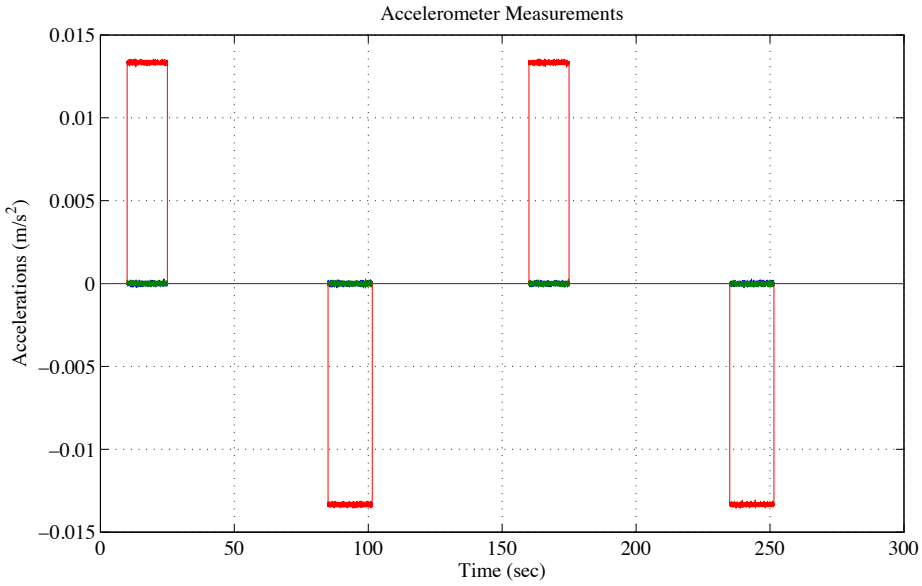


Fig. 8.1: Nominal accelerations.

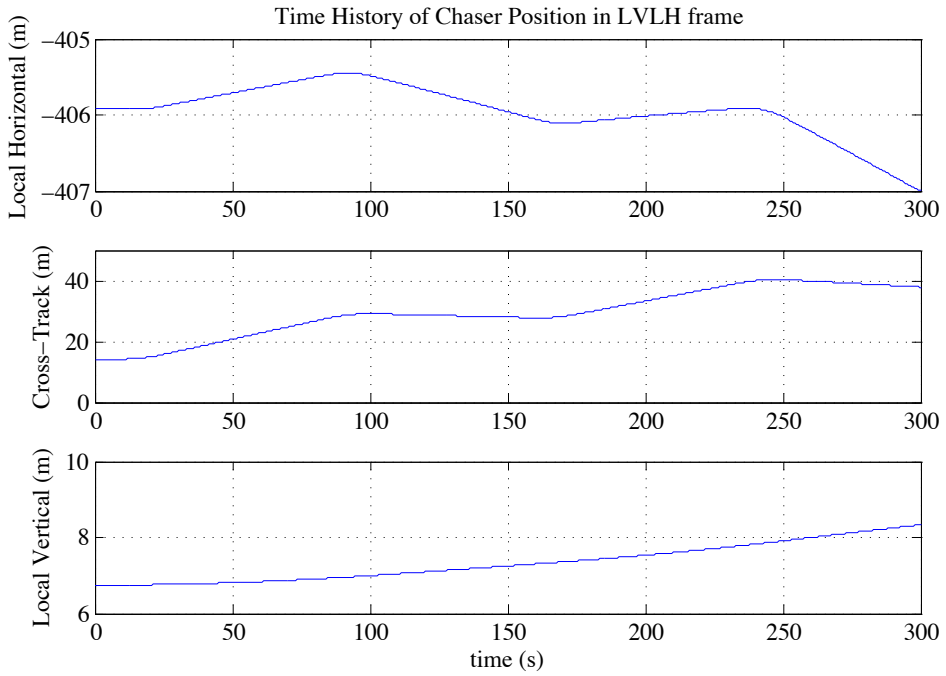


Fig. 8.2: Chaser flightpath broken down by LVLH coordinate frame components.

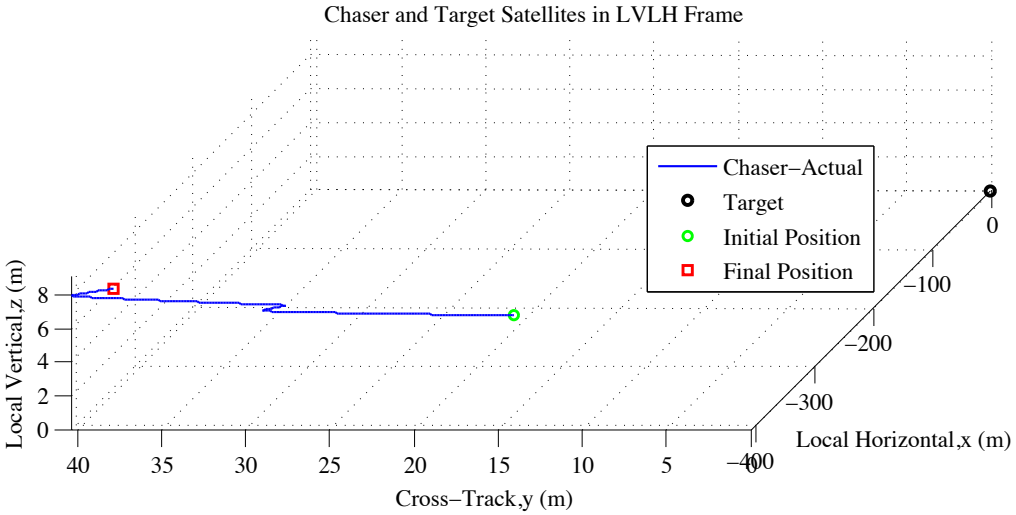


Fig. 8.3: Chaser flightpath and target in LVLH coordinate frame.

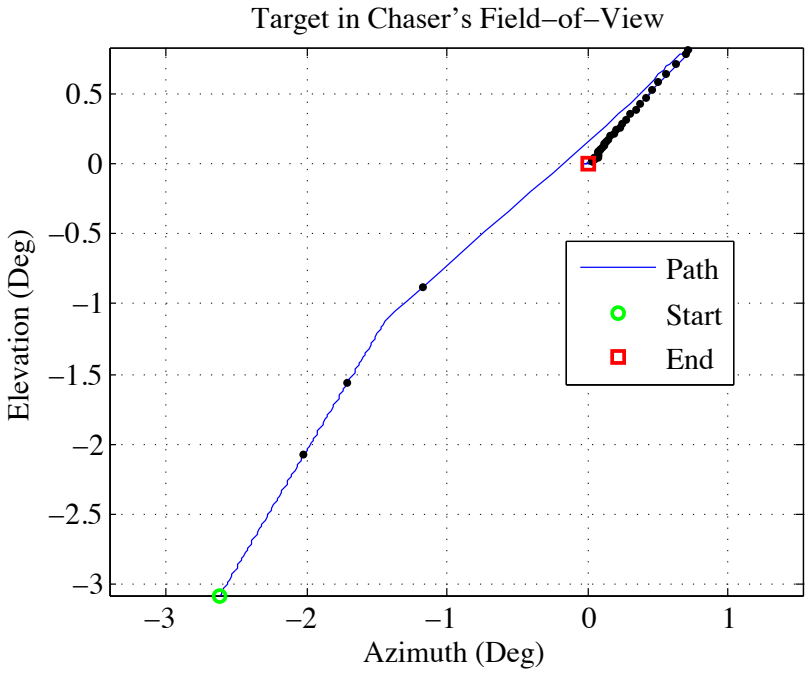


Fig. 8.4: Target spacecraft in chaser spacecraft's LOS camera field-of-view.

Table 8.1: Position and Velocity Initial Conditions in ECI Frame

State	Description	True Initial Condition
$\bar{r}_c$	Chaser Position	$\begin{matrix} x \\ y \\ z \end{matrix} = \begin{bmatrix} 5114.081942 \\ -3998.006812 \\ -334.998180 \end{bmatrix}$ km
$\bar{v}_c$	Chaser Velocity	$\begin{matrix} x \\ y \\ z \end{matrix} = \begin{bmatrix} 4.819968 \\ 6.171422 \\ -0.071192 \end{bmatrix}$ km/s
$\bar{r}_t$	Target Position	$\begin{matrix} x \\ y \\ z \end{matrix} = \begin{bmatrix} 5114.325821 \\ -3997.682427 \\ -335.016751 \end{bmatrix}$ km
$\bar{v}_t$	Target Velocity	$\begin{matrix} x \\ y \\ z \end{matrix} = \begin{bmatrix} 4.819554 \\ 6.171718 \\ -.071167 \end{bmatrix}$ km/s

The accelerometer and LOS camera specifications are categorized as good, average and poor and are given in the tables below. The star-tracker was assumed to give perfect measurements.

Table 8.2: Accelerometer Specifications

Accelerometers			
	Noise Strength 1σ per axis	Bias 1σ per axis	Misalignment 1σ per axis
good	$3 \times 10^{-9} m^2/s^3$	$3 \times 10^{-4} m/s^2$	$1 \times 10^{-5}\, rad$
average	$3 \times 10^{-7} m^2/s^3$	$3 \times 10^{-3} m/s^2$	$1 \times 10^{-4}\, rad$
poor	$3 \times 10^{-5} m^2/s^3$	$3 \times 10^{-2} m/s^2$	$1 \times 10^{-3}\, rad$
The nominal accelerometer performance is good			
Accelerometer quantization is held fixed at $1 \times 10^{-9}\, m/s^2$			

Table 8.3: Line-of-sight Camera Specifications

Line-of-Sight Camera		
	Measurement Noise 1σ per axis	Misalignment 1σ per axis
good	$1 \times 10^{-5}\, rad$	$1 \times 10^{-5}\, rad$
average	$1 \times 10^{-4}\, rad$	$1 \times 10^{-4}\, rad$
poor	$1 \times 10^{-3}\, rad$	$1 \times 10^{-3}\, rad$
The nominal LOS camera performance is good with misalignment 1σ set to $1 \times 10^{-6}\, rad$		

8.3 Filter Setup

The filter initial state and state covariance values are summarized in tables 8.4 and 8.5. Because this study did not perform a Monte Carlo analysis, the initial positions and velocities have a fixed 1σ error on each axis. This 1σ error is added to the chaser initial state, and subtracted from the target state. Thus, the inertial states are in error by $1\sigma/axis$ and the relative state is off by 2σ in each axis. By default, the position error 1σ is 10m/axis and velocity 1σ is 1cm/s/axis.

Filter bias and misalignment states are treated a little differently. These states are initialized to zero, but the true misalignment or bias is a random value with a standard deviation corresponding to the sensor specifications for that run. The initial filter covariances is tuned accordingly.

The measurement covariance matrix ($\boldsymbol{R}$) is a diagonal matrix whose elements are set to the LOS camera angle variance specifications defined in table 8.3. If the initial position covariance is large, and an accurate LOS camera measurement is taken, the filter will overwhelmingly favor the LOS camera measurement. This is not desirable, because too few LOS measurements have been taken to be statistically representative of the camera's accuracy. An initial 3σ measurement error can drastically degrade filter performance. To overcome this problem, the measurement covariance matrix is initially oversized, and then exponentially decays to the correct value. This can be tuned from case to case, but for this simulation $\boldsymbol{R}$ is varied according to $\boldsymbol{R}_{actual} = \boldsymbol{R}_{nominal} + \boldsymbol{I}e^{-3t}$, where t is the simulation time.

The strength of the process noise in the filter is dependent on the dynamic model inadequacies and sensor parameters. The values are summarized in table 8.6. $\boldsymbol{Q}_{\eta_c}$ is the strength of the accelerometer noise. The process noise strengths for the accelerometer bias ($\boldsymbol{Q}_{\omega_{acc\beta}}$), accelerometer misalignment ($\boldsymbol{Q}_{\omega_{acc}}$), and camera misalignment ($\boldsymbol{Q}_{\omega_{cam}}$) are functions of the their associated ECRV time constants and standard deviation. They are calculated by

$$\boldsymbol{Q}_{\omega_{acc\beta}} = \frac{2\sigma^2_{acc\beta}}{\tau_{acc\beta}}$$

$$\boldsymbol{Q}_{\omega_{acc}} = \frac{2\sigma^2_{acc}}{\tau_{acc}}$$

$$\boldsymbol{Q}_{\omega_{cam}} = \frac{2\sigma^2_{cam}}{\tau_{cam}}$$

The variances (σ^2) comes from tables 8.2 and 8.3, and the time constants from table 8.7.

Other filter parameters, including gravitational constants, time constants, simulation stepsize, and the LOS camera position in the chaser body frame are listed in table 8.7. For the open-loop analysis with a stepsize of .01 seconds, an Euler integrator was found to be sufficiently accurate for analysis.

Table 8.4: Filter State Initial Conditions in ECI

State	Description	Filter initial condition	Units
$\hat{r}_c$	Chaser Position	Truth+1 σ error/axis	km
$\hat{v}_c$	Chaser Velocity	Truth+1 σ error/axis	km/s
$\hat{r}_t$	Target Position	Truth-1 σ error/axis	km
$\hat{v}_t$	Target Velocity	Truth-1 σ error/axis	km/s
$\hat{\beta}$	Bias of accelerometers	[0 0 0]	km/s^2
$\hat{\epsilon}_{acc}$	Misalignment of accelerometers	[0 0 0]	rad
$\hat{\epsilon}_{cam}$	Misalignment of camera	[0 0 0]	rad

Table 8.5: Filter State Covariance Initial Conditions

Component of Covariance	Description	Value (1σ)
$\boldsymbol{P}_{r_c r_c}$	Chaser Position	$10\,m/axis$
$\boldsymbol{P}_{v_c v_c}$	Chaser Velocity	$1\,cm/s/axis$
$\boldsymbol{P}_{r_t r_t}$	Target Position	$10\,m/axis$
$\boldsymbol{P}_{v_t v_t}$	Target Velocity	$1\,cm/s/axis$
$\boldsymbol{P}_{\beta\beta}$	Accelerometer Bias	see table 8.2
$\boldsymbol{P}_{\epsilon_{acc}\epsilon_{acc}}$	Accelerometer Misalignment	see table 8.2
$\boldsymbol{P}_{\epsilon_{cam}\epsilon_{cam}}$	LOS Camera Misalignment	see table 8.3

Table 8.6: Process Noise

Parameter	Description	Value in Filter (1σ)	Units
$\boldsymbol{Q}_{\omega_c}$	Strength of process noise due to random accelerations on Chaser	1.0×10^{-18}	km^2/s^3
$\boldsymbol{Q}_{\eta_c}$	Strength of Process Noise due to accelerometer measurement noise being processed directly in propagator	Reference acclerometer noise strength in table 8.2	km^2/s^3
$\boldsymbol{Q}_{\omega_t}$	Strength of process noise due to random accelerations on target	10^{-18}	km^2/s^3
$\boldsymbol{Q}_{\omega_{acc\beta}}$	Strength of process noise on accelerometer bias	$\frac{2\sigma^2_{acc\beta}}{\tau_{acc\beta}}$ 1.8×10^{-19} for good case	km^2/s^3
$\boldsymbol{Q}_{\omega_{acc}}$	Strength of process noise on accelerometer misalignment	$\frac{2\sigma^2_{acc}}{\tau_{acc}}$ 2×10^{-16} for good case	rad^2/s
$\boldsymbol{Q}_{\omega_{cam}}$	Strength of process noise on camera misalignment	$\frac{2\sigma^2_{cam}}{\tau_{cam}}$ 2×10^{-16} for good case	rad^2/s

Table 8.7: Navigation Filter Constants

Symbol	**Definition**	**Value**
μ	Gravitation Constant	398600.4415 km^3/s^2
J_2	Second order gravitation parameter	0.0010826269
R_e	Radius of the Earth	6378.1367 km
$\tau_{acc\beta}$	ECRV time constant for accelerometer bias	10^6 sec
τ_{acc}	ECRV time constant for accelerometer misalignment	10^6 sec
τ_{cam}	ECRV time constant for camera misalignment	10^6 sec
dt	Propagation step size	0.01 sec
r_{cam}	Position of the LOS camera in the Chaser Satellite Body Frame	[0 0 0] m

8.4 Effect of Accelerometer Errors

The effect of accelerometer accuracy is quantified by performing runs with the nominal specifications on the LOS camera, and good, average and poor specifications for the accelerometers (see tables 8.2 and 8.8).

As the sensor accuracy becomes worse, it is more and more difficult to identify thruster burns (figure 8.5). For the poor accelerometers, the acceleration measurements are almost completely swamped by the noise.

As seen in figure 8.6, all three components of the relative position estimate are affected by th accelerometer errors, but the dominant effect is on the range estimate. Thus accelerometer accuracy requirements are strongly determined by range error requirements.

For the poor case, the true errors grew to be greater than the 3σ value (figure 8.7). This is undesireable because it indicates that the filter is not modeling the covariance correctly. Increasing the initial relative range solves this problem (figure 8.8) demonstrating that non-linear effects were the culprit. In other words, if the range covariance is a significant percentage of the total range, then the linear assumptions used to propagate the covariance are no longer valid.

Overall, the good and average accelerometers demonstrate good relative navigation performance, with good range observability. The poor accelerometers are not sufficiently accurate to observe range reliably.

Table 8.8: Cases Comparing Effects of Accelerometer Errors

Study	Thruster Acceleration	Accels	LOS Cam	Chaser Location
Explore effects of accelerometer accuracy	13.33 mm/s^2	Good	Nominal	V-bar, 406 m behind
		Average		
		Poor		
Verify nonlinearity effects				V-bar, 1620 m behind

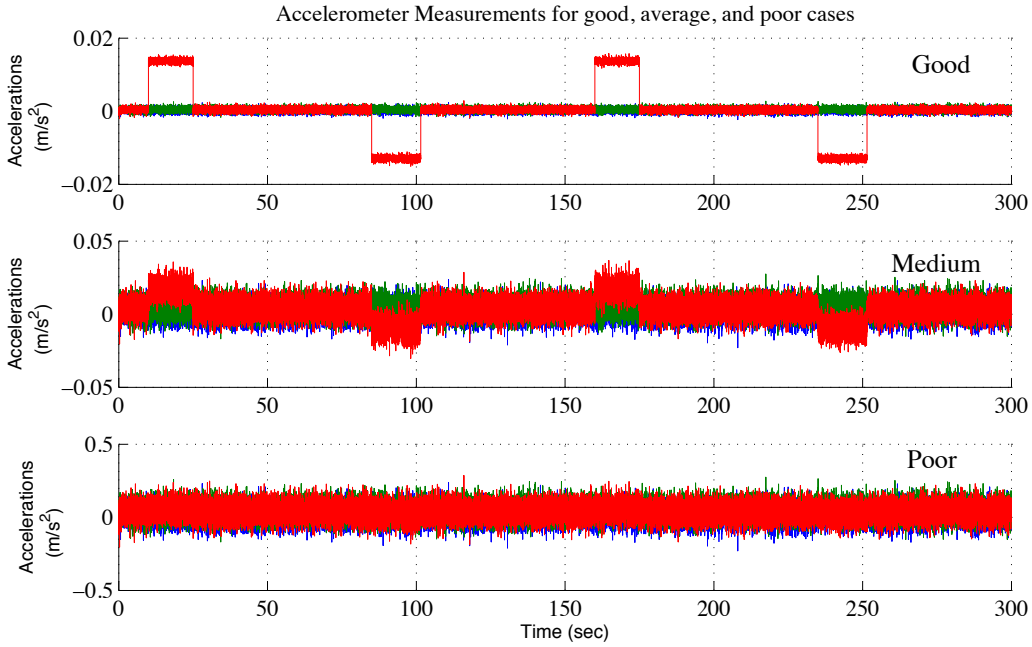


Fig. 8.5: Accelerometer measurements for good, average, and poor cases.

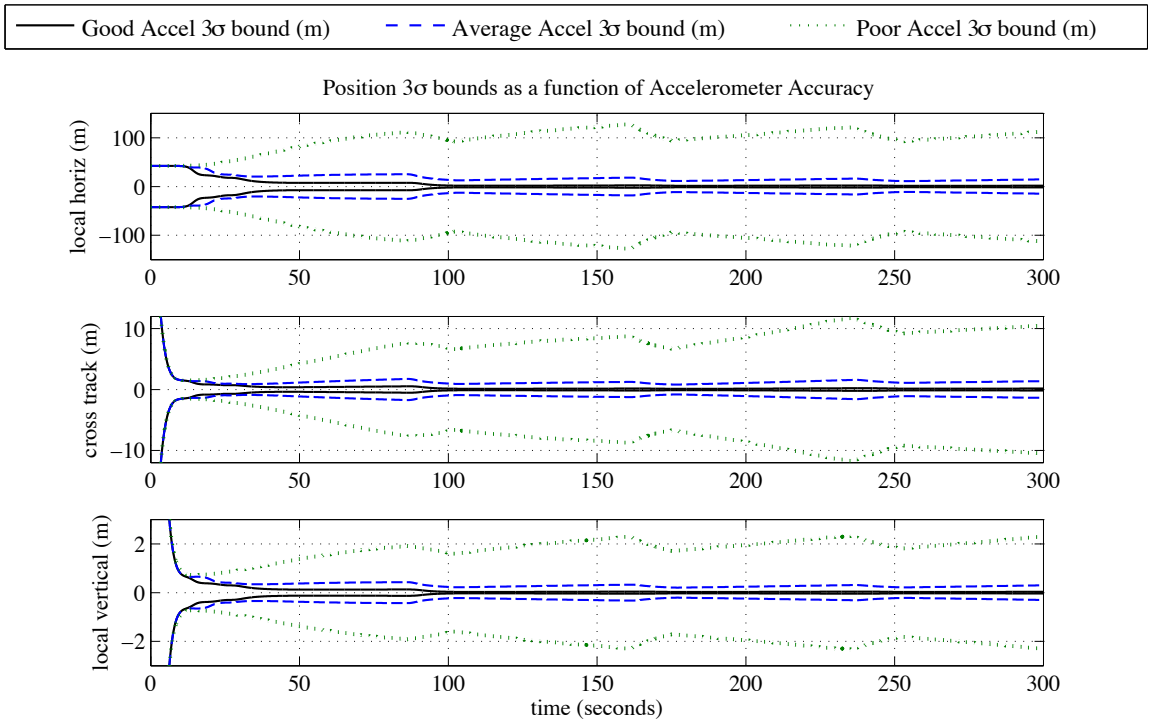


Fig. 8.6: Effects of accelerometer errors on relative position navigation errrors (good-solid, average-dashed, poor-dotted).

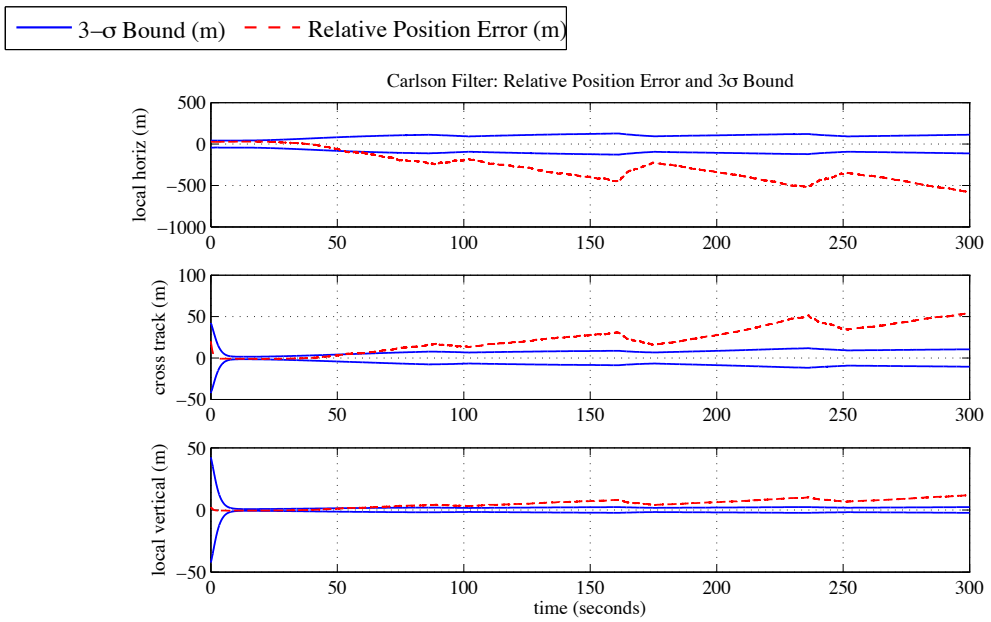


Fig. 8.7: With poor accelerometers at close range (420m), the size of the filter covariance is such that non-linear effects become significant, causing the true relative position errors to exceed the 3σ bound.

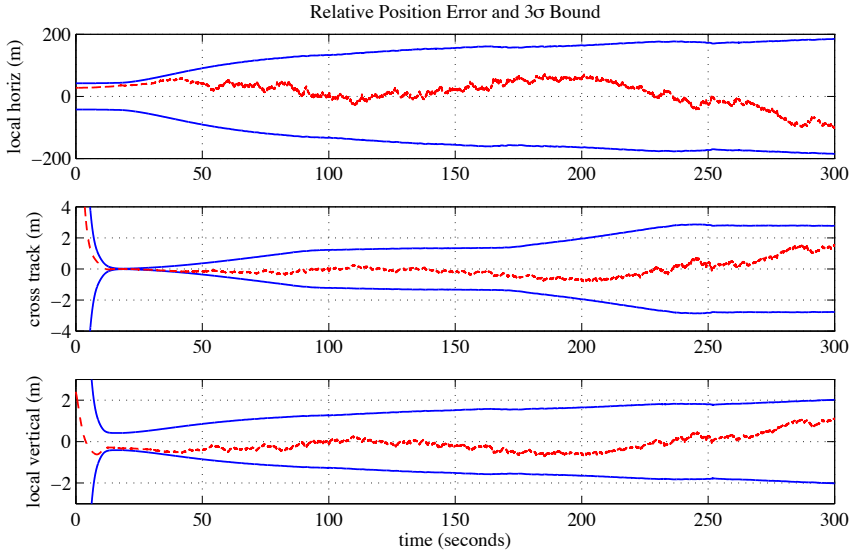


Fig. 8.8: With poor accelerometers at a far range (1620m), the non-linear effects are not significant and true relative position errors stay inside the 3σ bound.

8.5 Effect of Line-of-sight Camera Errors

The effect of line-of-sight camera accuracy was quantified by performing a set of runs with nominal settings on the accelerometers and good, average and poor specifications for the LOS camera (see tables 8.3 and 8.9).

As seen in figure 8.9, relative position estimation is not as sensitive to LOS camera errors as it is to accelerometer errors. While the acceleration returns to zero when a burn in finished, the LOS measurement continues to change until it has significantly diverged from the non-perturbed measurement. Thus even a poor LOS camera permits range observability. However, the less accurate LOS camera is slower to estimate range when a thruster is fired. This can be seen in figure 8.9 when a thruster fires at 85 seconds. The good LOS camera improves the range estimate almost immediately. The average LOS camera is a few seconds behind, and the poor LOS camera doesn't improve the range estimate until around 100 seconds.

In this example camera misalignment is not large enough to be a significant problem. However, if the misalignment is large, it can be estimated if the filter has a good *a priori* estimate of the relative state. If misalignment is significant and not properly characterized, then significant errors will be introduced into the state estimate.

Table 8.9: Cases Comparing Effects of Line-of-sight Camera Errors

Study	Thruster Acceleration	Accels	LOS Cam	Chaser Location
Explore effects of LOS camera accuracy	13.33 mm/s^2	Nominal	Good	V-bar, 406 m behind
			Average	
			Poor	

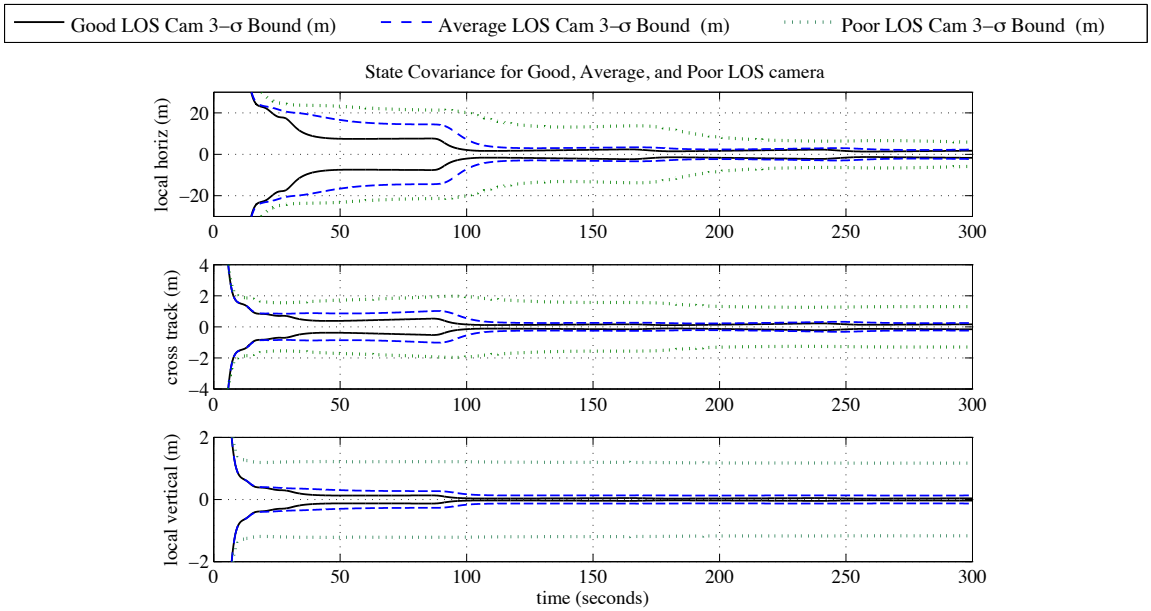


Fig. 8.9: Effect of LOS camera errors on relative position navigation errors (good-solid, average-dashed, poor-dotted).

8.6 Effect of Good, Average, and Poor Sensor Suites

Finally a general trade analysis was performed with high, average, and low cost sensor suites (see table 8.10). These classes correspond to the good, average, and poor sensors listed in tables 8.2 and 8.3. The comparative performance of these suites is shown in figure 8.10. The 3σ error bounds and the true error for each case are shown in figures 8.11 through 8.13. The good sensor suite can resolve range to $\pm 3\,m$, the average sensor suite resolves down to $\pm 25\,m$ and the poor sensor suite struggles to resolve range any better then $\pm 160\,m$ with non-linear effects causing the true error to leave the 3σ bound. The complete estimation histories for each case and all the states are found in Appendix B.

Overall, the good and average sensor suites are sufficiently accurate for range observability. However, the poor sensor suite is not sufficiently accurate to observe range reliably.

Table 8.10: Cases Comparing Effects of Good, Average, and Poor Sensor Suites

Study	Thruster Acceleration	Accels	LOS Cam	Chaser Location
Explore performance of difference sensor suites	13.33 mm/s^2	Good	Good	V-bar, 406 m behind
		Average	Average	
		Poor	Poor	

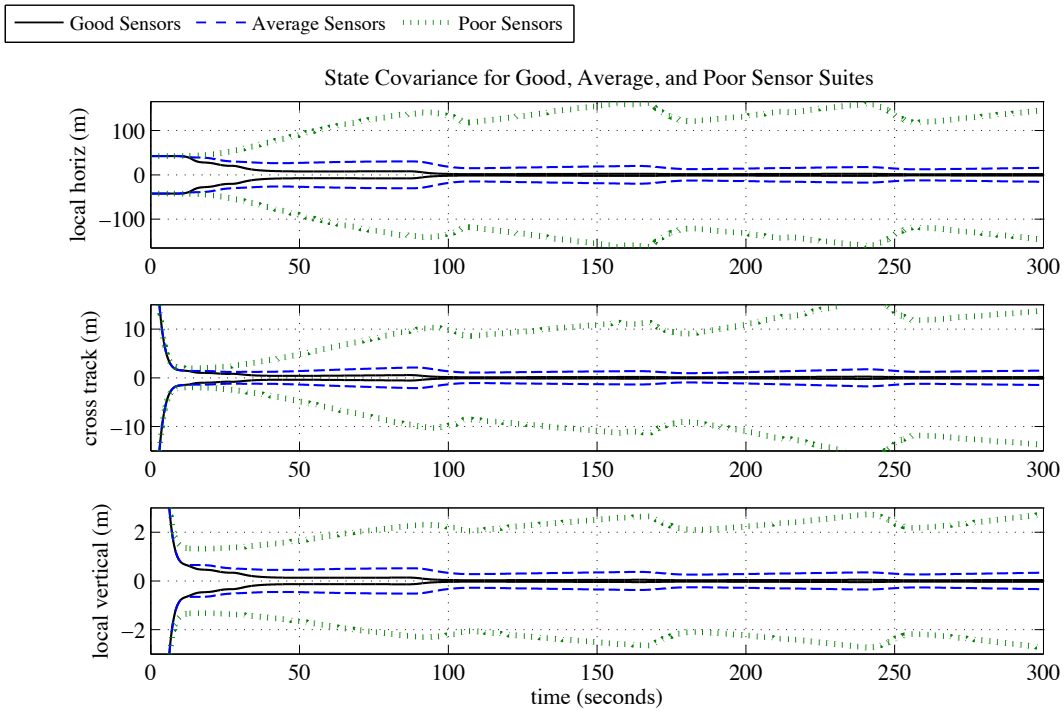


Fig. 8.10: Effects of good, average, and poor sensor suites on relative position navigation errors (good-solid, average-dashed, poor-dotted).

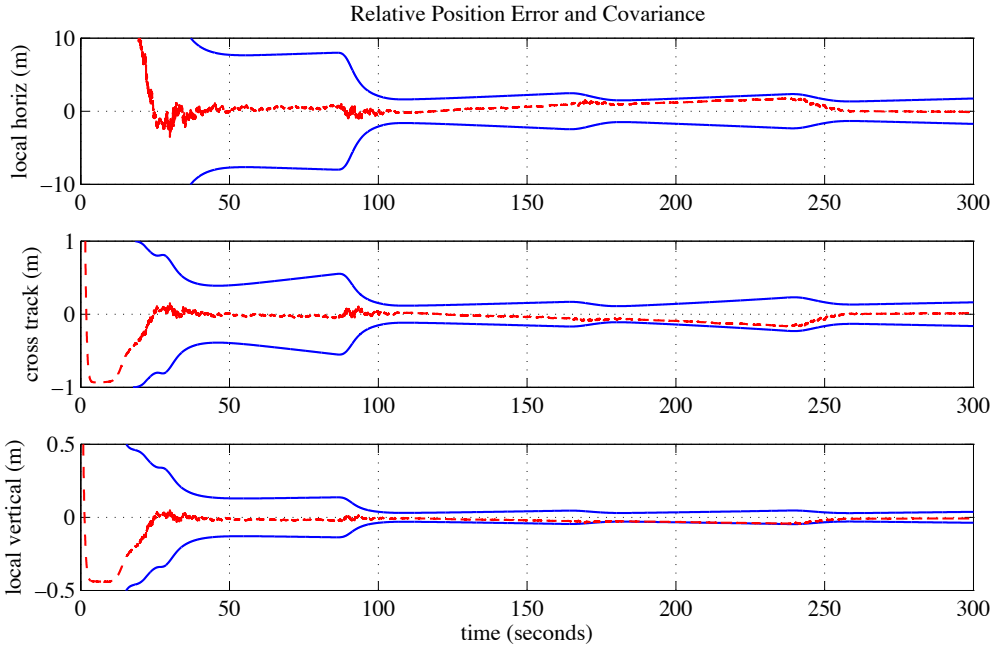


Fig. 8.11: Good sensor suite relative position navigation error and 3σ bound.

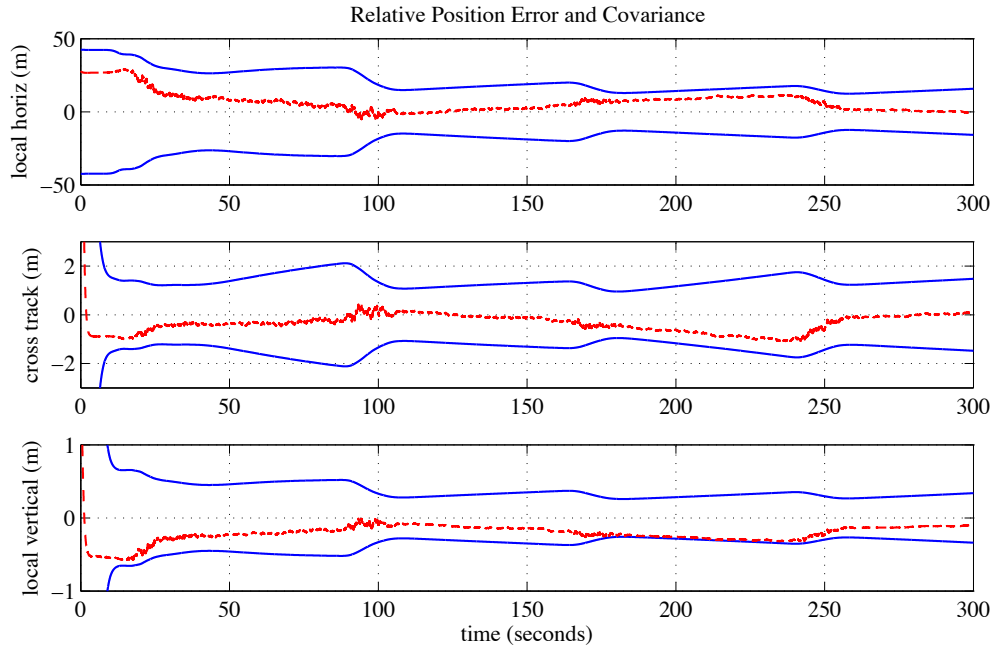


Fig. 8.12: Average sensor suite relative position navigation error and 3σ bound.

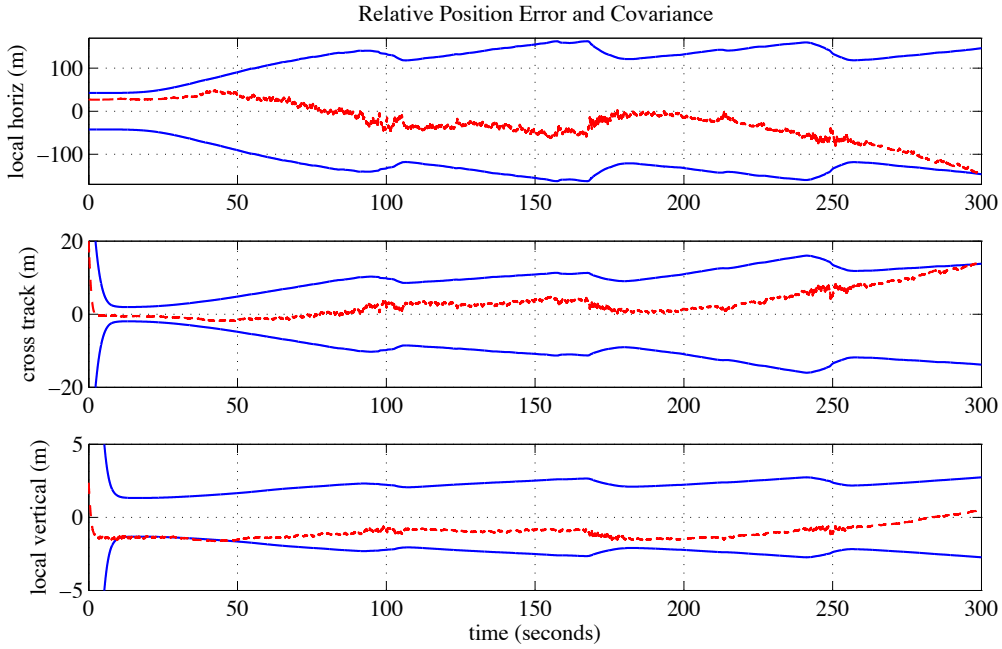


Fig. 8.13: Poor sensor suite relative position navigation error and 3σ bound.

8.7 Alternate Methods for Improving Filter Performance

There are a number of ways to improve filter performance without modifying the sensors. This section examines the effect of increasing the thrust acceleration and selectively integrating the acceleration data. The cases are summarized in table 8.11.

The usefulness of the accelerometer measurements can be improved by increasing the level of thrust acceleration. This causes the acceleration to stand out from the accelerometer measurement noise. Figure 8.14 shows the accelerometer measurements (with average noise and bias) when the thrust is increased from 0.1 N to 5 N. The ontime is shortened accordingly to keep the ΔV the same. The improved filter performance over the original thrust is shown in figure 8.15. Note that the covariance shrinks faster for the larger thrust case than for the original thrust case, but does not shrink by as much. This is because acceleration bias and misalignment are not as well known early on. This implies that it would be good practice to characterize errors like misalignment and bias before attempting maneuvers.

Sections 8.4 and 8.6 showed that poor accelerometers are inadequate when estimating

range. Performance can be improved by integrating the very noisy accelerometer measurements only when the thrusters have been turned on. Whenever the accelerometer data is not being processed, $\partial\dot{\hat{v}}_c/\partial\hat{\beta}$ and $\partial\dot{\hat{v}}_c/\partial\hat{\epsilon}_{acc}$ in equation 7.11, $\boldsymbol{T}_{b\rightarrow I}[\boldsymbol{I}_{3x3} + [\bar{\epsilon}_{acc}\times]][\widetilde{a} - \bar{\beta}]$ in equation 7.2, and $\boldsymbol{Q}_{\eta_c}$ in equation 7.15 are multiplied by zero. The effect is to change $f(x)$ and $\boldsymbol{F}$ so $\hat{x}$ and $\boldsymbol{P}$ are propagated as if there were no accelerometers. In addition, the modified $\boldsymbol{F}$ causes $\boldsymbol{K}$ to be modified so correlations between states grow or shrink depending on whether the accelerometer data is being processed.

This alternate method can tolerate high noise accelerometers. The results for the poor sensor suite with no bias on the accelerometers is shown with continuous and selective integration in figure 8.16. While the continuous integration navigation error and covariance grows large, the selective integration navigation error and covariance stays relatively small.

The results for the poor sensor suite with bias on the accelerometers with continuous and selective integration is shown in figure 8.17. The continuous integration results are identical to figure 8.13, where the true errors leave the 3σ bounds. Fortunately, selective integration allows the poor sensor suite to resolve range 3σ to an order of $\pm 30m$! This is a tremendous improvement and most useful when long periods of time go by with no thrust commands. Because the impact of accelerometer noise is eliminated, velocity (and position) uncertainties do not grow as quickly.

The poor sensor suite can produce even better results (range error on the order of $\pm 15m$) by combining large thrusters with selective accelerometer integration. The selective integration results seen in figure 8.17 are further improved by using larger thrusters (figure 8.18).

Table 8.11: Cases Comparing Alternate Methods of Improving Filter Performance

Study	Thruster Acceleration	Accels	LOS Cam	Chaser Location
Results with big thrusters	666.66 mm/s^2	Average	Average	V-bar, 406 m behind
Results with selective Integration	13.33 mm/s^2	Poor-with no bias	Poor	
		Poor		
Results with big thrusters and selective integration	666.66 mm/s^2	Poor		

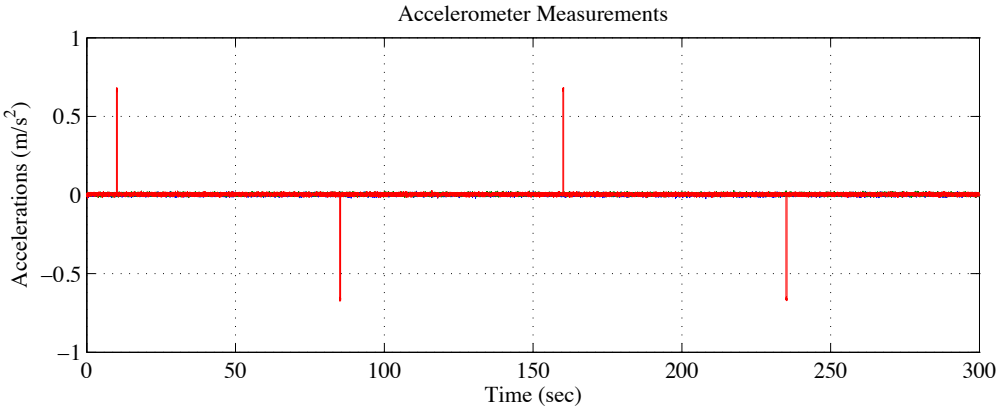


Fig. 8.14: Plot of larger thrust accelerations.

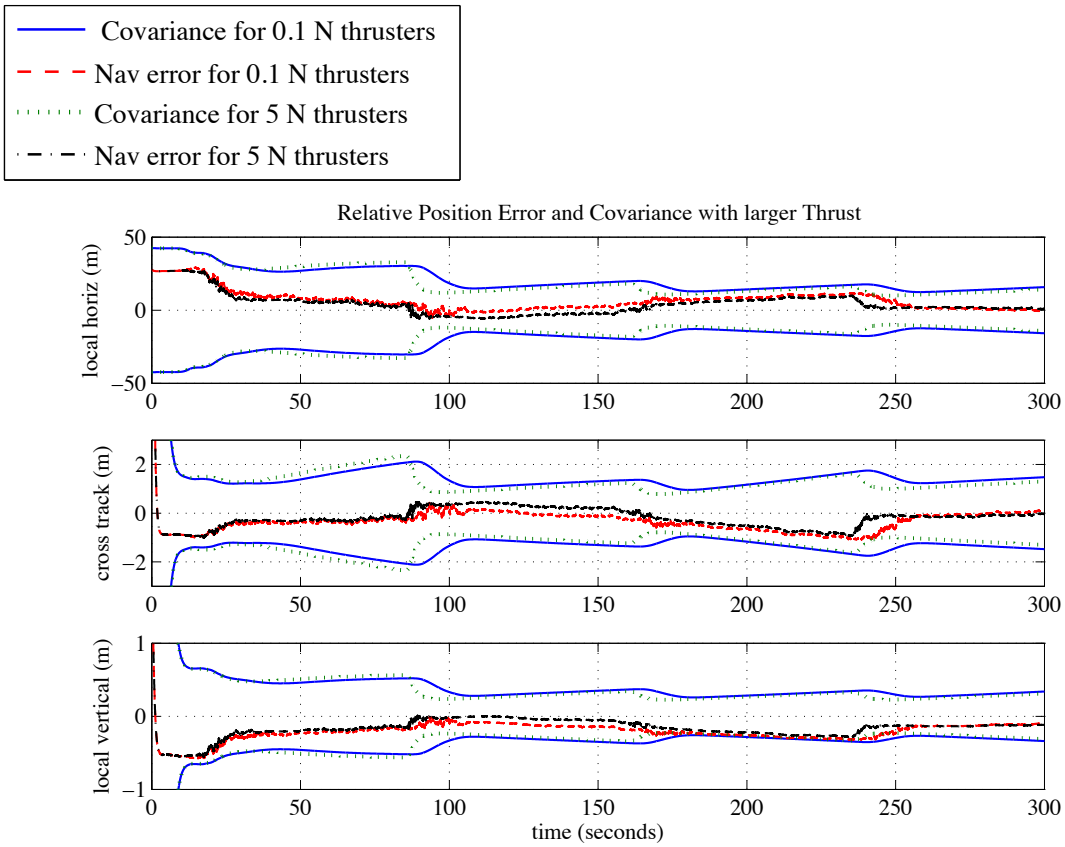


Fig. 8.15: Average sensor suite performance with larger thrusters.

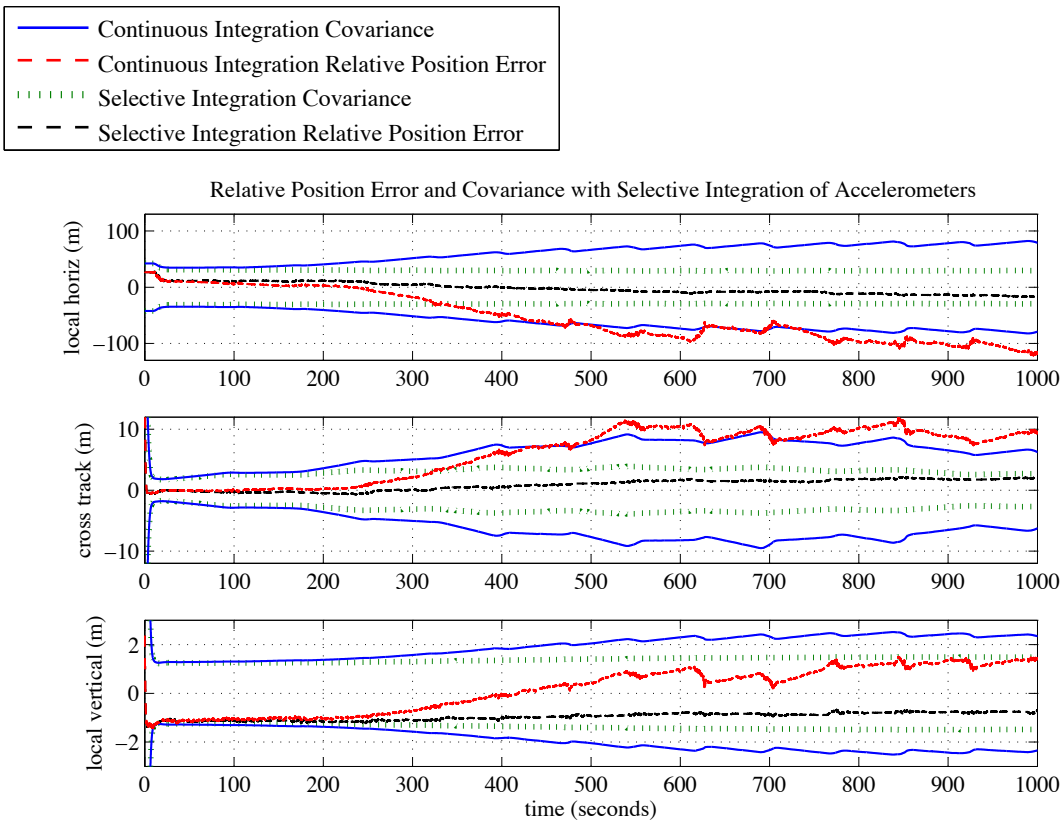


Fig. 8.16: Navigation errors and covariance for continuous and selective accelerometer data integration with the poor sensor suite with no accelerometer bias.

8.8 Conditions Leading to Numerical Failure

Numerical errors occur when an extremely accurate LOS camera causes two components of the relative position covariance to become extremely small, while the range component stays large. The sensor accuracy that will result in numerical problems is a function of the size of the initial position covariance. A LOS camera angle variance of 7×10^{-15} rad^2 ($\sigma = 8.4 \times 10^{-8}$) and a initial position covariance $100\,m^2$ will not cause any problems in a Conventional or Carlson filter. However, if the LOS camera angle variance is decreased to 7×10^{-16}rad^2 ($\sigma = 2.64 \times 10^{-8}$), numerical failure will occur (figure 8.19). Numerical failure will also occur if the LOS camera angle variance is kept at 7×10^{-15} rad^2 but the initial position covariance is increased to $10,000\,m^2$ (figure 8.20). In each case, the Conventional

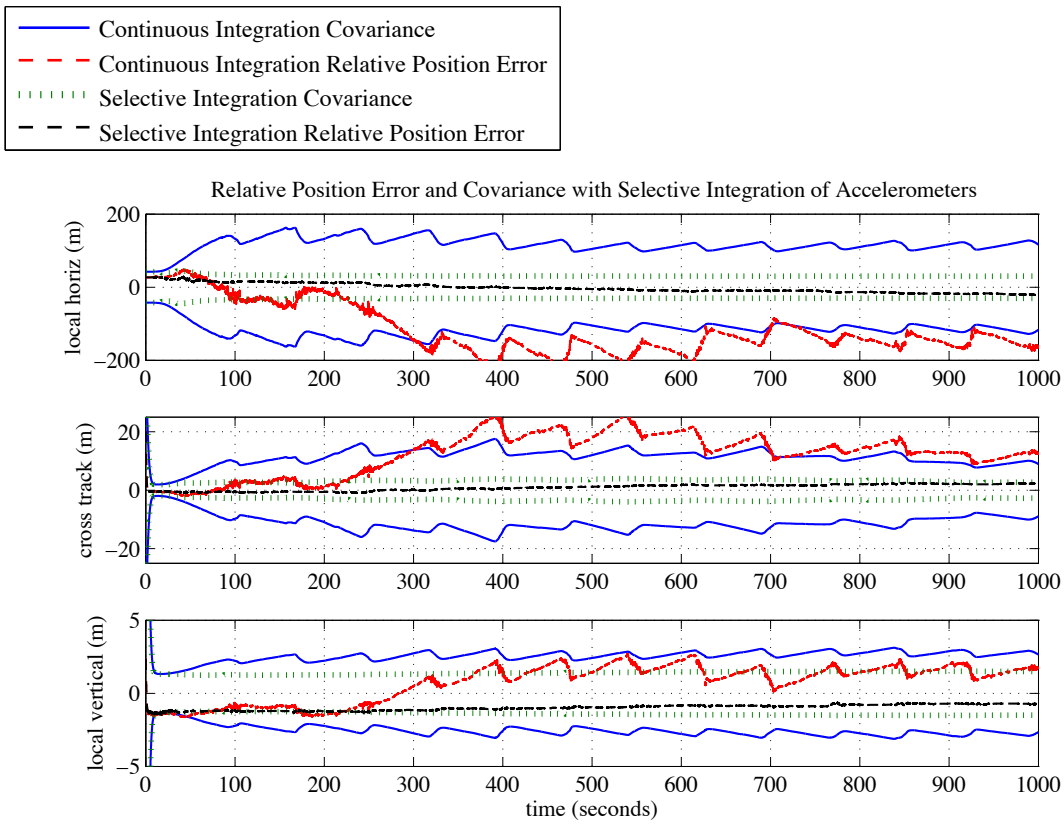


Fig. 8.17: Navigation errors and covariance for continuous and selective accelerometer data integration with poor sensor suite.

results are indistiguishable from the Carlson results until the Conventional filter fails. These cases are summarized in table 8.12. While these camera accuracies are unrealistic, satellites often employ single precision hardware. In that case, realistic camera hardware might cause problems.

Numerical errors can also occur when the inertial covariances are large but the chaser and target position and velocities are highly correlated. In fact, as seen in table 8.12, even the average sensors suite can cause the Conventional Kalman filter to experience numerical failure if the inertial position covariance is $100,000\,m^2$ but the correlations between the target and chaser are defined such that $P_{chaser,target} = 0.9999 \times (P_{chaser} + P_{target})/2$. This is a relatively common situation when only relative measurements are processed in the filter. Under such conditions the inertial state covariances grow without bound, while the relative

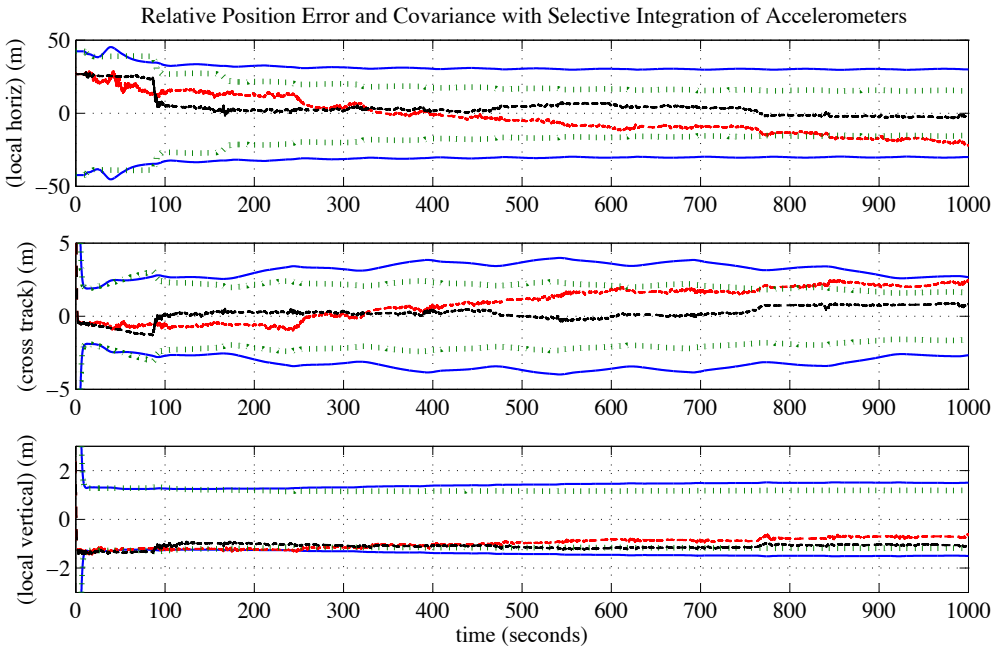


Fig. 8.18: Navigation errors and covariance selective accelerometer data integration with 5.0 N thrusters (nav error=black and covariance=dotted) vs selective integration with 0.1 N thrust (nav error=red and covariance=solid).

state covariance becomes smaller as the chaser and target states become highly correlated.

It should be noted that, as expected, the Conventional and Joseph filters do not fail identically. As seen in figure 8.21, the Joseph filter buys a few more seconds before it also succumbs to numerical errors caused by finite word length. These extra few seconds show the superiority of the Joseph formulation over the conventional formulation (Section 4.2.1).

Table 8.12: Cases Comparing Numerical Performance of Different Filters

Study	Thruster Acceleration	Accels	LOS Cam	Chaser Location
Po=100 m^2, both conventional and Carlson work	13.33 mm/s^2	Good	Very good	V-bar, 406 m behind
Po=100 m^2, conventional fails			Too good	
Po=10,000 m^2, conventional fails			Very good	
Po=10,000 m^2, 0.9999 correlated, conventional fails			Good	
Po=100,000 m^2, 0.9999 correlated, conventional fails		Average	Average	

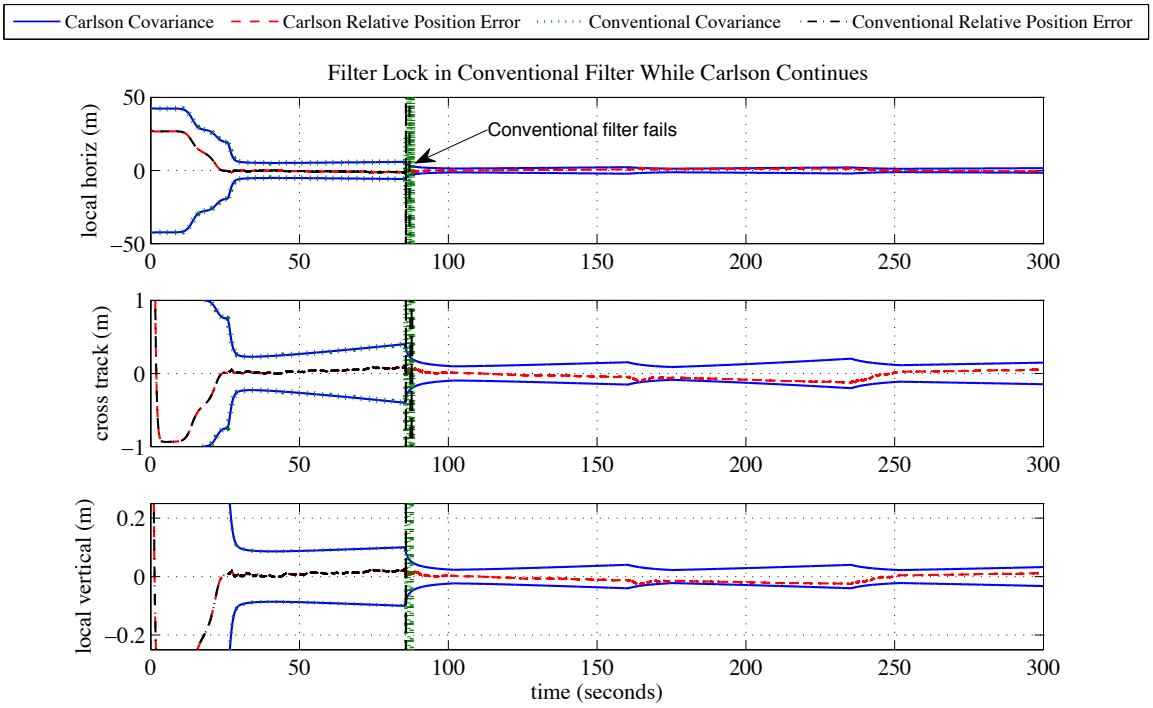


Fig. 8.19: Demonstration of numerical failure in Conventional filter due to more accurate camera measurements while Carlson continues without problems. Note that the Conventional and Carlson results are indistinguishable until the Conventional filter fails.

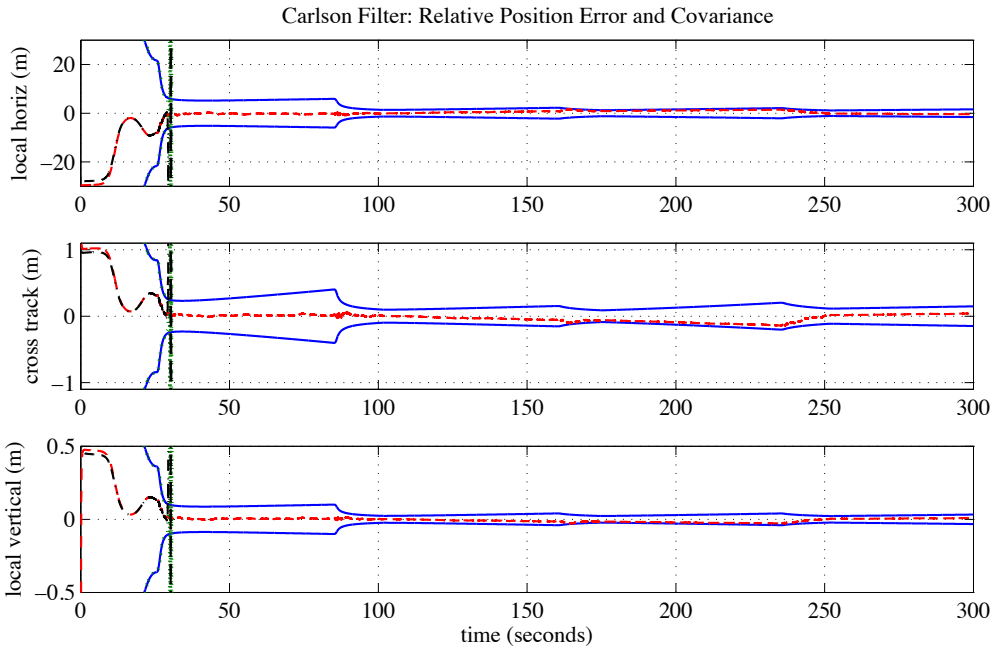


Fig. 8.20: Demonstration of numerical failure in Conventional filter due to larger initial position covariance, while Carlson continues without problems. Note that the Conventional and Carlson results are indistinguishable until the Conventional filter fails.

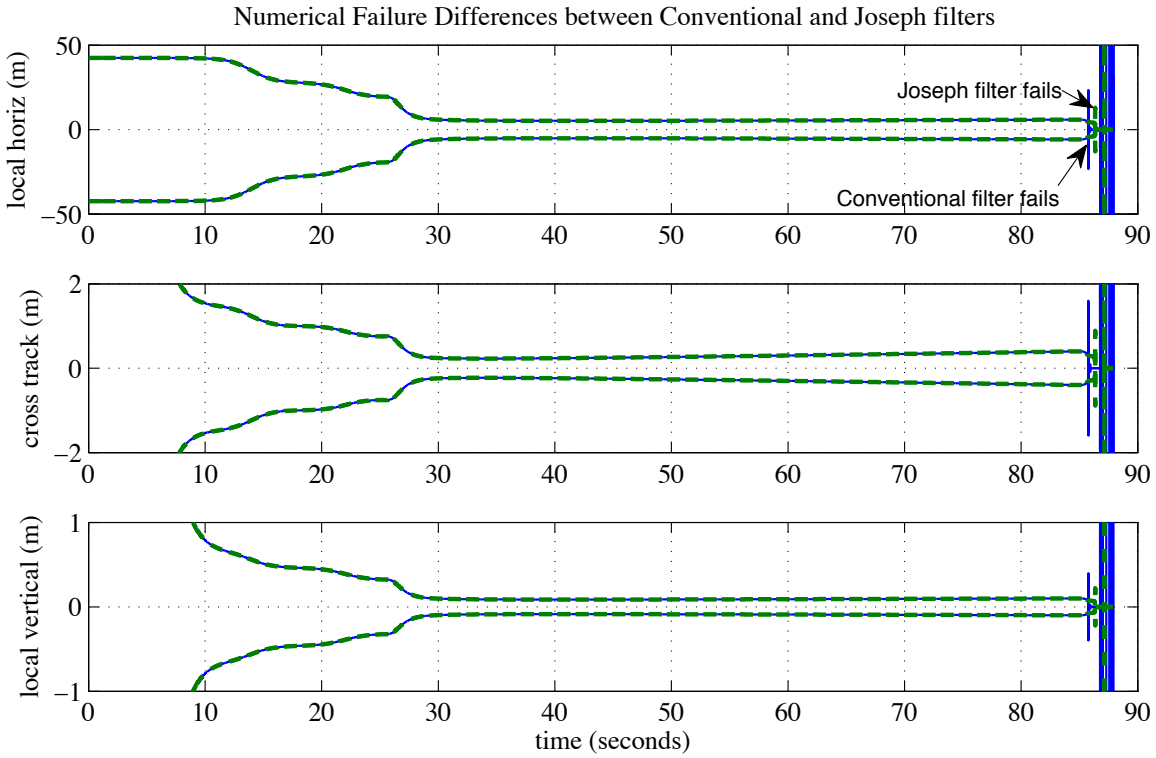


Fig. 8.21: Numerical failure: Conventional vs Joseph filter.

Chapter 9

Closed-Loop Control Simulation Results

In this chapter we examine relative navigation performance and relative position control performance in a closed loop GN&C environment. A position-derivative (PD) controller is implemented to maintain a "station-keeping" condition in the LVLH frame by driving the estimated position and velocity to the desired position and velocity. This presents a problem if the estimated position and velocity exhibit erratic behavior because of poor measurements forcing the thrusters to fire an inordinate amount of time. To better understand filter performance in conjunction with controller performance, several scenarios are used for the closed loop analysis (see table 9.1).

First, the controller is given the true position and velocity derived directly from the simulation. The navigation filter is running, but its results are not used for control purposes. This perfect navigation scenario provides reasonable estimates of how observable the system is when under closed-loop control. Unreasonable cases can quickly be eliminated without massive control issues confusing analysis.

Second, the reasonable cases discovered with the perfect navigation scenarios are analyzed with a true closed system where the filter estimates are given to the PD controller. As expected "chatter" and other fuel wasting patterns are observed, but more realistic results are obtained for a variety of interesting cases.

Finally, in an attempt to reduce fuel use, the controller is modified by adding a position and velocity error deadband determined by the covariance of the estimates coming from the filter. Filter performance and ΔV costs are compared with unmodified scenarios.

To improve simulation runtimes, the simulation stepsize was increased from .01 to .05 seconds and the Euler integrator was replaced with a 4th order Runga-Kutta integrator.

Table 9.1: Closed-loop Cases

Study	Thruster minimum Delta V	Accels	LOS Cam	Chaser Location	Comments
Closed-loop stationkeeping with perfect navigation	0.1 mm/s	Good		V-bar, 500 m back	Min DeltaV is much too small to get range estimate
	1 mm/s				Min DeltaV is still too small to get range estimate
	10 mm/s				Min DeltaV is sufficient to get range estimate
	50 mm/s				Min DeltaV is sufficient to get range estimate, but controllability issues are starting to occur
	10 mm/s			100 m below, 500 m back	Moving off the V-bar provides negligible observability improvements
				200 m below, 500 m back	
Closed-loop stationkeeping with navigation errors	1 mm/s			V-bar, 500 m back	Requires more DeltaV than the perfect navigation counterpart, but exhibits improved observability too
	10 mm/s				
	50 mm/s				
	10 mm/s			100 m below, 500 m back	Moving off the V-bar provides negligible observability improvements
				200 m below, 500 m back	
	10 mm/s			V-bar, 500 m back	Enhanced controllers

The majority of these plots show only the 3σ bounds of the navigation errors. Results with both 3σ error bounds and the true navigation errors for a particular run are shown in Appendix C.

9.1 Perfect Navigation Scenarios

Two studies are conducted with the perfect navigation scenario. The first study examines the effect of thruster minimum ΔV on navigation errors. The second study examines the effect of station keeping below the V-bar on navigation errors. In both cases the PD controller is given perfect estimates of position and velocity derived directly from the simulation and commanded to station-keep 500 m behind the target. Due to the nature of a PD controller and the minimum impulse thrusters, a repeating thrusting pattern is observed.

A range of minimum ΔV are selected for study one (table 9.2). The effect on position control and filter performance is examined. The position control dispersions for $\Delta V = 10\,mm/s$ are shown in figure 9.1. Because the PD controller is actually targeting positions in the inertial frame that have been transformed from the LVLH frame, control dispersions are largly a function of steady state errors innate to a PD control tracking a moving target. In this case there is a steady state error of 10 m in the local vertical, and an oscillatory

error on the same order of magnitude in the cross track. Range hovers around the desired value with chatter very evident.

The control dispersions are more erratic for the larger ΔV cases. To a degree, this is actually desirable, because forced changes in position allow range observability. As seen in figure 9.2, 0.1 mm/s ΔV is too small to observe range. 1 mm/s min ΔV is right on the edge of observability with true errors often leaving the 3σ bound for the perfect navigation case (figure 9.3). 1, 10, and 50 mm/s minimum ΔV thrusters result in the 3σ position covariance bounds shown in figure 9.4. If thruster minimum ΔV is too large then the system becomes unstable due to controllability issues.

Table 9.2: Table of Minimum ΔV Values

min ΔV	stepsize	acceleration	
$0.1\,mm/s$	0.05 s	$2\,mm/s^2$	0.0002039 g
$1\,mm/s$	0.05 s	$20\,mm/s^2$	0.002039 g
$10\,mm/s$	0.05 s	$200\,mm/s^2$	0.02039 g
$50\,mm/s$	0.05 s	$1000\,mm/s^2$	0.102 g

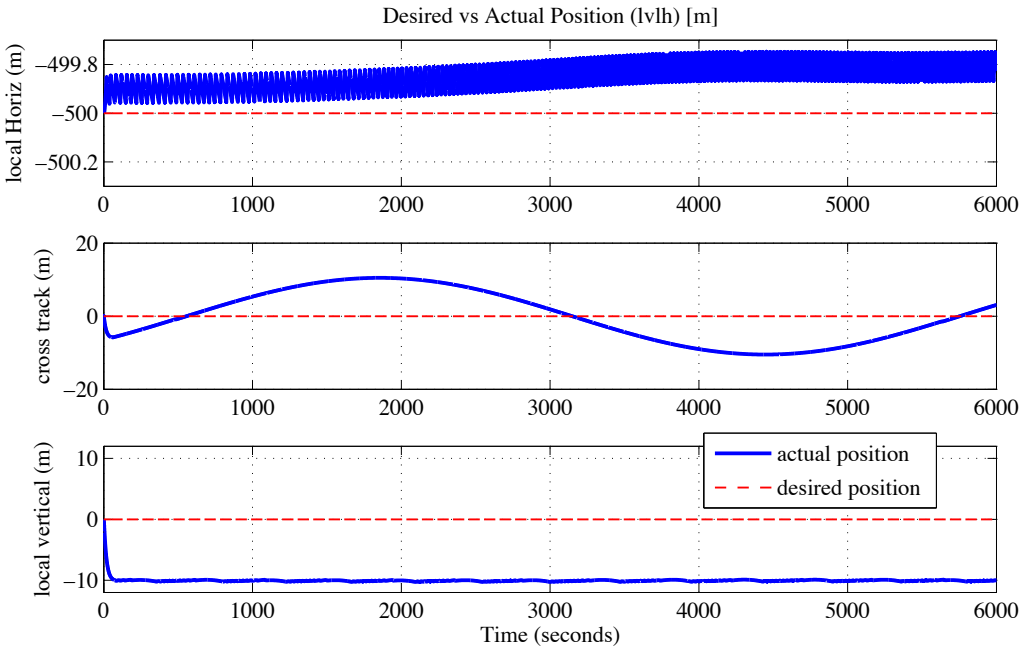


Fig. 9.1: Closed-loop control with perfect nav: Relative position control dispersions with 10 mm/s minimum ΔV thrusters (compare with figure 9.7).

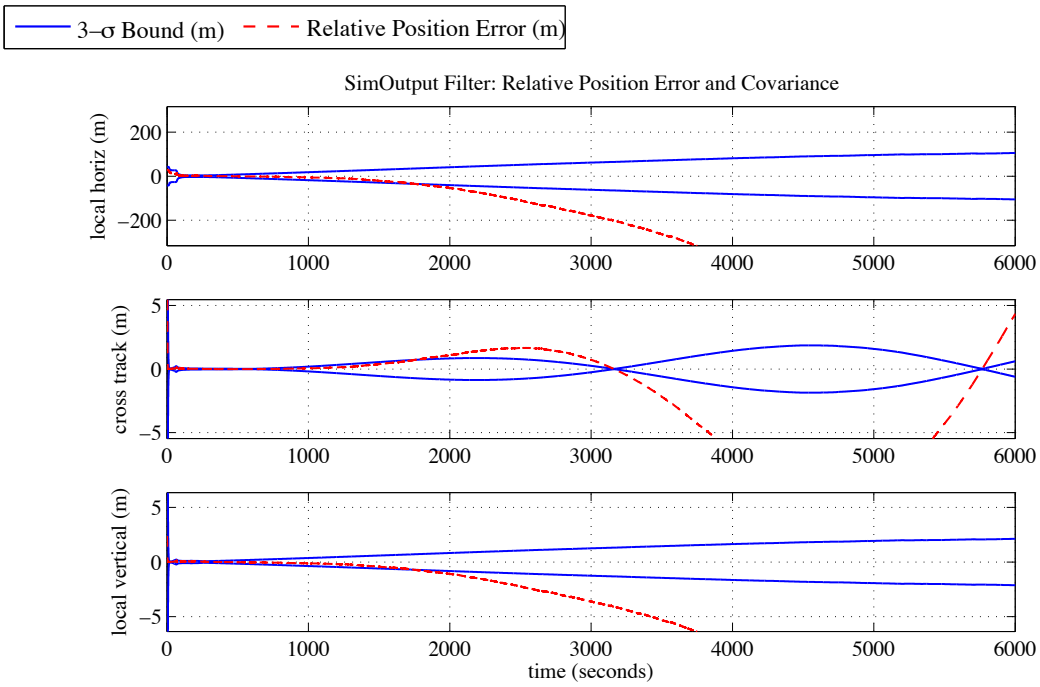


Fig. 9.2: Closed-loop control with perfect nav: Relative navigation error and 3σ bound with 0.1 mm/s minimum ΔV thrusters.

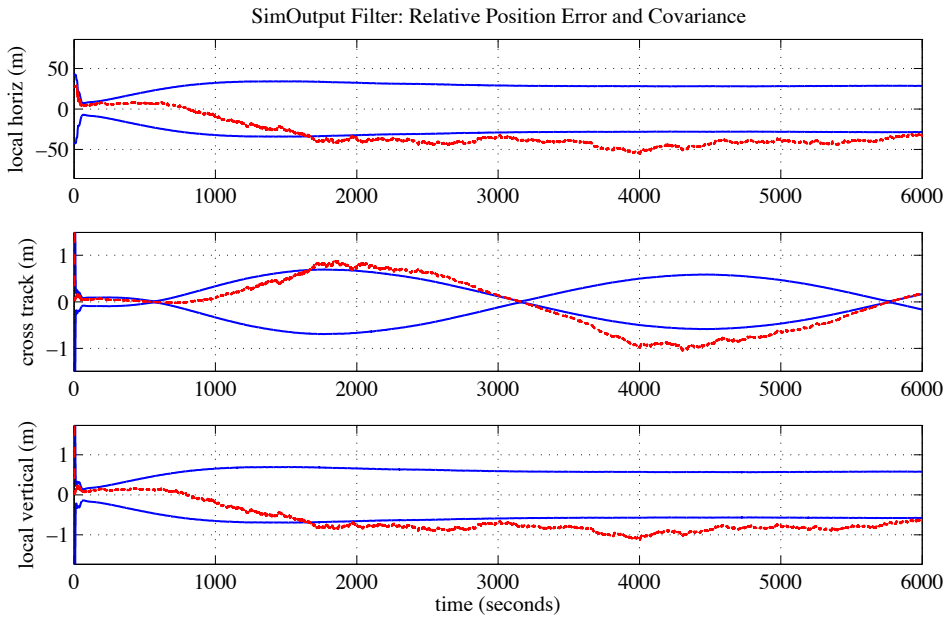


Fig. 9.3: Closed-loop control with perfect nav: Relative navigation error and 3σ with 1 mm/s minimum ΔV thrusters.

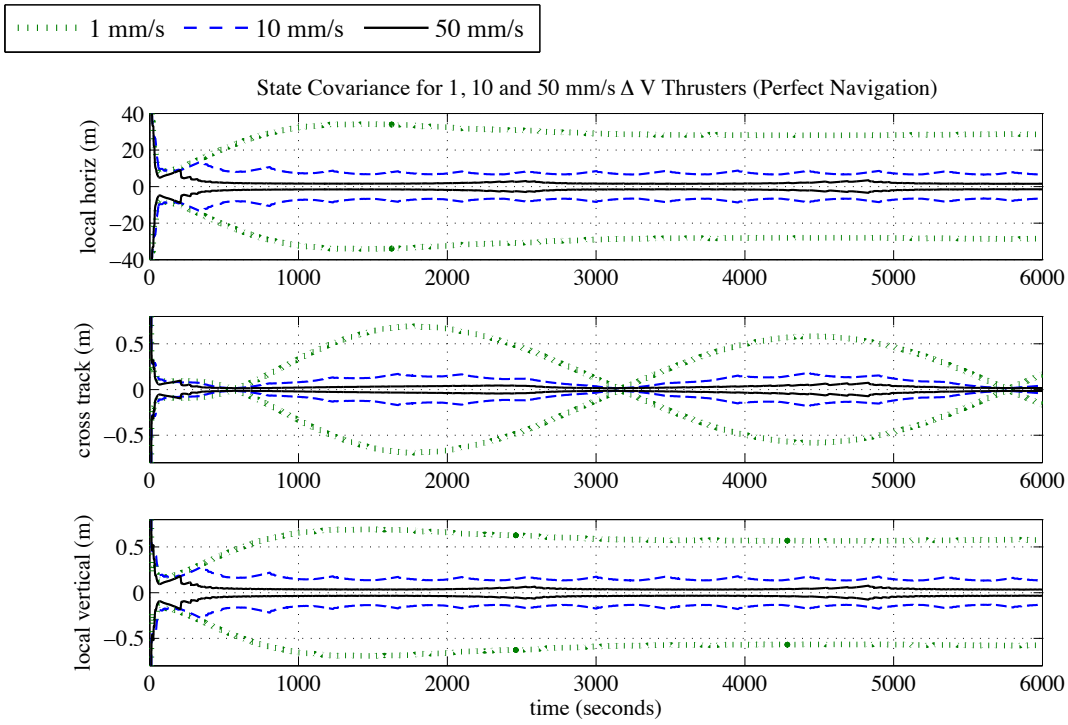


Fig. 9.4: Closed-loop control with perfect nav: Relative navigation error and 3σ bound for stationkeeping on the V-bar at 500m with 1,10, and 50 mm/s minimum ΔV thrusters.

The second study involves moving the location of the chaser. When the chaser is stationkeeping above or below the V-bar, a continuous thrust is required to prevent relative orbital dynamics from moving the chaser away from the target. The magnitude of the required thrust depends on how far off the Vbar the chaser is. This effect can be seen in figure 9.5. When the chaser is on the V-bar, body y-axis accelerations (which correspond to the local vertical) are simply chatter. When the chaser is 200 m below the V-bar, all of the accelerations are in the negative body y-axis direction. The body x-axis sees a similar effect.

The 3σ position covariance bounds for 10 mm/s thrusters with the chaser at locations 500 m behind the target, and 0, 100, and 200 m below the target are shown in figure 9.6. Note that as the satellite moves down, more of the range error is transformed into the local vertical component. Surprisingly, the position estimate does not improve much for this case because the minimum ΔV is still 10 mm/s and they are fired on about the same intervals (figure 9.5). If a different controller was implemented, one that did not fire at regular intervals (due to Chatter) when on the Vbar, dramatic improvement could probably be observed by stationkeeping off of the Vbar.

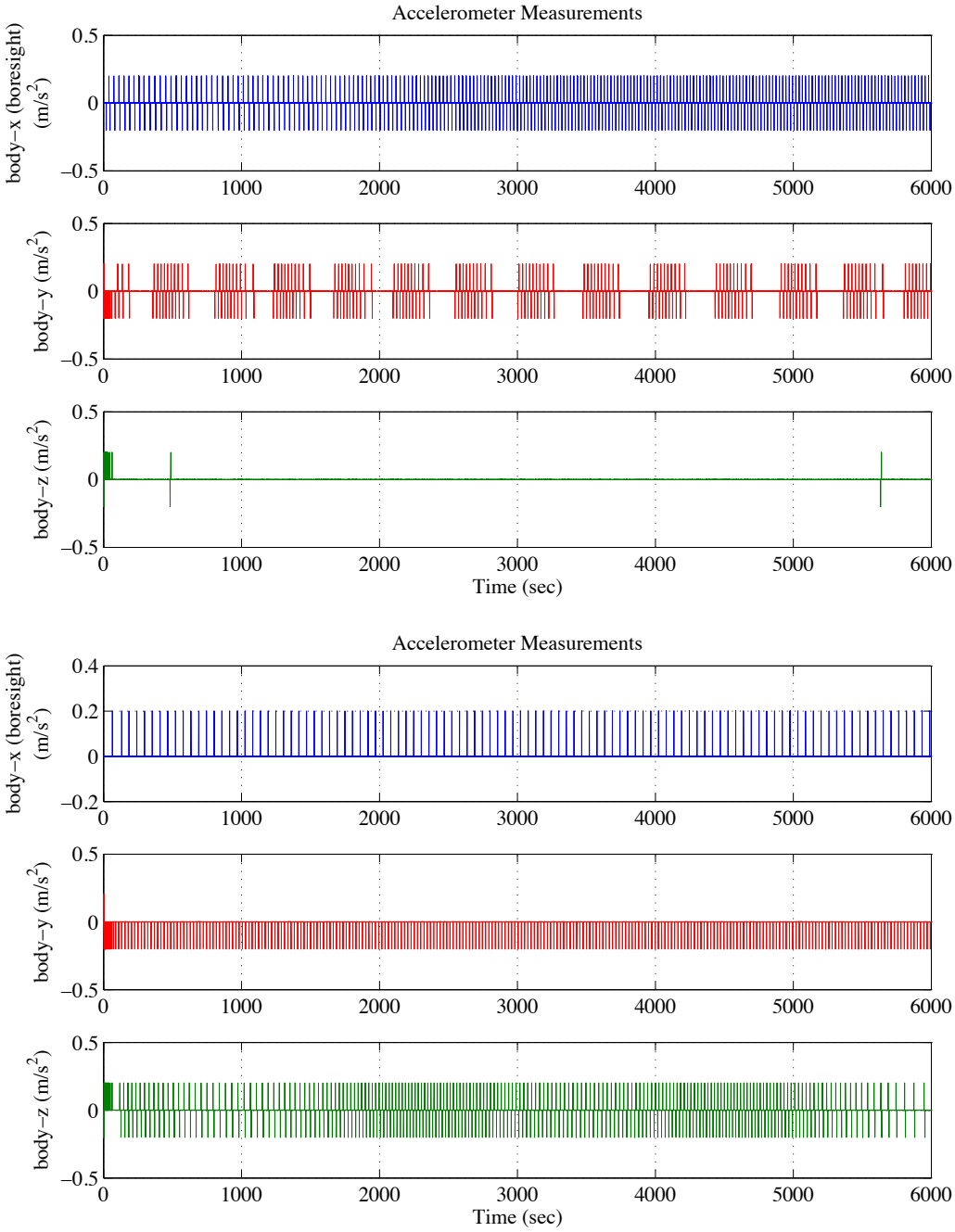


Fig. 9.5: Closed-loop control with perfect nav: Accelerometer measurements with 10 mm/s minimum ΔV thrusters on the Vbar (top) and 200 meters below (bottom).

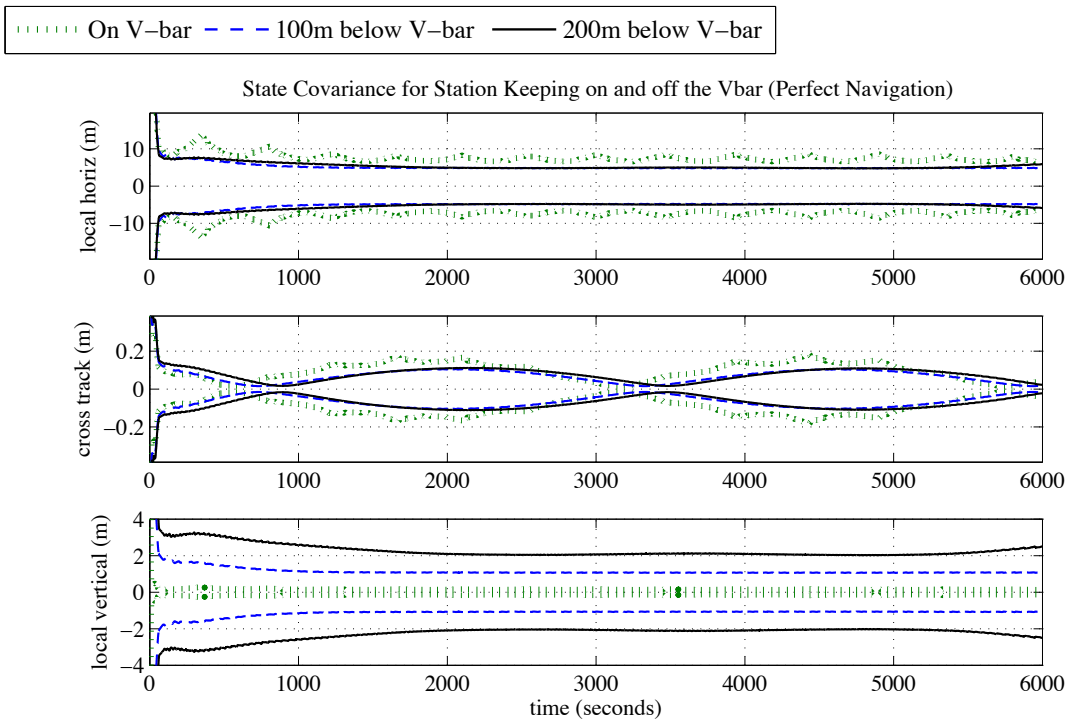


Fig. 9.6: Closed-loop control with perfect nav: Relative navigation performance (3σ), stationkeeping 500 m behind the target and 0, 100 and 200 m below the Vbar, with 10 mm/s minimum ΔV thrusters.

9.2 Close-loop Control with Navigation Errors

The next step is to use the filter estimates to drive the position controller. This causes a number of interesting things to happen. Extraneous thrusters fire (as compared to the perfect navigation scenario) because the estimated location is noisy, while the desired location is static. The controller tries to overcome a position error that promptly disappears once better estimates of location are obtained. The effect is especially severe in the range component, as it has the largest estimation errors.

There are now two performance metrics that are closely linked. One is control dispersions, or the difference between the actual and desired location. Some of this error is attributed directly to the controller, as seen in section 9.1, while the rest of the error is attributed to the second performance metric, the filter state estimation errors. The control dispersions when 1, 10 and 50 mm/s minimum ΔV thrusters are used are shown in figure

9.7. As expected, the 1 mm/s thrusters result in the largest dispersions and the 50 mm/s thrusters the smallest. With the exeption of the 1 mm/s thrusters, they compare favorably with the perfect navigation dispersions seen in figure 9.1.

The range dispersion in the 1 mm/s ΔV case is due to poor filter performance in the range component as seen in figure 9.8. Comparing figure 9.8 with figure 9.4 shows some interesting changes. Because large navigation errors lead to additional thruster firings, filter performance is better when navigation errors are injected into the PD position controller.

As predicted by its perfect navigation counterpart, the change in position study (see Section 9.1 for description) does not show much improvement by moving off of the V-bar (figure 9.9).

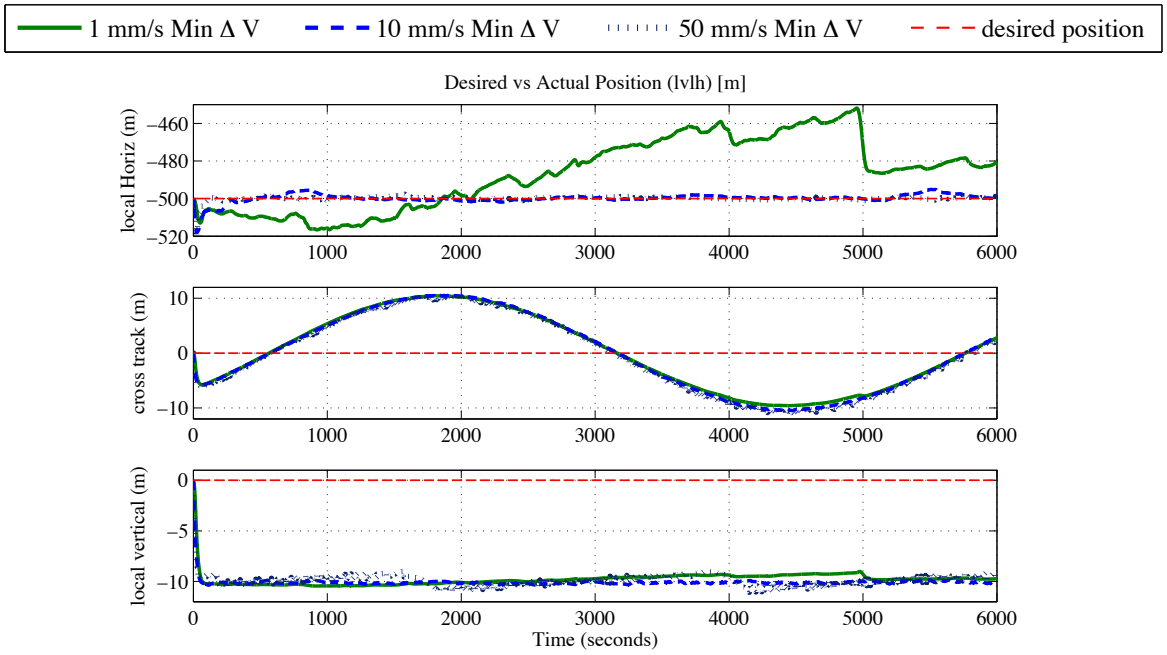


Fig. 9.7: Closed-loop control with nav errors: Relative position dispersions for station keeping on the Vbar at 500m with 1, 10 and 50 mm/s minimum ΔV thrusters (compare with figure 9.1).

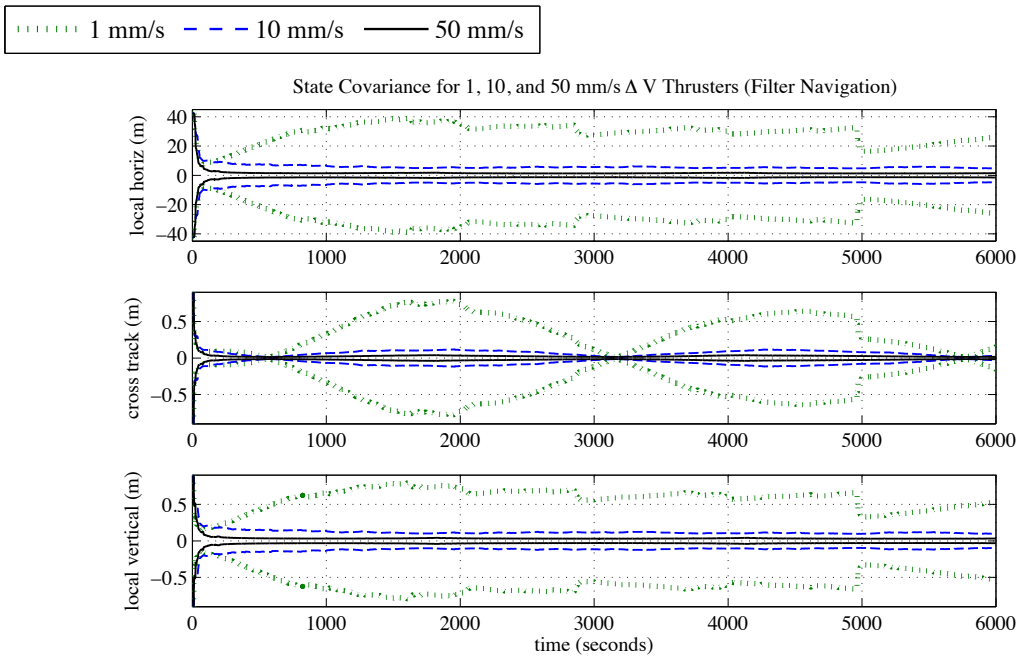


Fig. 9.8: Closed-loop control with nav errors: Relative navigation performance error (3σ) for station keeping on the Vbar at 500m with 1, 10 and 50 mm/s minimum ΔV thrusters (compare with figure 9.4).

9.3 Modified Controller Scenarios

The biggest problem with the AON filter in conjunction with the PD position controller is the tremendous amount of fuel wasted in trying to correct for navigation error. The controller with navigation errors uses about three times as much fuel as the perfect navigation controller. In this section a number of different controllers are compared in an attempt to reduce ΔV requirements.

One proposed solution is to create a controller deadband on each body axis whose size is dependent on the relative position and velocity covariance on that axis. The enhanced PD controller will not command the thrusters to fire unless the errors are larger then the deadband. The magnitude of the position and velocity error is compared to the magnitude of the square root of the position and/or velocity variance in that component. In the boresight direction (body x-axis), if the error is less then the 3σ value of the covariance, the error is reset to zero. In the other two directions, the deadband is at 1σ. In another version of this controller "enhanced 1σ," all three deadbands are set at the 1σ bound. Another

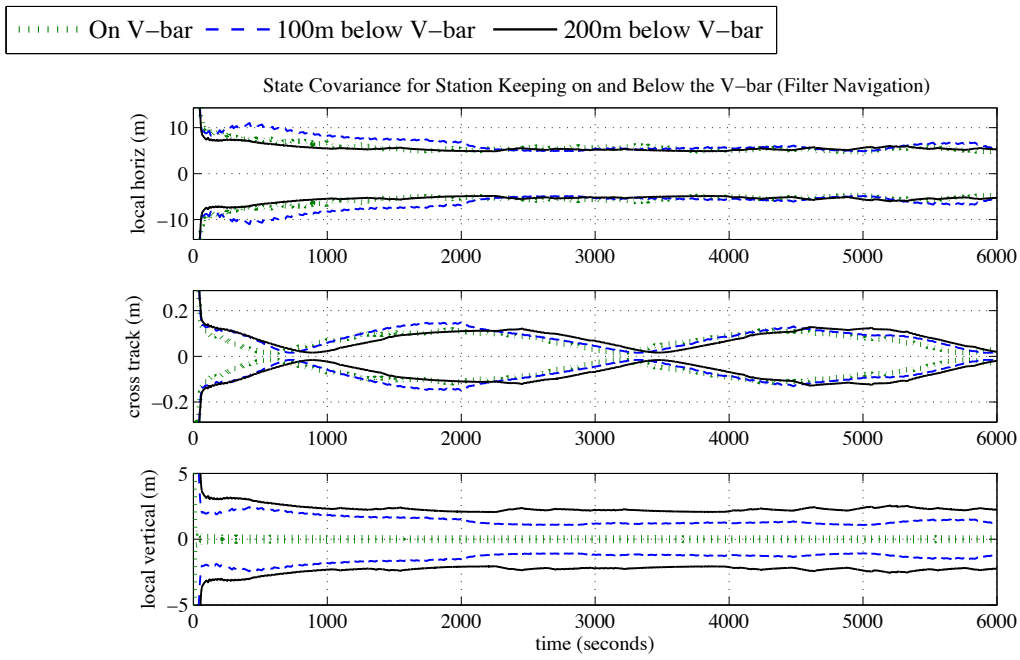


Fig. 9.9: Closed-loop control with nav errors: Relative navigation performance error (3σ), stationkeeping 500 m behind the target and 0 m, 100 m and 200 m below the Vbar, with 10 mm/s minimum ΔV thrusters (compare with figure 9.6).

proposed solution is a "quadratic" control law. In this setup, if the any of the errors are less than the 3σ bound then they are scaled such that

$$\Delta V_{com} = K_p \frac{(r - r_{des})^2}{3\sigma} + K_d \frac{(v - v_{des})^2}{3\sigma}$$

The "quadratic" and deadband control laws are illustrated in figure 9.10.

The ΔV expended for each of the controllers are shown in figure 9.11. Though none of these controllers are superior to the straight PD controller, they illustrate the complex issues that must be solved if a superior control is to be found.

Simulation shows that even though the new controllers fire less often in the range component, they have to correct larger errors, resulting in a higher overall ΔV requirements. Of all the proposed solutions, the enhanced setup came the closest to the original PD ΔV requirements.

Despite these disappointments, the resulting position dispersions as compared with the

plain PD controller are encouraging. The enhanced controller is compared with the original PD in figure 9.12. Only the range component has a little more error. The navigation errors are hardly affected (figure 9.13). This is because the majority of the thruster firings that have been eliminated were in the boresight direction and have essentially no effect on range observability. Comparing the original PD controller accelerations with the enhanced (figure 9.14) shows that the enhanced controller goes more than 1000 seconds before it corrects the boresight component of position. However, the error has grown so large by this point that corrections are more expensive than the nearly continuously firing original controller.

Further tuning, for example, combining the enhanced controller with the quadratic control law so corrections are not so violent when the deadband is exited, might yet enable a modified controller to require less ΔV then the original PD controller.

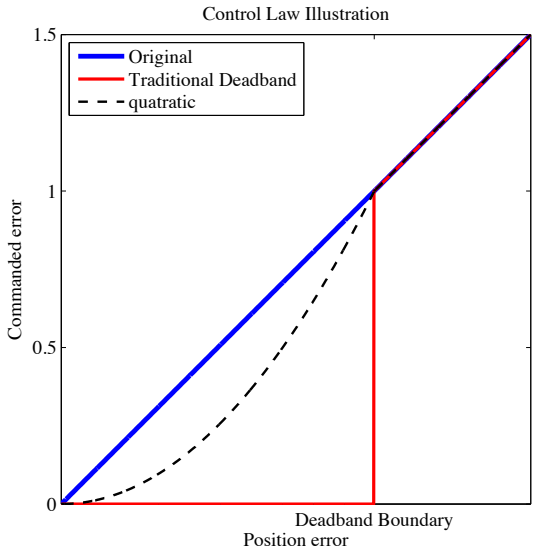


Fig. 9.10: Illustration of controller modifications.

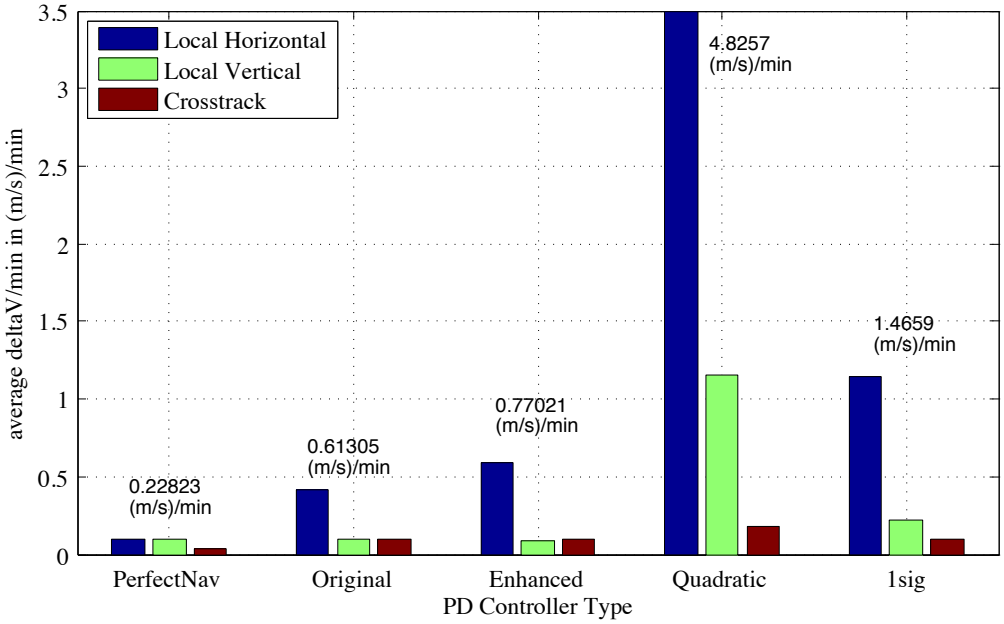


Fig. 9.11: Average ΔV/minute requirements for different controller schemes.

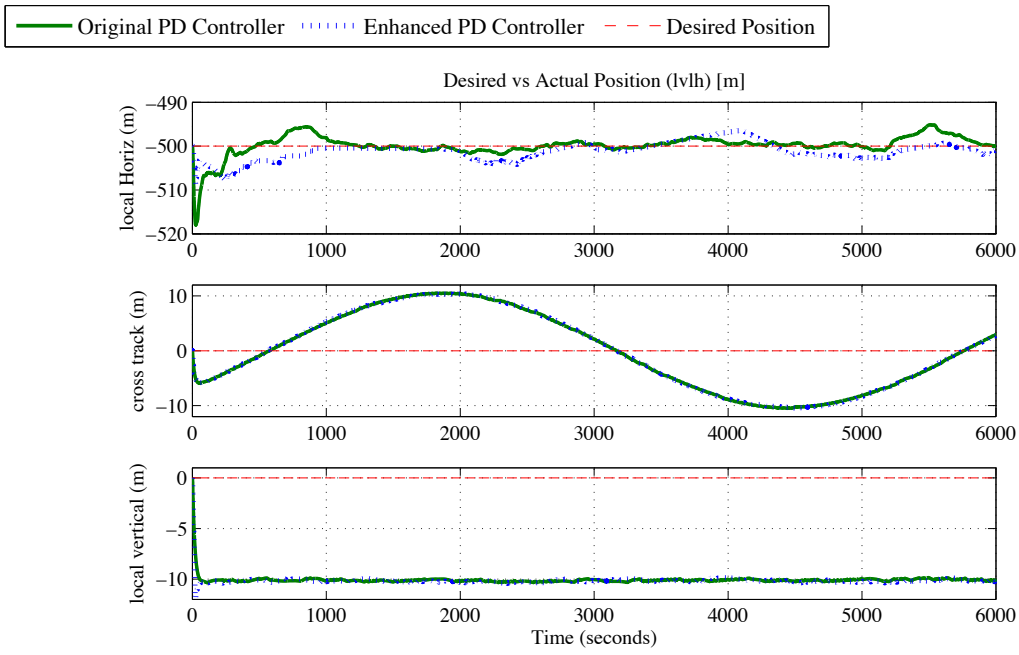


Fig. 9.12: Relative position control dispersions for enhanced PD controller station keeping on the Vbar at 500m with 10 mm/s minimum ΔV thrusters.

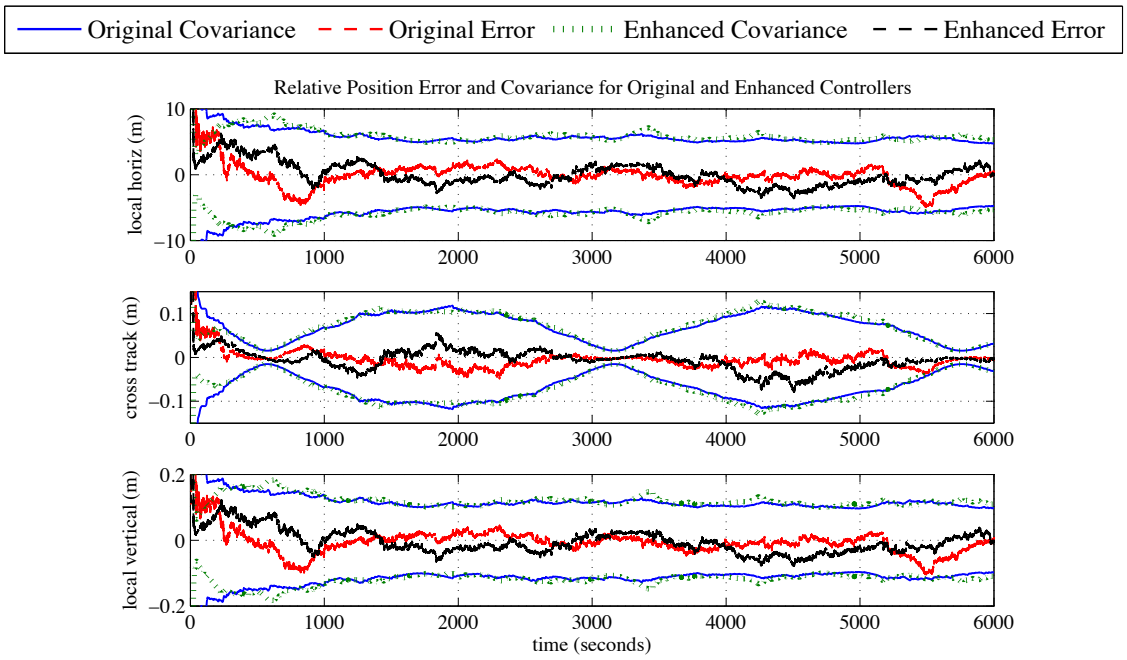


Fig. 9.13: Relative navigation performance (3σ) for station keeping on the Vbar at 500m with enhanced PD controller.

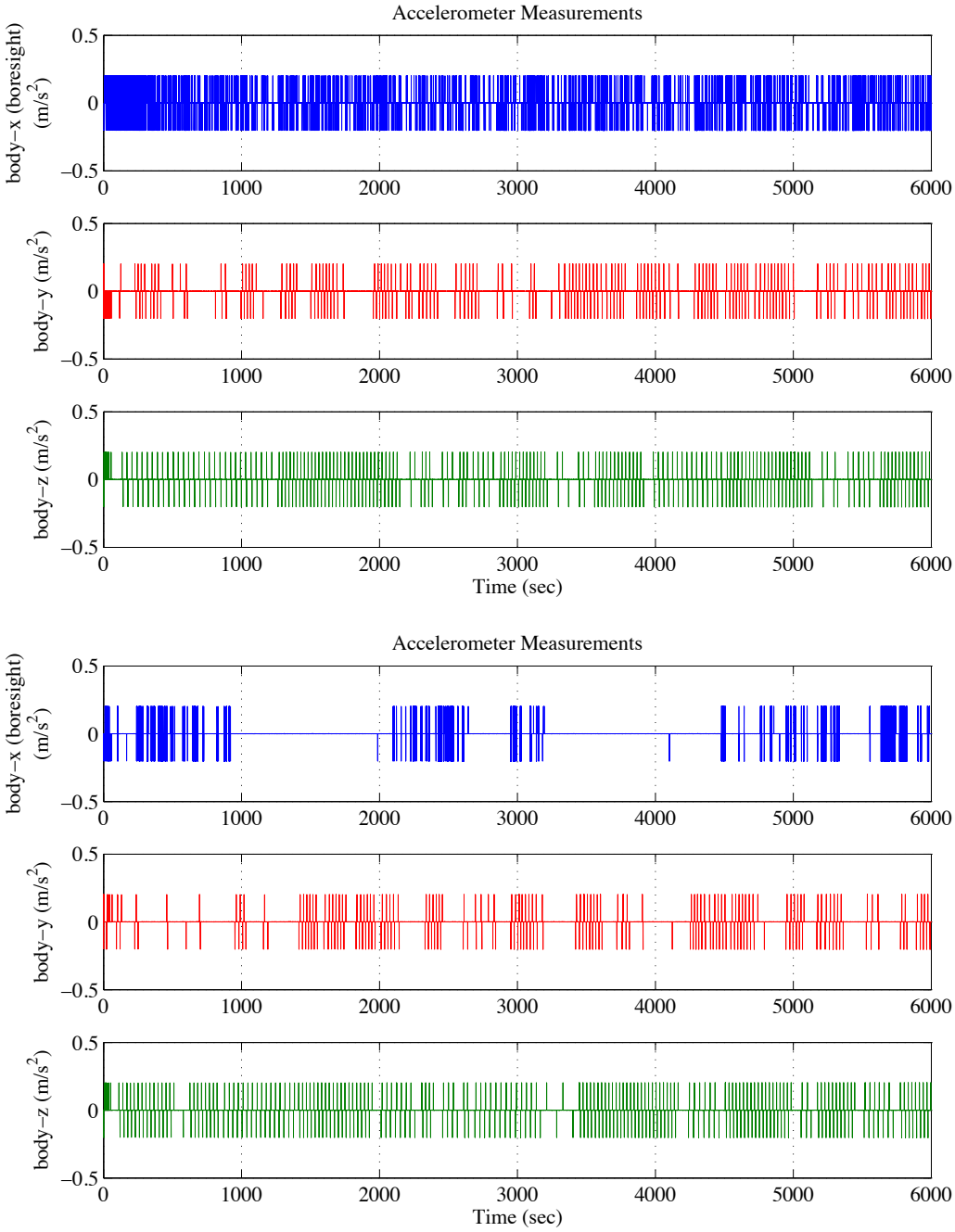


Fig. 9.14: Thruster activity for original PD controller (top) and enhanced PD controller with deadband (bottom).

Chapter 10

Conclusions and Future Work

The aim of this research was to verify that square-root formulations of the Kalman filter would exhibit better numerical stability during angles-only orbital rendezvous operations, and to assess the sensor accuracy required for such operations.

It has been shown that numerical stability of a filter during angles-only operations is a function of both the measurement accuracy and the inertial covariance. Numerical problems result when the filter maintains large inertial state covariances but a small relative state covariance. This happens whenever the target and chaser satellite states are highly correlated. This is a very common occurrence when accurate relative measurements are processed in the filter. If the difference between the largest and smallest eigenvalues in the covariance matrix is larger than the precision of the filter, numerical failure will occur. Because square-root filters have twice the precision of conventional filters, they are much more robust under these conditions.

These findings are supported by results from the high fidelity angles-only navigation simulation. While extremely accurate measurements and/or large initial covariances were required to make the high-fidelity simulation filters fail, actual satellite hardware may have a much shorter numerical word length. Flight computers are often implemented on single precision or even fixed-point (integer) hardware. Under such conditions, only moderately accurate measurements can cause numerical issues. Such implementations have much to benefit from using square-root formulations of the Kalman filter.

It was shown that the thruster acceleration to accelerometer noise ratio has a tremendous impact on whether range can be estimated. If the thrust cannot be detected, then the range estimate cannot be improved. Open-loop analysis showed that if the range covariance is allowed to grow to a significant fraction of the total range, then non-linear effects

render the linear propagation of the covariance invalid, and true errors will often leave the estimated 3σ bound.

In contrast, it was shown that the Kalman filter range estimate is not as sensitive to LOS camera noise because the observation angle continue to grow with time. However, more accurate LOS camera measurements allow observation burns to improve the range estimate more quickly. Future work could examine the long-term impact of unrecognized LOS camera misalignment errors on filter performance. Such work could also examine the effect of star-camera misalignment on filter performance.

Increasing the thruster acceleration (within limits) and only processing the accelerometer measurements when firing those thrusters, can significantly improve the performance of the Kalman filter when processing noisy accelerometers. The selective integration of accelerometers was found to be extremely useful if large amounts of time passed without thruster burns.

Thus, for a wide range of sensor and actuator parameters, it has been shown that range observability is achievable. Key requirements and trends include the following. Thruster accelerations must be distinguishable from accelerometer measurement noise. LOS camera accuracy directly affects how long it takes before an observability burn yields range observability. Using large thrusters and selectively integrating the accelerometer measurements can dramatically improve range observability and filter performance.

Finally, closed-loop analysis proved that station keeping with AON is possible, but requires a sizable ΔV budget. Much work could be done to replace the translational PD controller used for this research with controllers better suited for the peculiarities of angles-only navigation. Initial attempts to modify the PD controller in order to reduce unnecessary fuel use were unsuccessful, but indicate that such controllers should be possible, and will not significantly reduce the ability of the Kalman filter to estimate range.

Overall, this research demonstrates the superior numerical stability of square-root Kalman filters under realistic conditions and verifies that angles-only navigation can be used for short term orbital rendezvous operations as long as quality sensors are used. How-

ever, further work is required to improve fuel efficiency. This requires the development of a controller and guidance system that will 1) perform observability burns only when needed, and 2) refrain from correcting position and velocity errors when the associated navigation errors are large. Balancing these conflicting requirements may not be a simple task.

References

[1] Brown, O. and Eremenko, P., "Fractionated Architectures: Tracking the Path to Reality," Tactical Technology Office, DARPA, Small Satellite Conference, Logan, Utah, August 2009.

[2] Woffinden, D. C., *Angles-only Navigation for Autonomous Orbital Rendezvous*, Ph.D. thesis, Logan, Utah, 2008.

[3] Stastny, N. B., *Autonomous Optical Navigation at Jupiter: A Linear Covariance Analysis*, Master's thesis, Utah State University, Logan, Utah, 2006.

[4] Chari, R. J. V., *Autonomous Orbital Rendezvous Using Angles-Only Navigation*, Master's thesis, Massachusetts Institute of Technology, June 2001.

[5] Passerieux, J. M. and Van Cappel, D., "Optimal Observer Maneuver for Bearings-Only Tracking," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 34, 1998, pp. 777–788.

[6] Monahan, R., *The Optimal Maneuver for Bearings only Target Tracking*, Master's thesis, Royal Military College of Canada, Kingston, Ontario, April 1994.

[7] Fawcett, J. A., "Effect of Course Maneuvers on Bearings-Only Range Estimation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 36, No. 8, Aug. 1988.

[8] Nardone, S. C. and Aidala, V. J., "Observability Criteria for Bearings-Only Target Motion Analysis," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. Vol AES-17, IEEE, 1981, pp. 162–166.

[9] Parten, R. P. and Mayer, J. P., "Development of the Gemini Operational Rendezvous Plan," *Journal of Spacecraft and Rockets*, Vol. 5, No. 9, Sept. 1968, pp. 1023–1028.

[10] Young, K. A. and Alexander, J. D., "Apollo Lunar Rendezvous," *Journal of Spacecraft and Rockets*, Vol. 7, No. 9, September 1970, pp. 1083–1086.

[11] Pearson, D. J., "Shuttle Rendezvous and Proximity Operations," *Space Dynamics*, 1989, pp. 833–851.

[12] Fehse, W., *Automated Rendezvous and Docking of Spacecraft*, Chapter 7.2.5 Example: The Russian Kurs System, Cambridge University Press, 2003.

[13] Mitchell, I. T. and Gorton, T. B., "GN&C Development of the XSS-11 Micro-Satellite for GN&C Development of the XSS-11 Micro-Satellite for Autonomous Rendezvous and Proximity Operations," *Proceedings of the 29th Annual AAS Guidance and Control Conference*, No. AAS 06-014, AAS, Feb. 2006.

[14] Rumford, T. E., "Demonstration of Autonomous Rendezvous Technology (DART) Project Summary," *Proceedings of SPIE Space Systems Technology and Operations*, Vol. 5088, Orlando, FL, April 2003, pp. 10–19.

[15] NASA, "Overview of the DART Mishap Investigation Results," May 2006.

[16] Weismuller, T. and Leinz, M., "GN&C Technology Demonstrated by the Orbital Express Autonomous Rendezvous and Capture Sensor System," *AAS Rocky Mountain Section, 29th Annual AAS Guidance and Control Conference*, No. AAS 06-016, Boeing, Breckenridge, CO, Feb. 2006.

[17] Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME – Journal of Basic Engineering*, Vol. 82, 1960, pp. 35–45.

[18] Aidala, V. J., "Kalman Filter Behavior in Bearings-Only Tracking Applications," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-15, IEEE, January 1979, pp. 29–39.

[19] Maybeck, P. S., *Stochastic Models, Estimation, and Control*, Vol. 1, NavtechGPS, Springfield, VA, 1994.

[20] Bingham, B. E. and Geller, D. K., *Preliminary Orbit Determination for Orbital Rendezvous using Gauss' Method*, Master's thesis, Utah State University, Logan, Utah, 2007.

[21] Woffinden, D. C., *On Orbit Satellite Inspection*, Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 2004.

[22] Sidi, M. J., *Spacecraft Dynamics and Control*, Cambridge University Press, New York, NY, 1997.

[23] Wertz, J. R., editor, *Spacecraft Attitude Determination and Control*, Kluwer Academic Publishers, Norwell, MA, 1978.

Appendices

Appendix A

Symbols unique to Chapter 6: Simulation Development

Symbol	Description/Value	Equation #	Page #
$\boldsymbol{A}_c$	direction cosine matrix for chaser spacecraft	6.22	53
$\boldsymbol{A}_{des}$	desired direction cosine matrix for chaser spacecraft	6.22	53
$\boldsymbol{A}_E$	direction cosine error matrix	6.22	53
$\bar{dv}$	commanded change in velocity	6.21	50
$\boldsymbol{E}_{ortho}$	misalignment matrix	6.8, 6.14, 6.15	46, 47, 47
$\boldsymbol{F}^c$	matrix of unit vectors denoting the thrust direction for thrusters	6.12	47
$\bar{F}_{nominal}$	nominal force due to thrusters	6.14	47
$\bar{F}_{thrust}$	force due to thrusters	6.2	44
$\dot{\bar{h}}$	rate of change of the angular momentum for the momentum wheels	6.3	44
$\bar{h}$	total angular momentum of the momentum wheels in the body frame	6.3	44
$h_1, \ldots h_4$	angular momentum of the individual momentum wheels	6.16	48

Symbol	Description/Value	Equation #	Page #
$\mathbb{I}$	inertia matrix for the satellite	6.1, 6.3	42, 44
K_D	derivative gain	6.21	50
K_{momMan}	momentum management gain	6.16	48
K_P	proportional gain	6.21	50
mass	mass of the chaser satellite (15 kg)	6.2	44
$\boldsymbol{R}^c$	matrix of position vectors of the thrusters in the chaser body frame	6.13	47
R^{cam}	target satellite in camera frame	6.10,6.11	46,46
$\bar{r}_{com}$	commanded position	6.21	50
$\boldsymbol{S}$	Scale factor	6.14, 6.15	47
$\bar{\mathcal{T}}$	total torque on the spacecraft	6.3	44
$\bar{\mathcal{T}}_{nominal}$	nominal torque due to thrusters	6.15	47
$\bar{\mathcal{T}}_{thrust}$	actual torque due to thrusters	6.15	47
$T_1, \ldots T_4$	commanded torque for each momentum wheel	6.16, 6.17	48, 48
T_{cx}, T_{cy}, T_{cz}	commanded torque about each body axis	6.16, 6.17	48, 48
$\bar{v}_{com}$	commanded velocity	6.21	50
$\bar{w}_{acc}$	acceleration process noise normally distributed random variable with noise strength of 1e-12 m^2/s^3	6.2	44

Symbol	Description/Value	Equation #	Page #
$\dot{\bar{\omega}}$	angular acceleration	6.3	44
$\bar{\omega}$	angular rate	6.3	44

Appendix B

Open-Loop Sensor Accuracy Trade Study

The true error and covariance (3σ) is shown in figures B.1 through B.15 for the relative position and velocity, the accelerometer bias and misalignment and the camera misalignment for the good, average, and poor cases discused in Section 8.2.

B.1 Good Sensor Suite Results

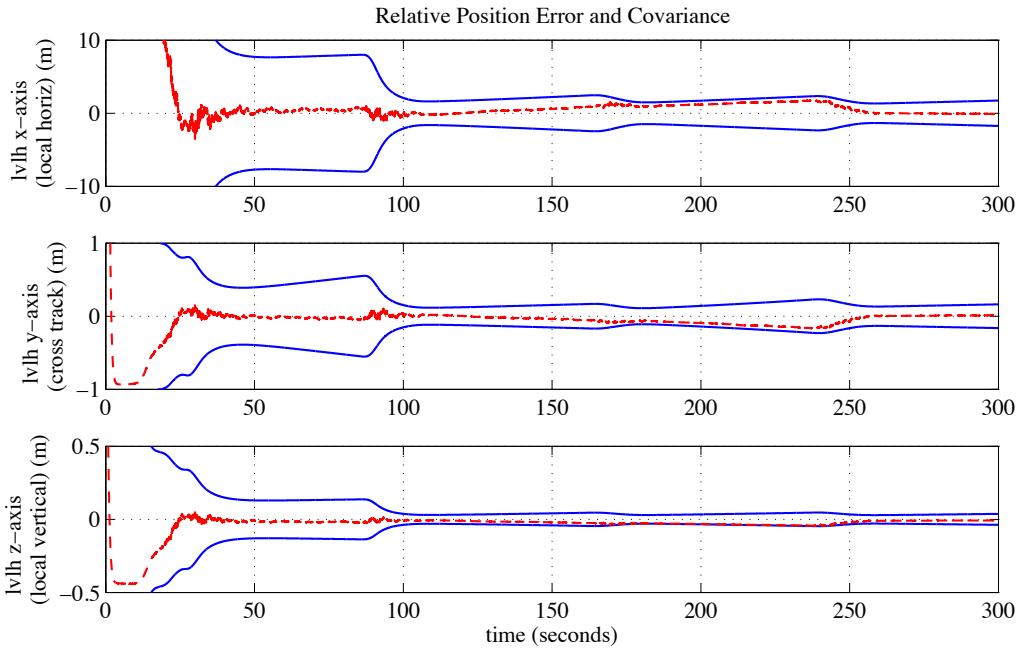


Fig. B.1: Good sensor suite: Relative position error and covariance (3σ).

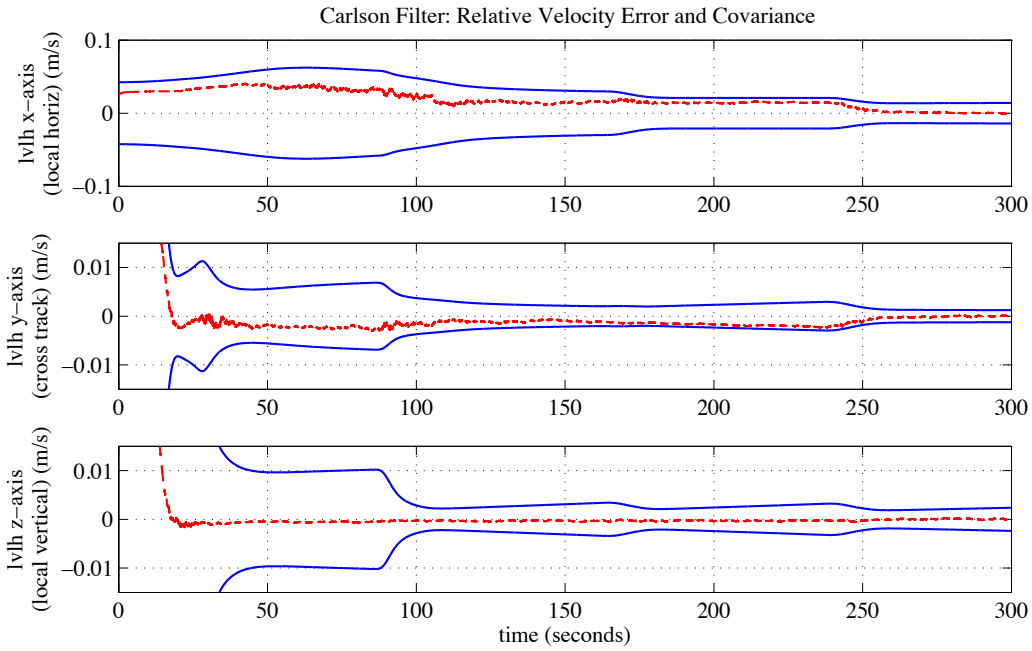


Fig. B.2: Good sensor suite: Relative velocity error and covariance (3σ).

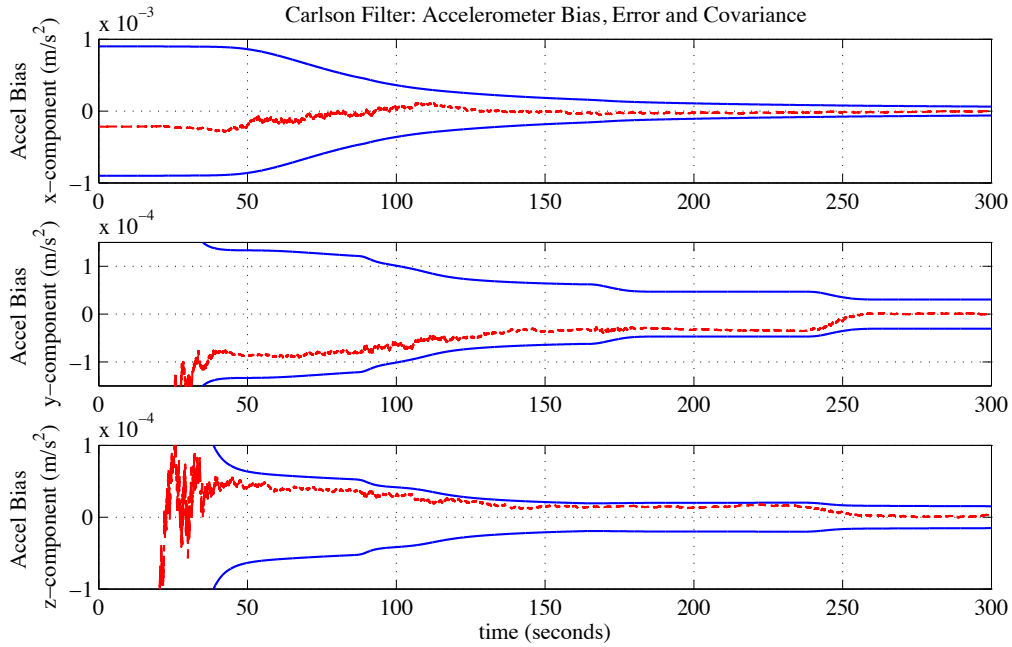


Fig. B.3: Good sensor suite: Accelerometer bias error and covariance (3σ).

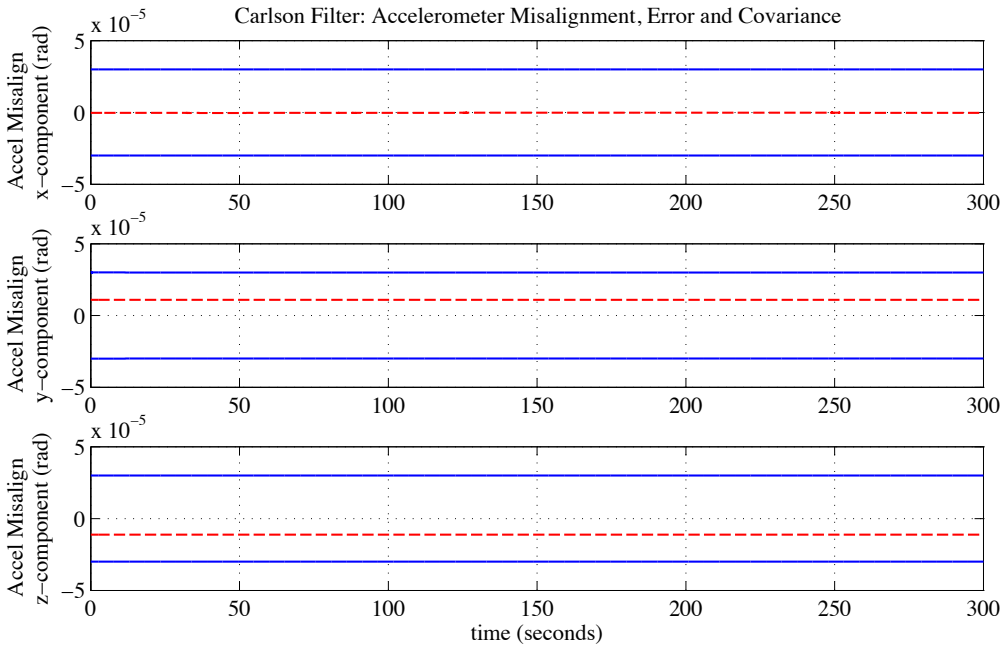


Fig. B.4: Good sensor suite: Accelerometer misalignment error and covariance (3σ).

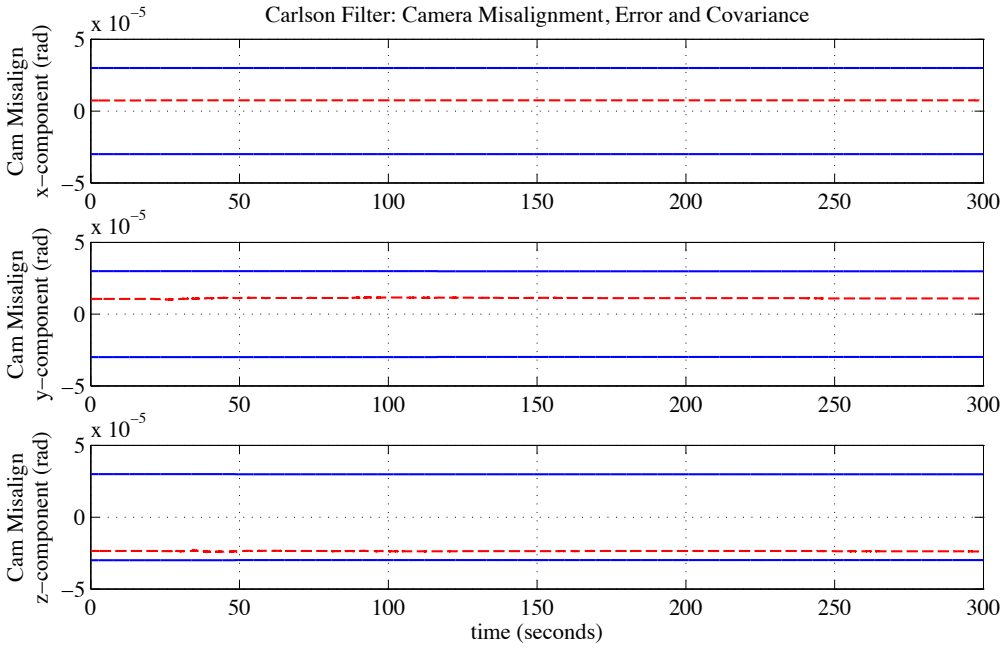


Fig. B.5: Good sensor suite: LOS camera misalignment error and covariance (3σ).

B.2 Average Sensor Suite Results

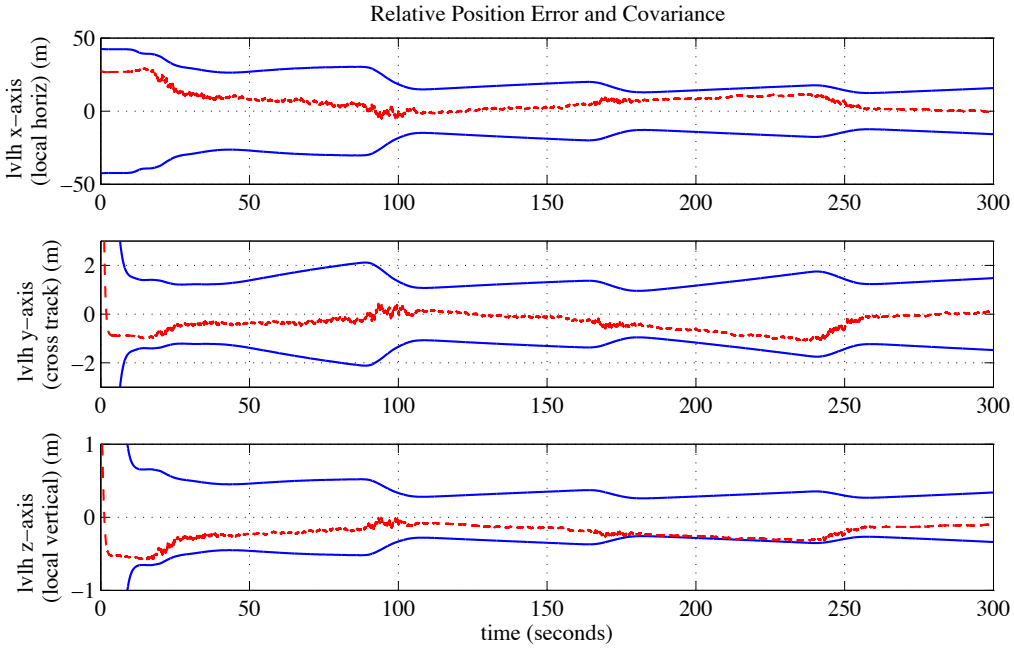


Fig. B.6: Average sensor suite: Relative position error and covariance (3σ).

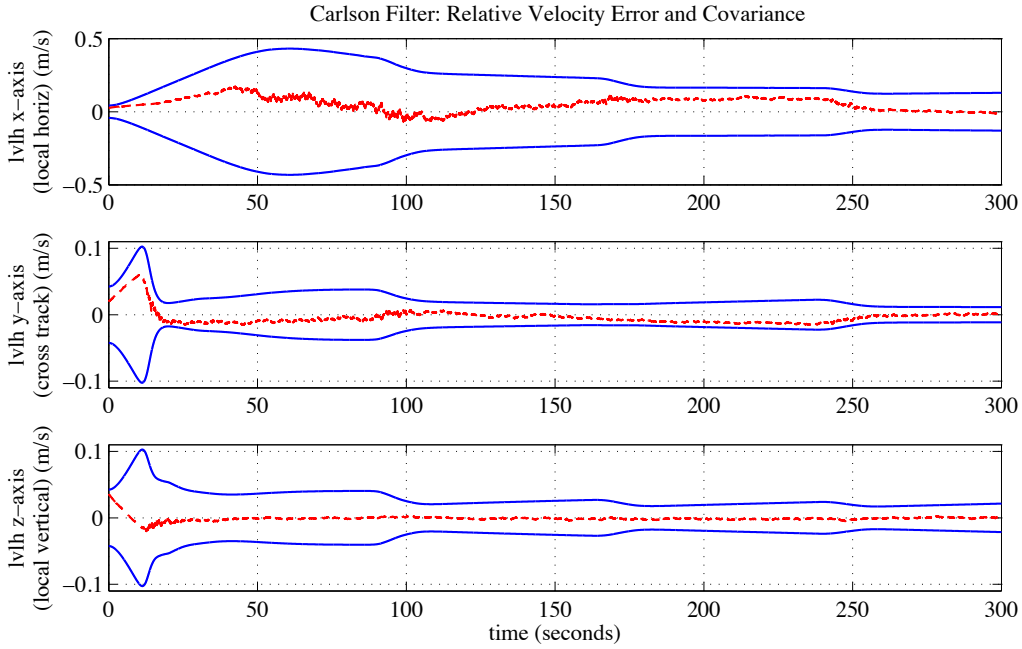


Fig. B.7: Average sensor suite: Relative velocity error and covariance (3σ).

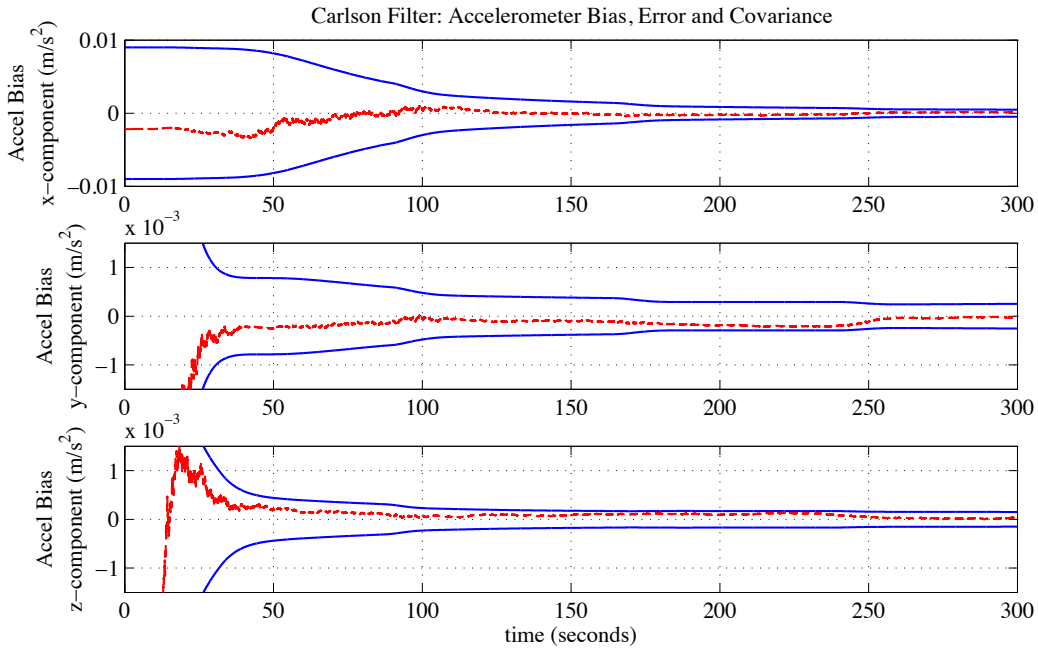


Fig. B.8: Average sensor suite: Accelerometer bias error and covariance (3σ).

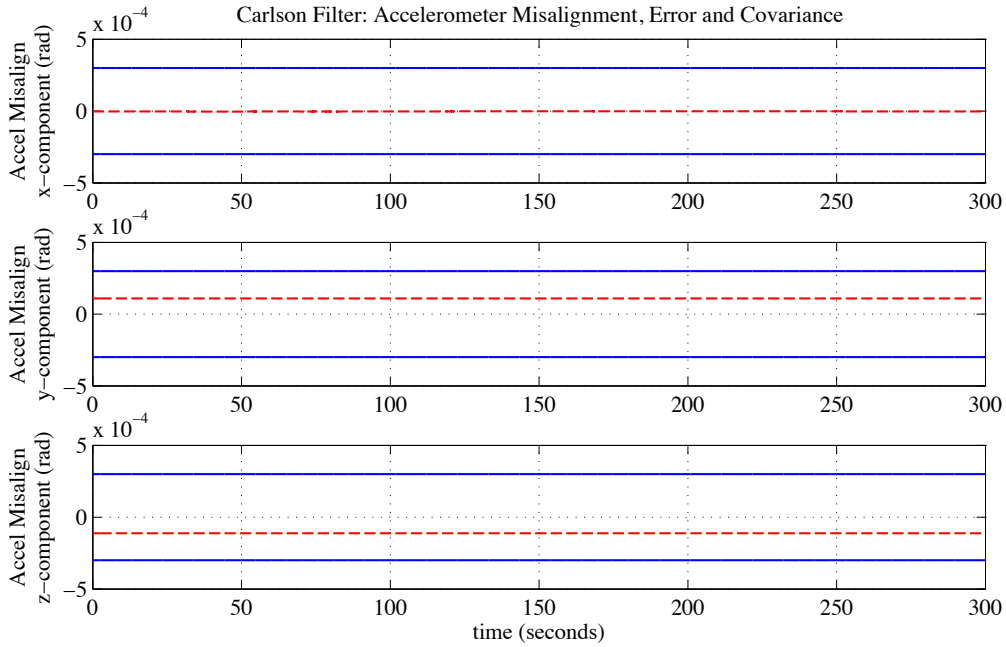


Fig. B.9: Average sensor suite: Accelerometer misalignment error and covariance (3σ).

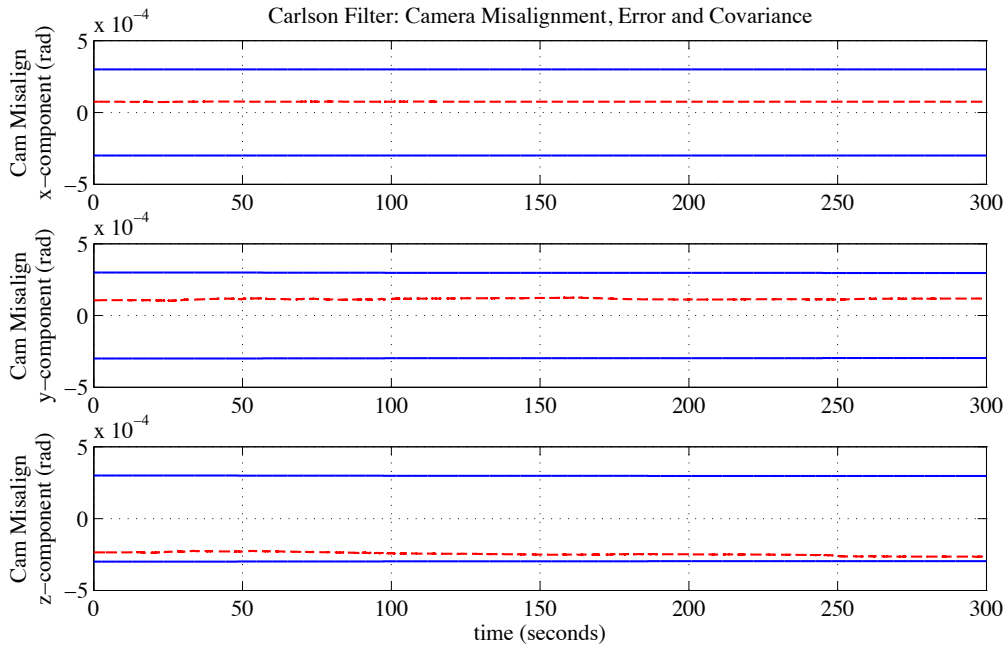


Fig. B.10: Average sensor suite: LOS camera misalignment error and covariance (3σ).

B.3 Poor Sensor Suite Results

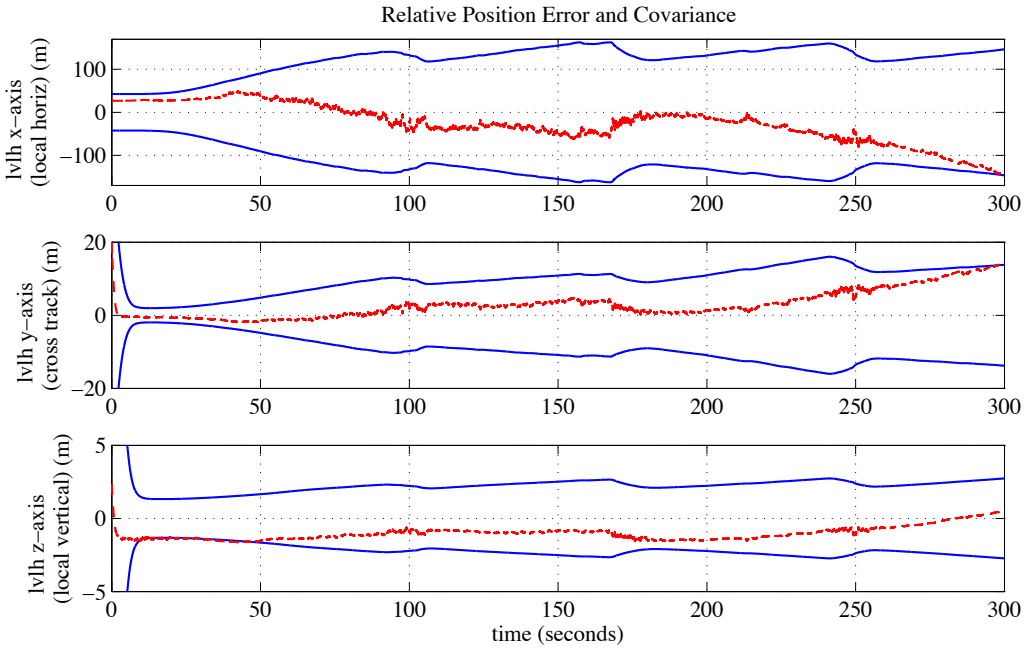


Fig. B.11: Poor Sensor Suite: Relative position error and covariance (3σ).

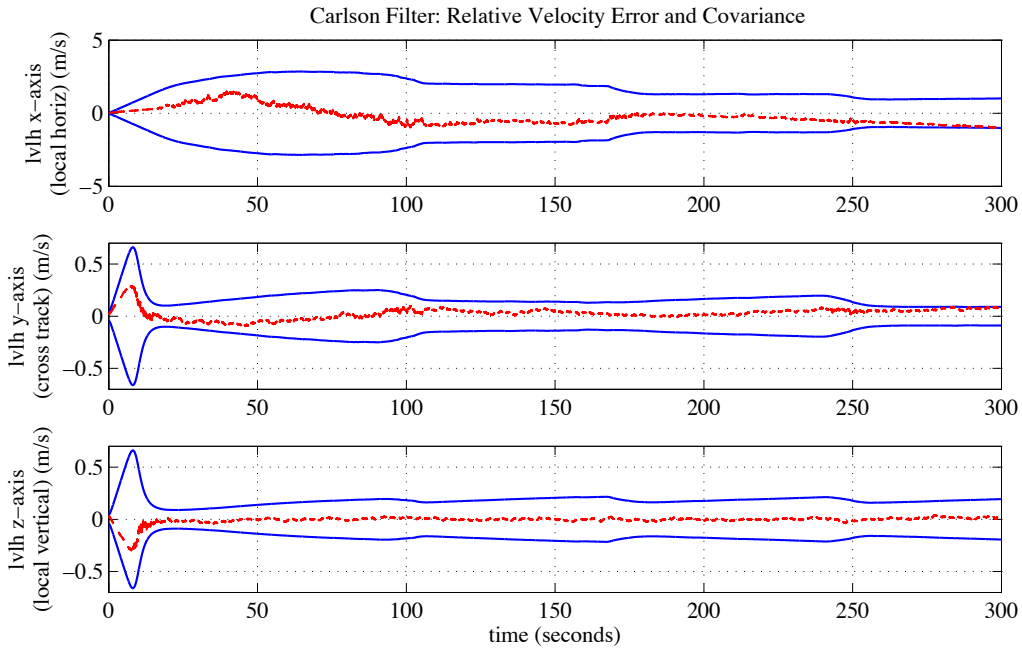


Fig. B.12: Poor Sensor Suite: Relative velocity error and covariance (3σ).

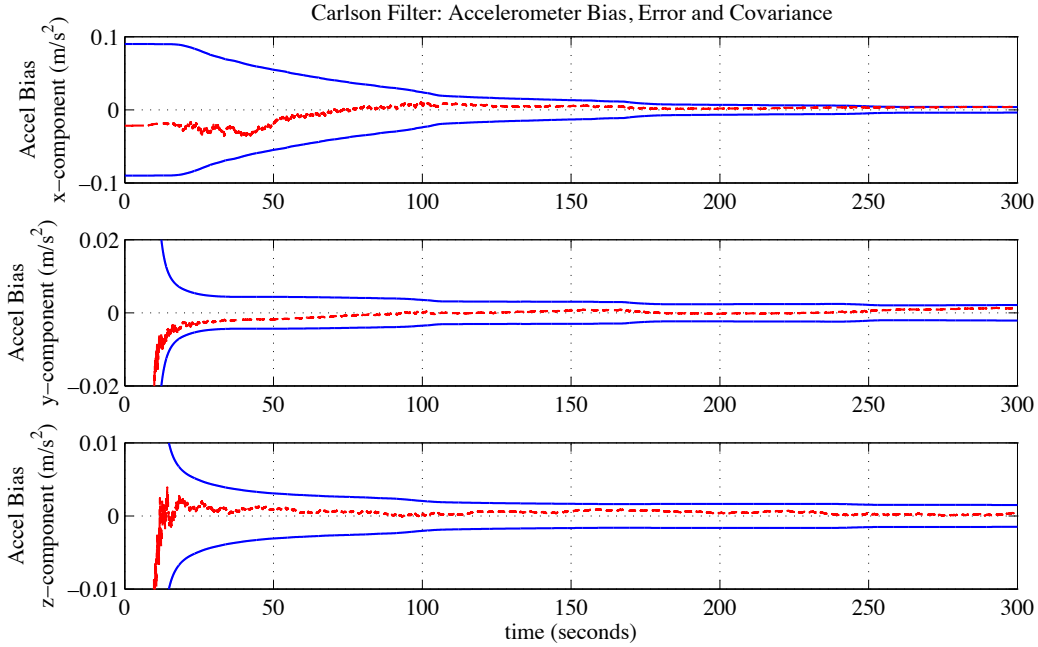


Fig. B.13: Poor Sensor Suite: Accelerometer bias error and covariance (3σ).

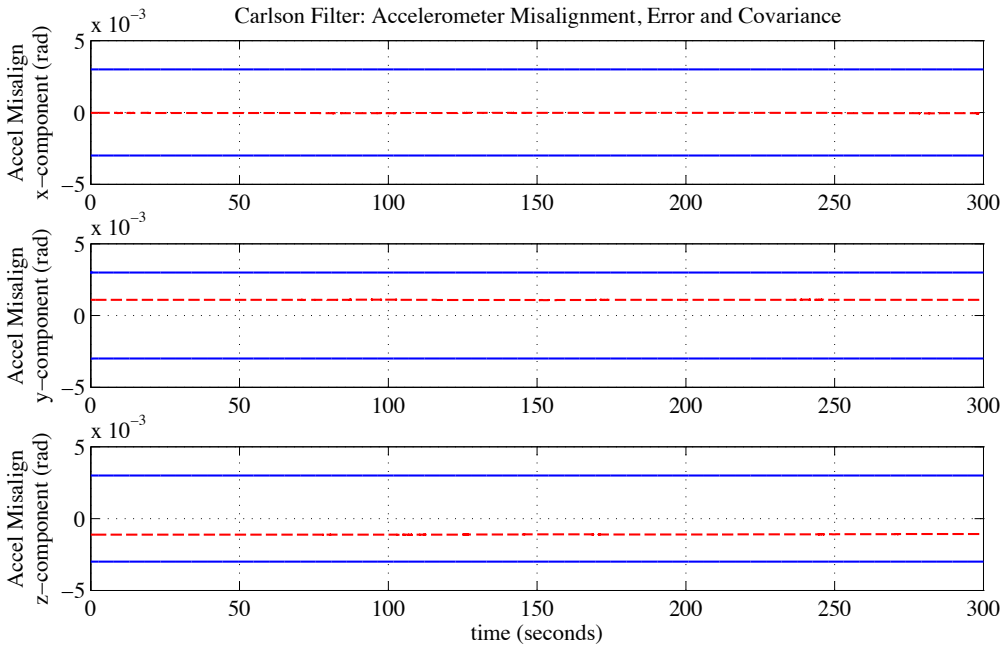


Fig. B.14: Poor Sensor Suite: Accelerometer misalignment error and covariance (3σ).

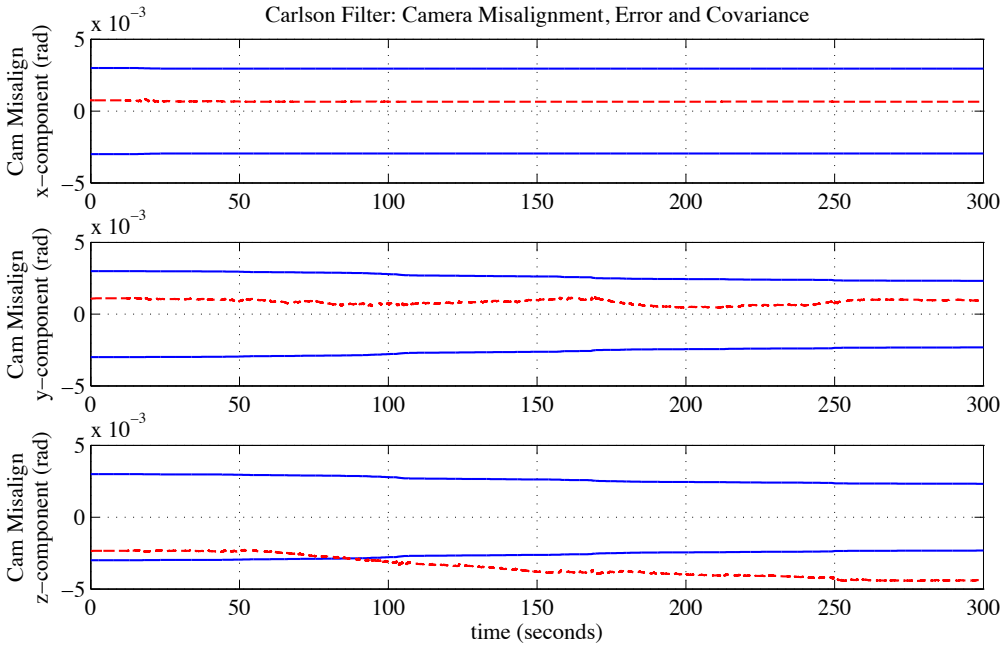


Fig. B.15: Poor Sensor Suite: LOS camera misalignment error and covariance (3σ).

Appendix C

Closed-Loop Minimum ΔV Trade Study

The true error and covariance (3σ) for position and velocity is shown in figures C.1 through C.30. The accelerometer bias and misalignment and the camera misalignment for the 1 mm/s, 10 mm/s, and 50 mm/s minimum ΔV cases are discused in Section 8.2.

C.1 1 mm/s minimum ΔV Results

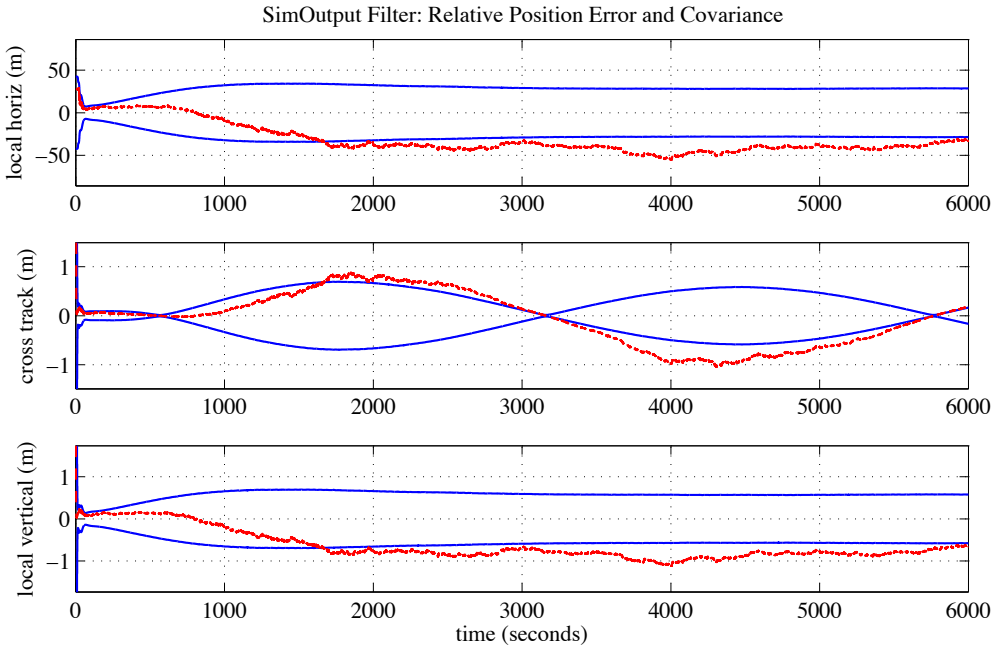


Fig. C.1: 1 mm/s ΔV Perfect nav: Relative position error and covariance (3σ).

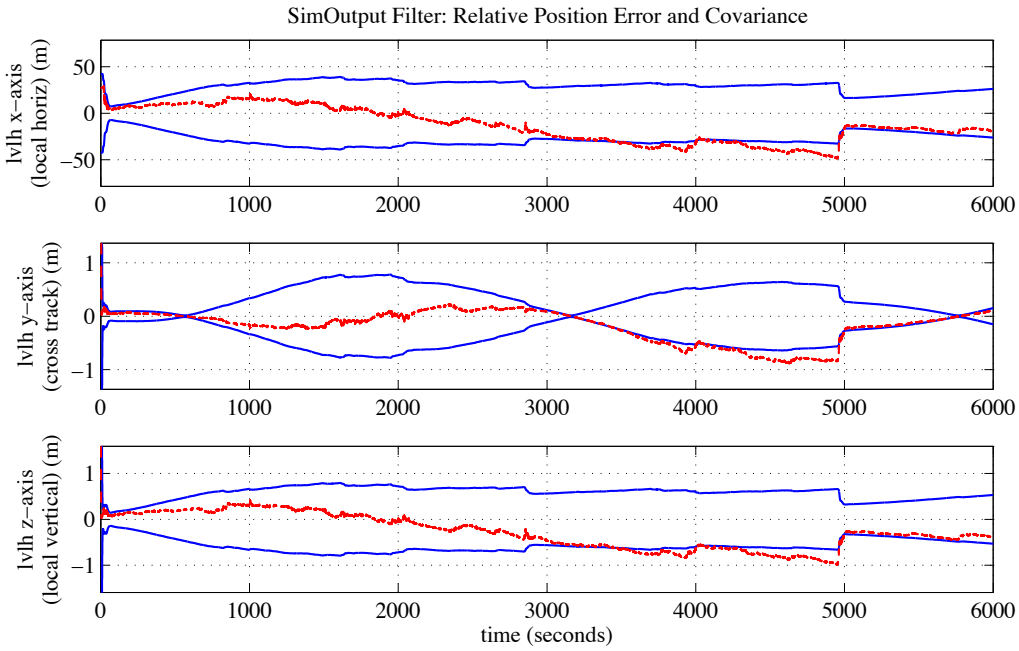


Fig. C.2: 1 mm/s ΔV Estimated nav: Relative position error and covariance (3σ).

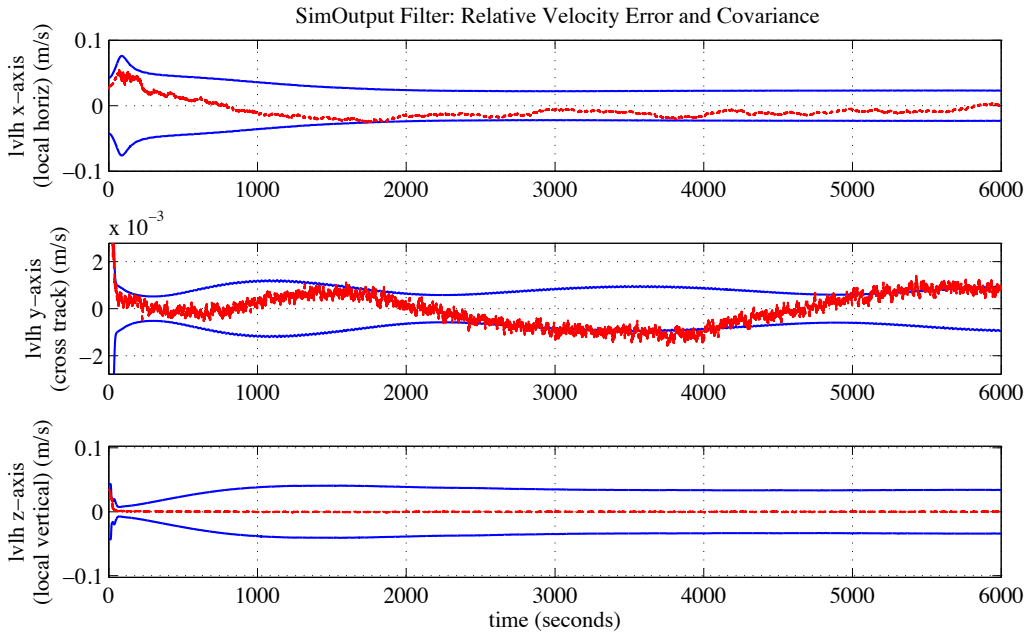


Fig. C.3: 1 mm/s ΔV Perfect nav: Relative velocity error and covariance (3σ).

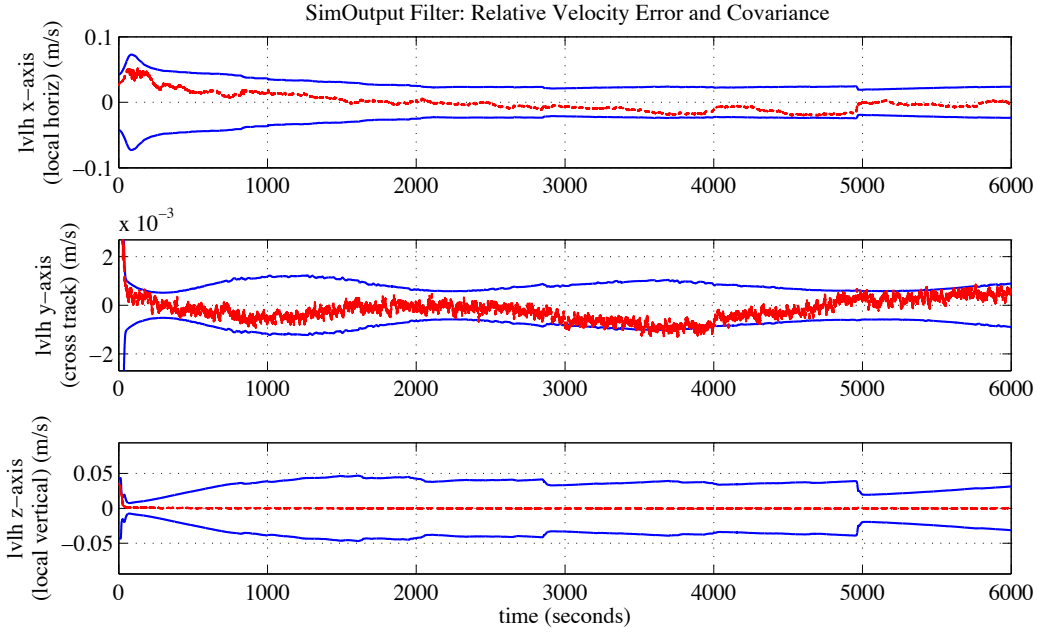


Fig. C.4: 1 mm/s ΔV Estimated nav: Relative velocity error and covariance (3σ).

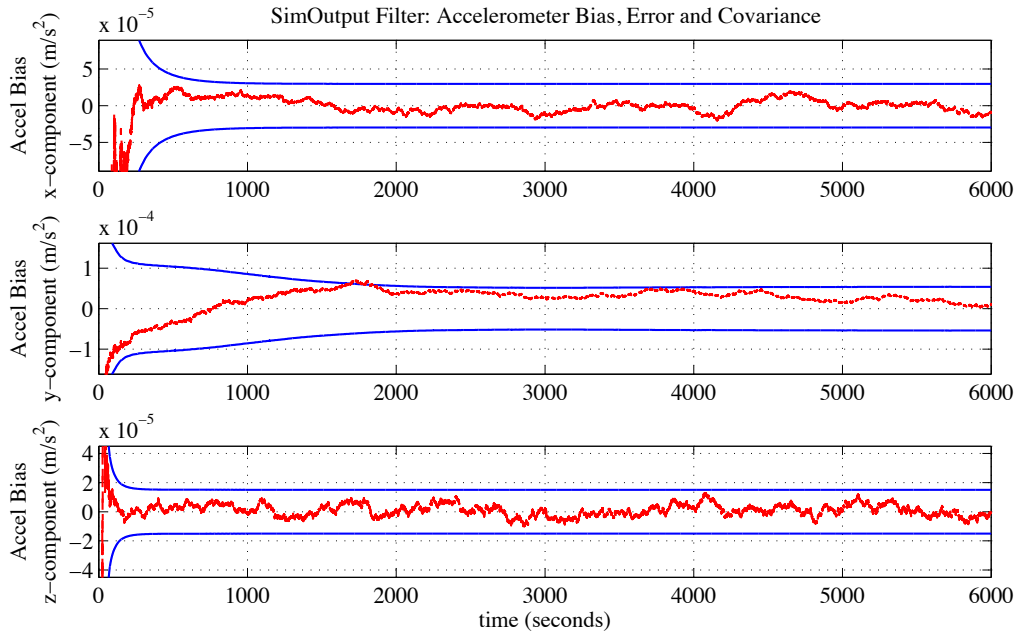


Fig. C.5: 1 mm/s ΔV Perfect nav: Accel bias error and covariance (3σ).

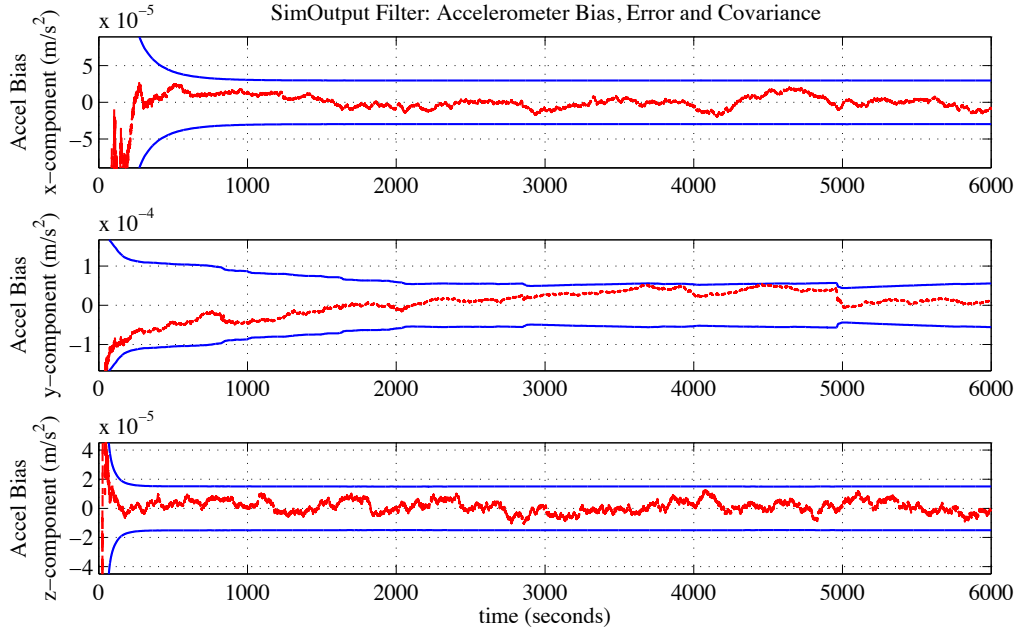


Fig. C.6: 1 mm/s ΔV Estimated nav: Accel bias error and covariance (3σ).

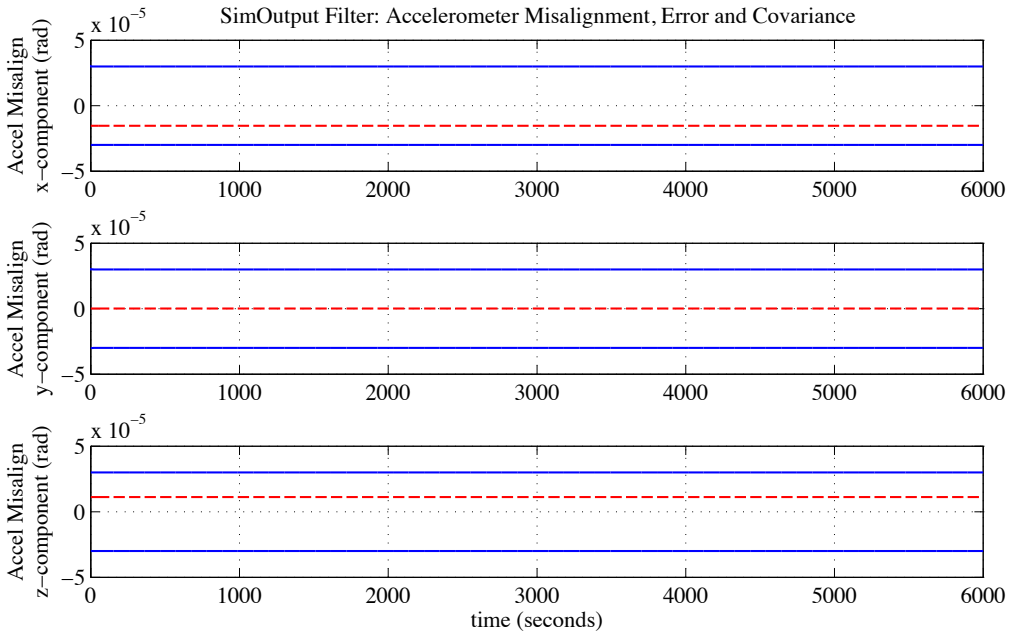


Fig. C.7: 1 mm/s ΔV Perfect nav: Accel misalignment error and covariance (3σ).

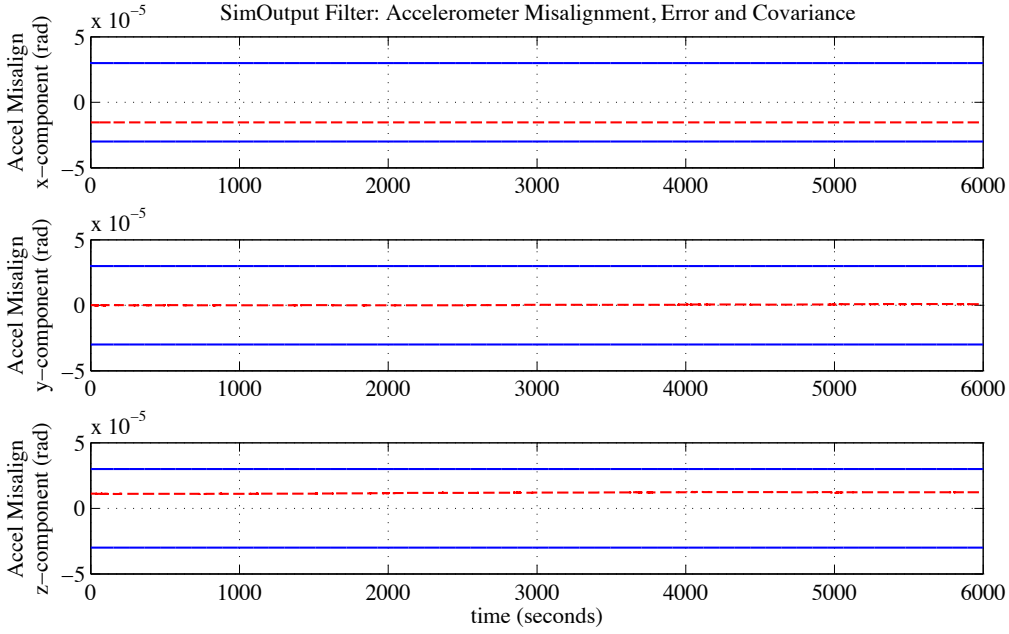


Fig. C.8: 1 mm/s ΔV Estimated nav: Accel misalignment error and covariance (3σ).

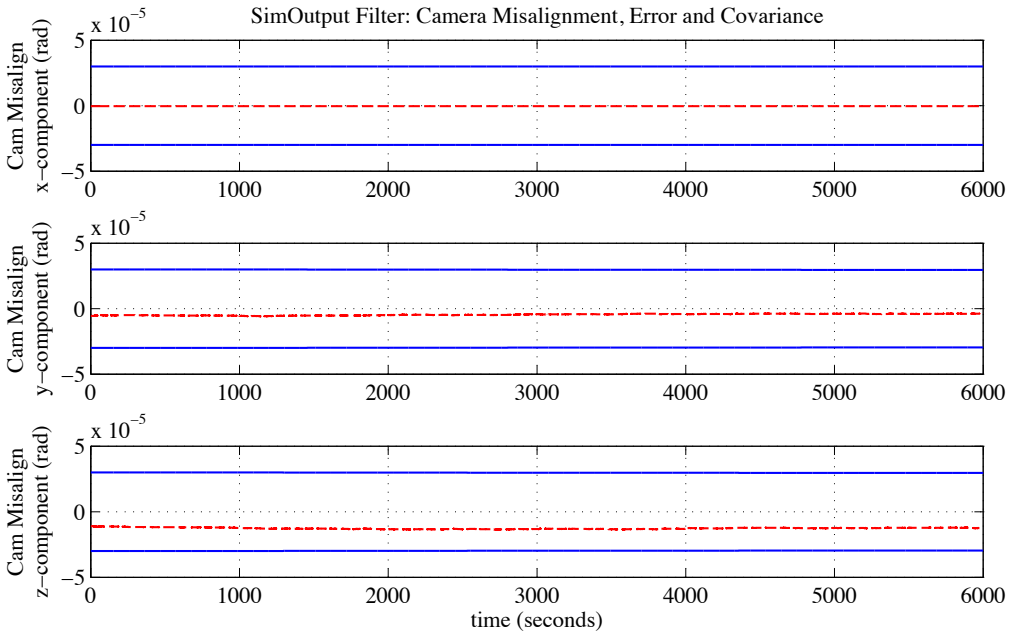


Fig. C.9: 1 mm/s ΔV Perfect nav: Accel misalignment error and covariance (3σ).

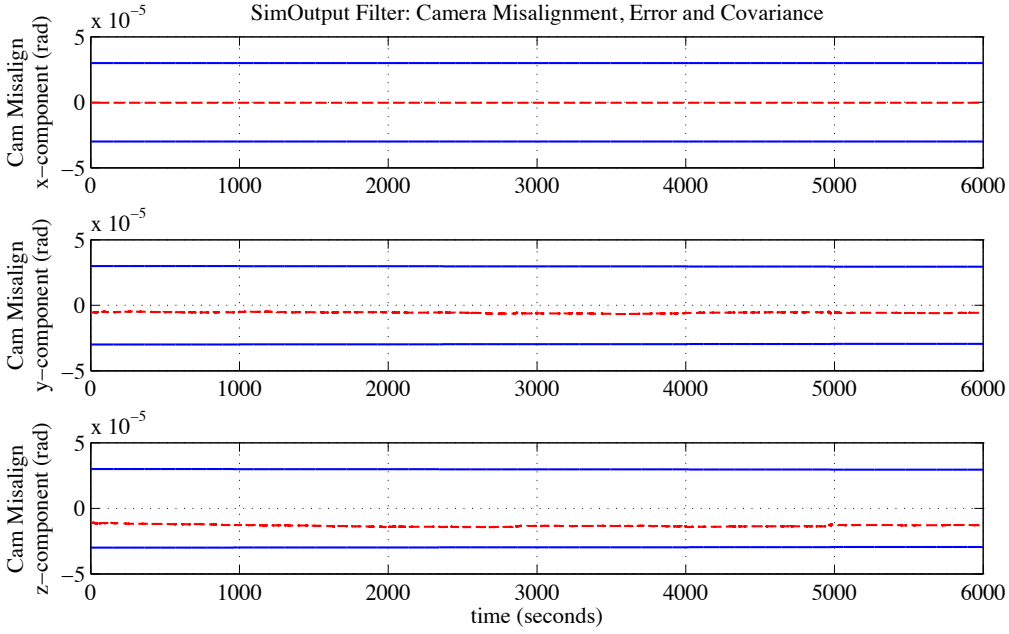


Fig. C.10: 1 mm/s ΔV Estimated nav: Accel misalignment error and covariance (3σ).

C.2 10 mm/s minimum ΔV Results

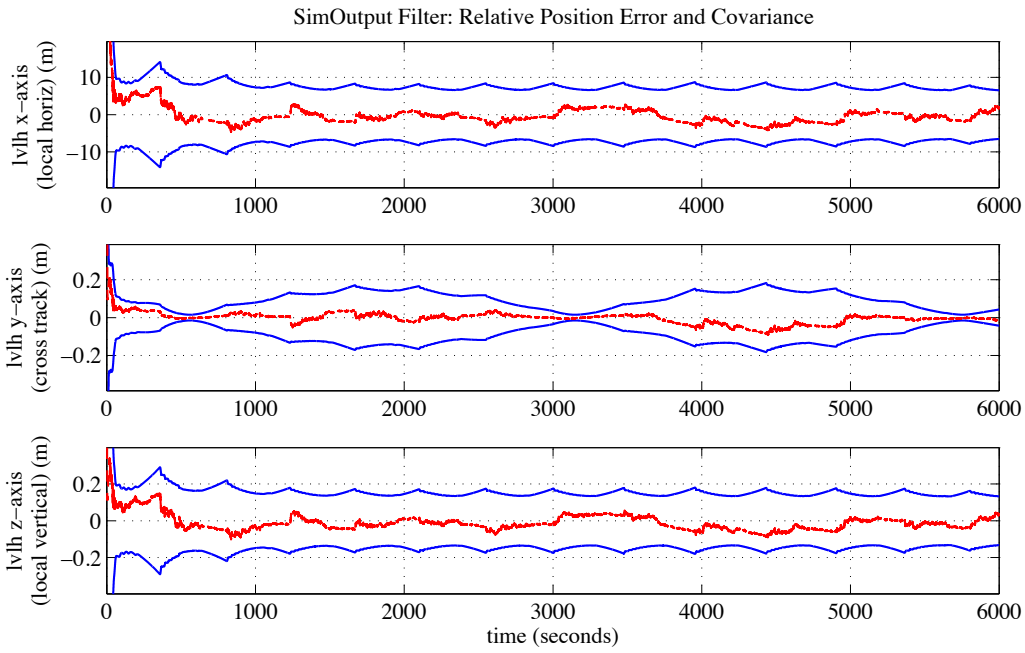


Fig. C.11: 10 mm/s ΔV Perfect nav: Relative position error and covariance (3σ).

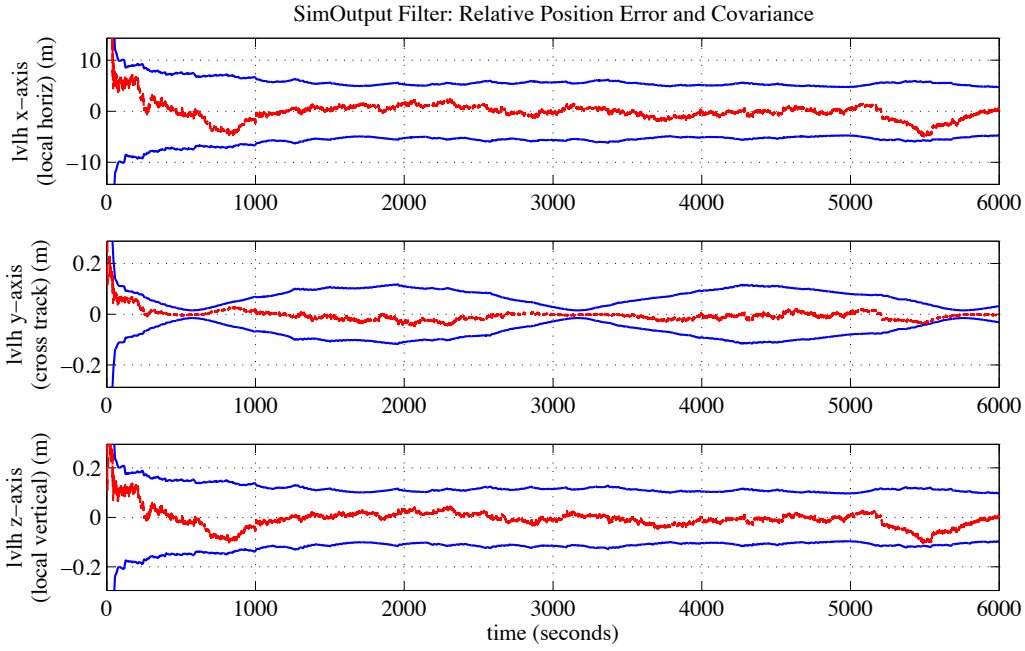


Fig. C.12: 10 mm/s ΔV Estimated nav: Relative position error and covariance (3σ).

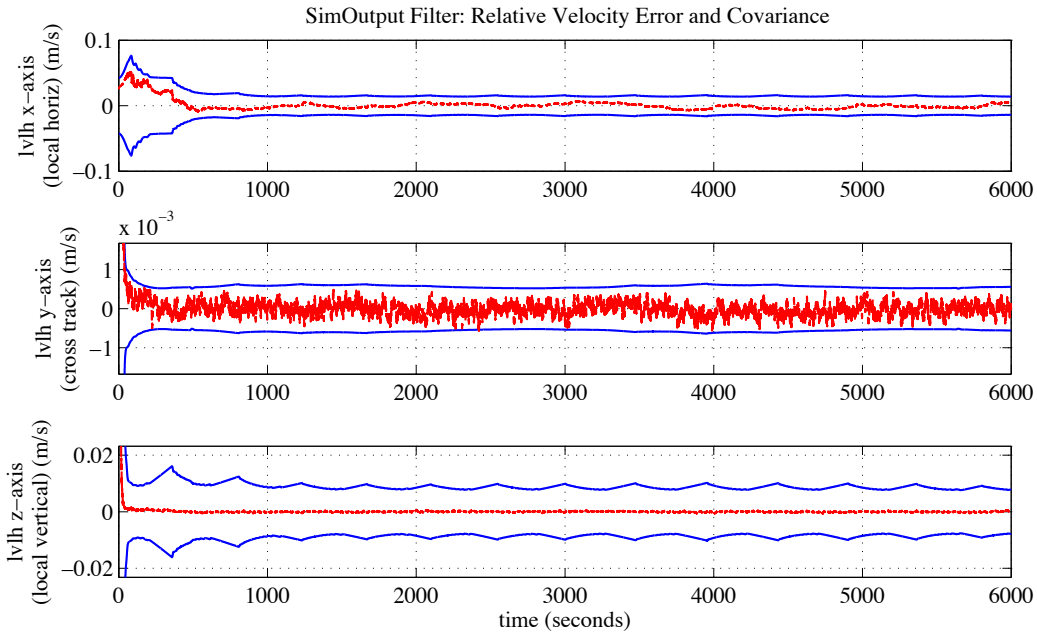


Fig. C.13: 10 mm/s ΔV Perfect nav: Relative velocity error and covariance (3σ).

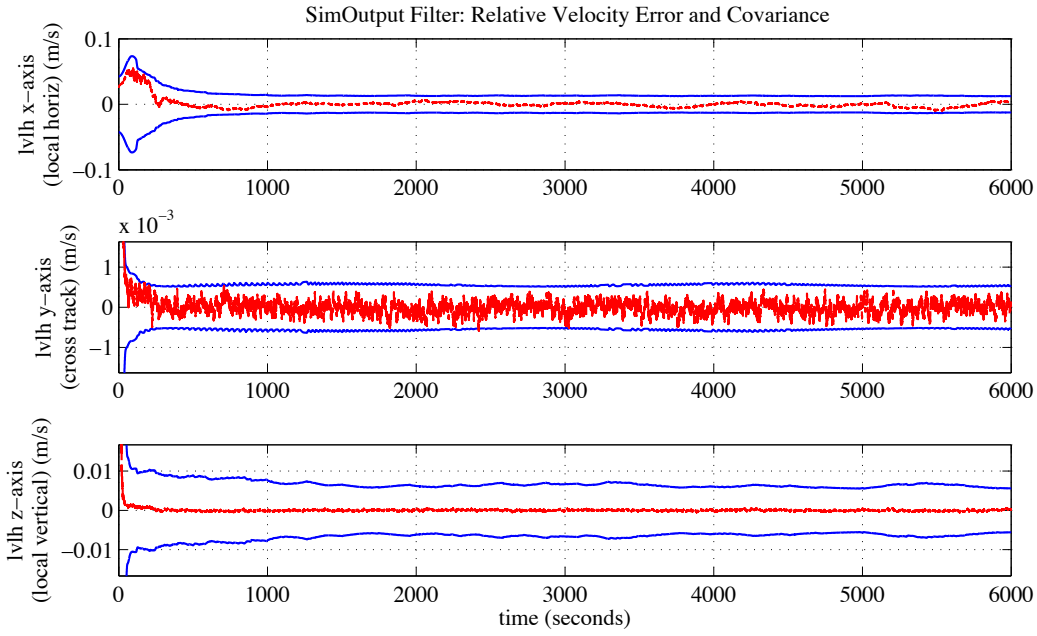


Fig. C.14: 10 mm/s ΔV Estimated nav: Relative velocity error and covariance (3σ).

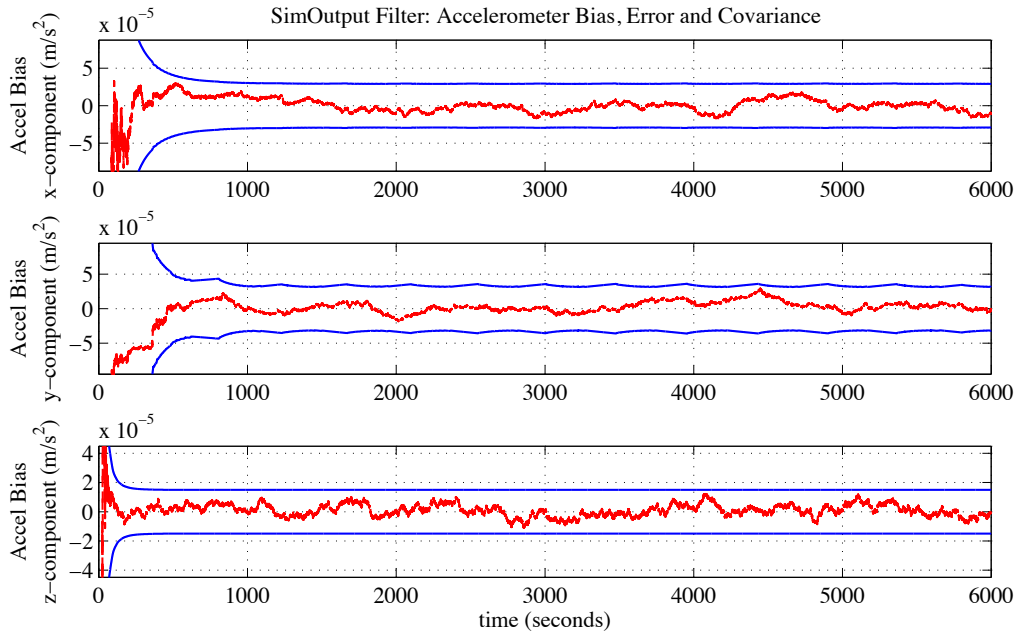


Fig. C.15: 10 mm/s ΔV Perfect nav: Accel bias error and covariance (3σ).

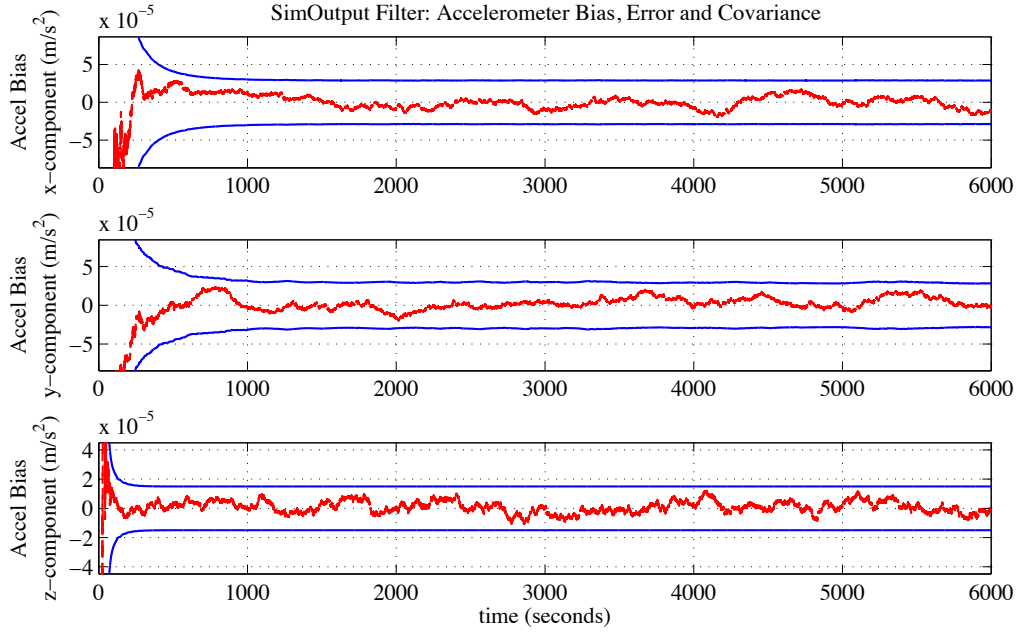


Fig. C.16: 10 mm/s ΔV Estimated nav: Accel bias error and covariance (3σ).

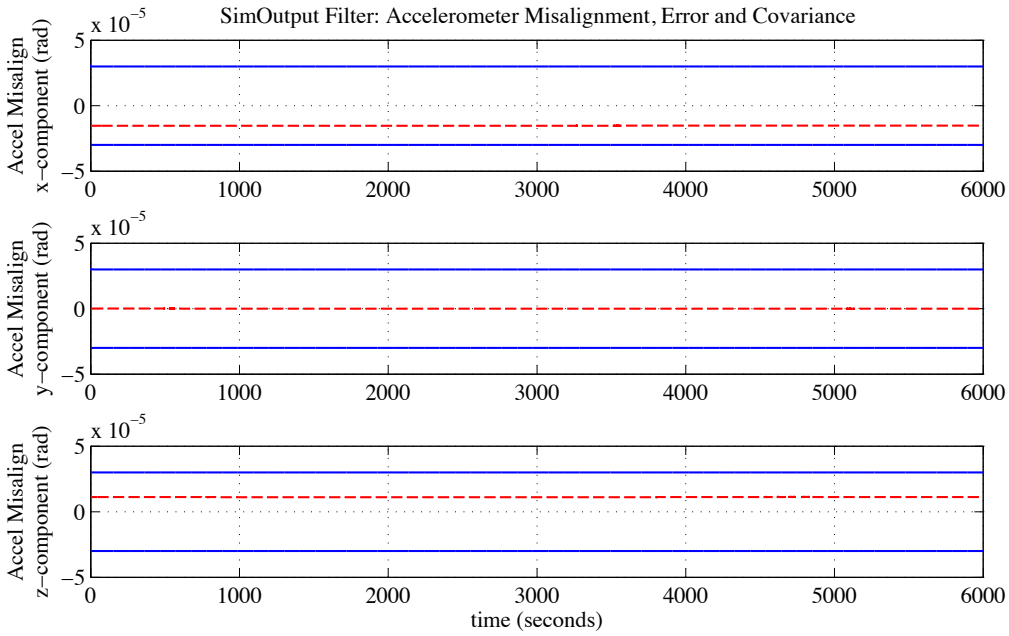


Fig. C.17: 10 mm/s ΔV Perfect nav: Accel misalignment error and covariance (3σ).

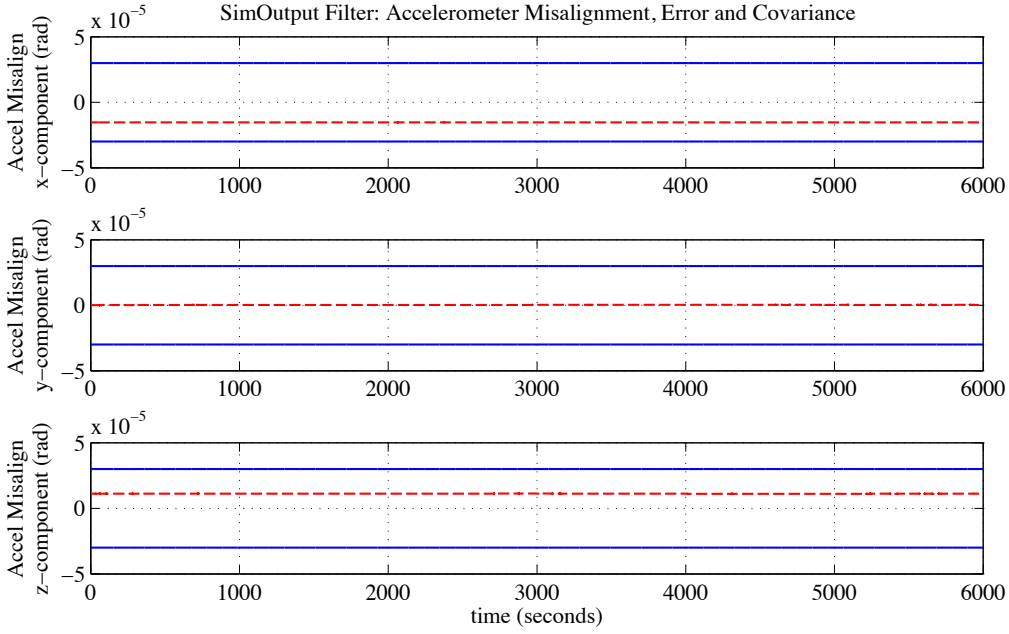


Fig. C.18: 10 mm/s ΔV Estimated nav: Accel misalignment error and covariance (3σ).

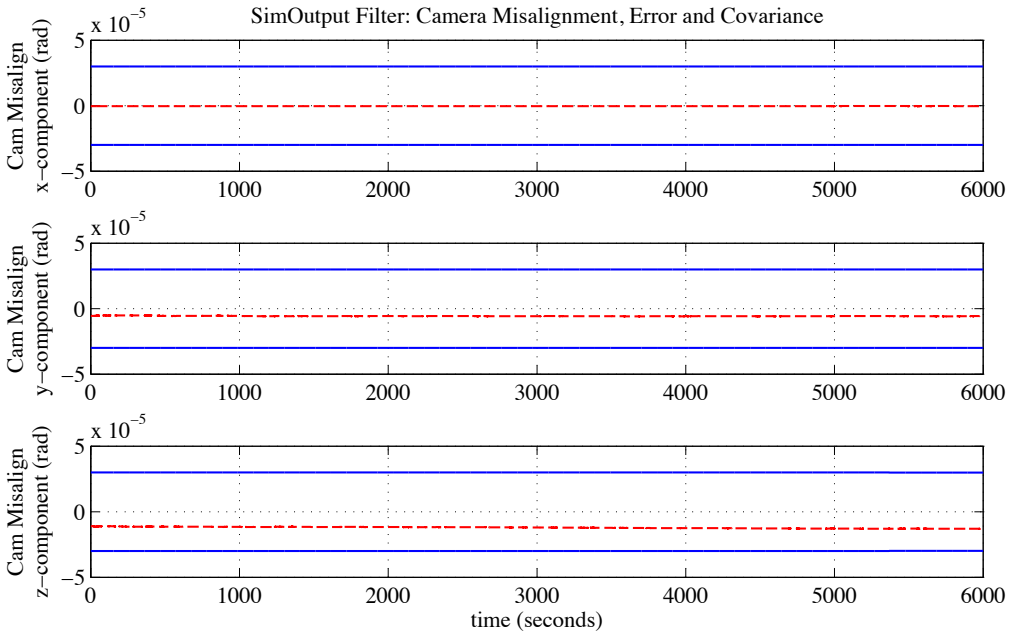


Fig. C.19: 10 mm/s ΔV Perfect nav: Accel misalignment error and covariance (3σ).

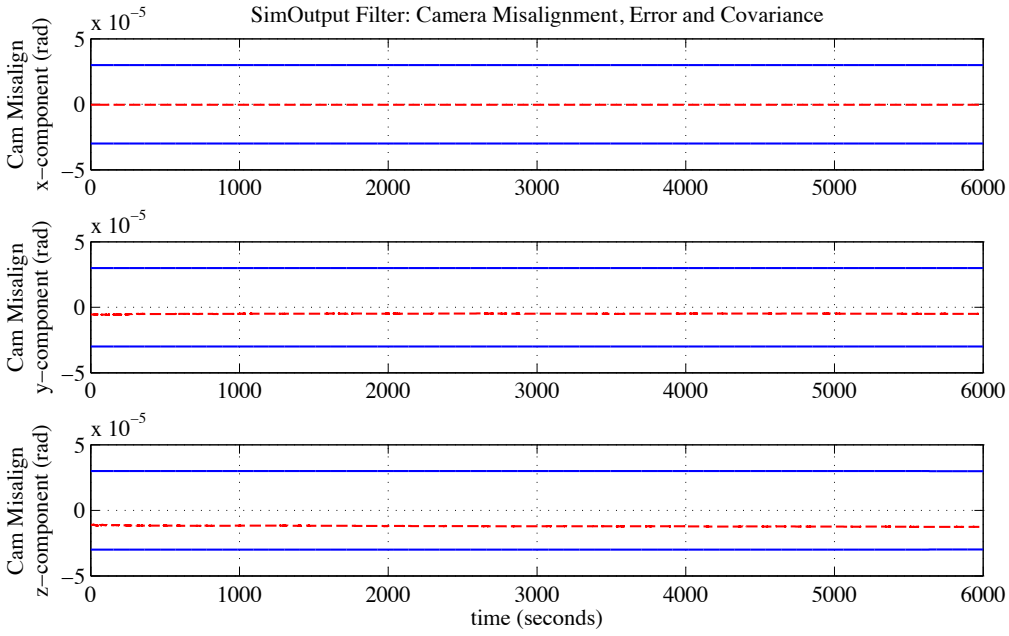


Fig. C.20: 10 mm/s ΔV Estimated nav: Accel misalignment error and covariance (3σ).

C.3 50 mm/s minimum ΔV Results

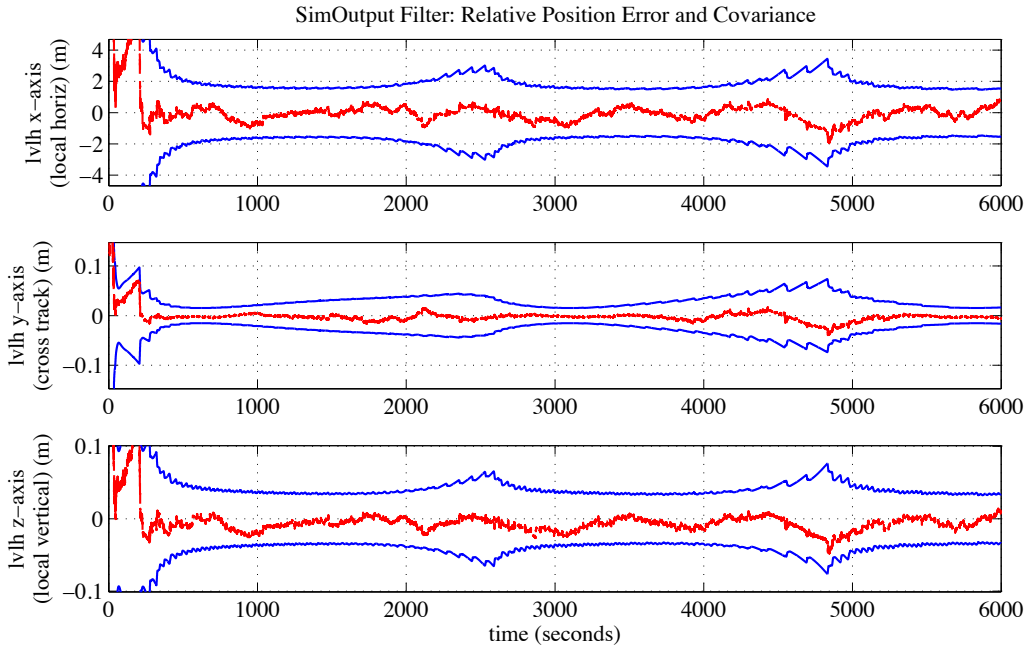


Fig. C.21: 50 mm/s ΔV Perfect nav: Relative position error and covariance (3σ).

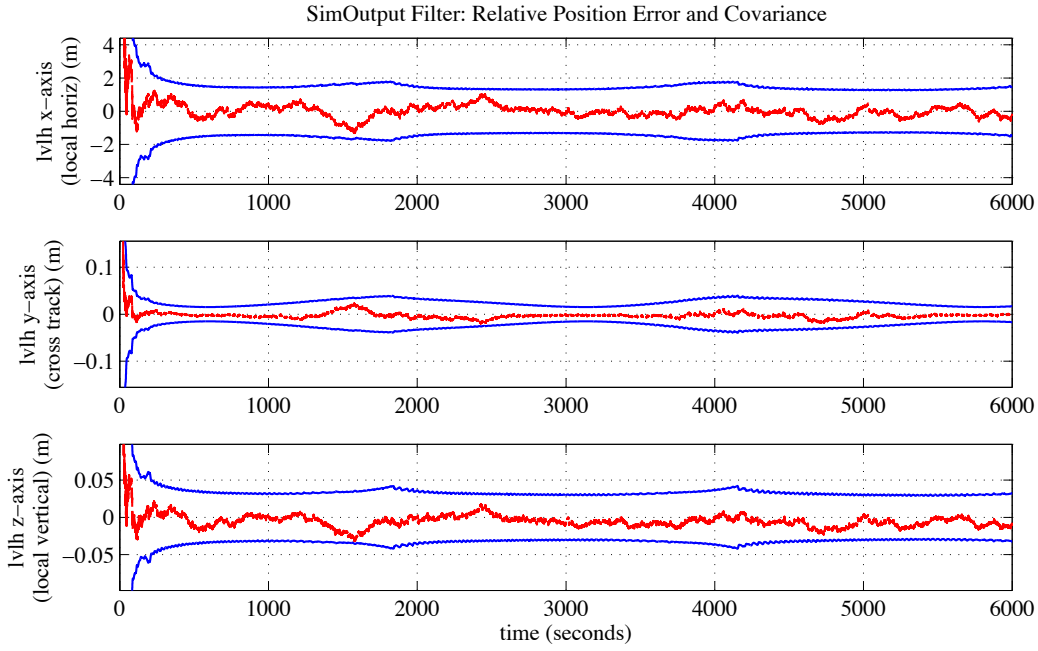


Fig. C.22: 50 mm/s ΔV Estimated nav: Relative position error and covariance (3σ).

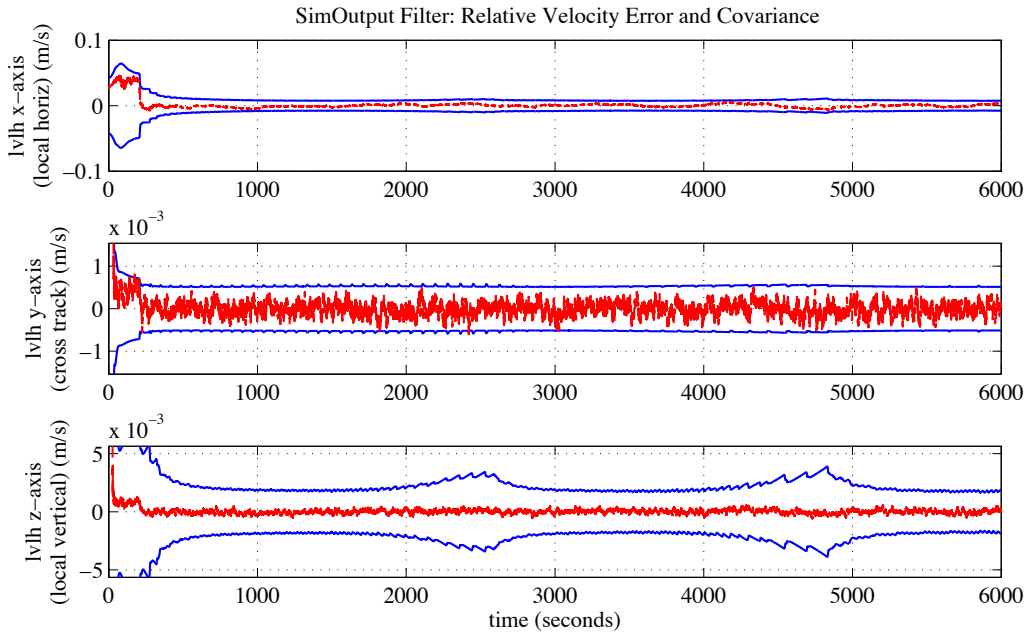


Fig. C.23: 50 mm/s ΔV Perfect nav: Relative velocity error and covariance (3σ).

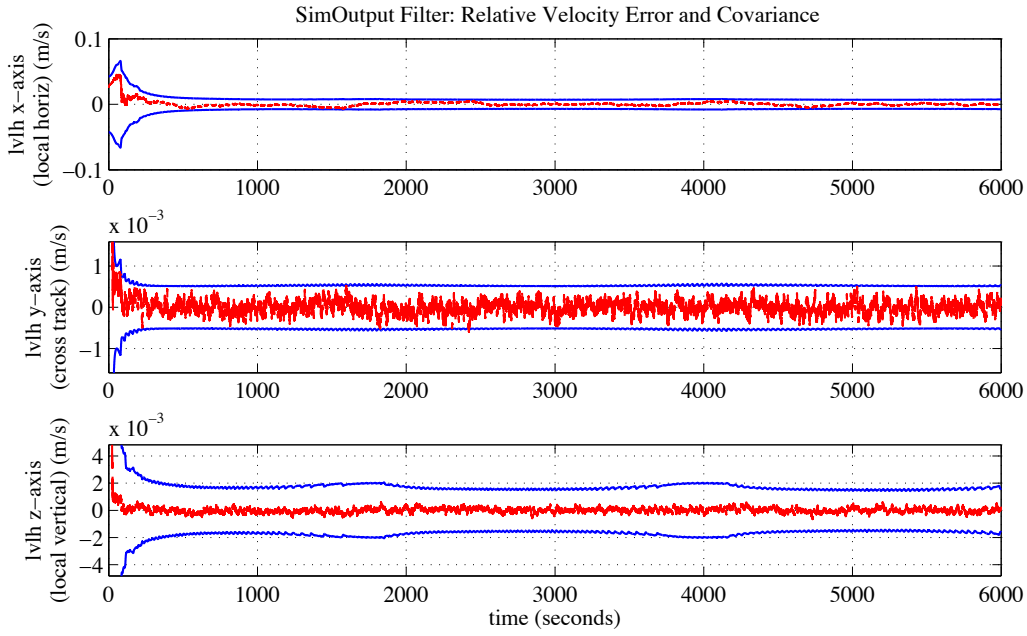


Fig. C.24: 50 mm/s ΔV Estimated nav: Relative velocity error and covariance (3σ).

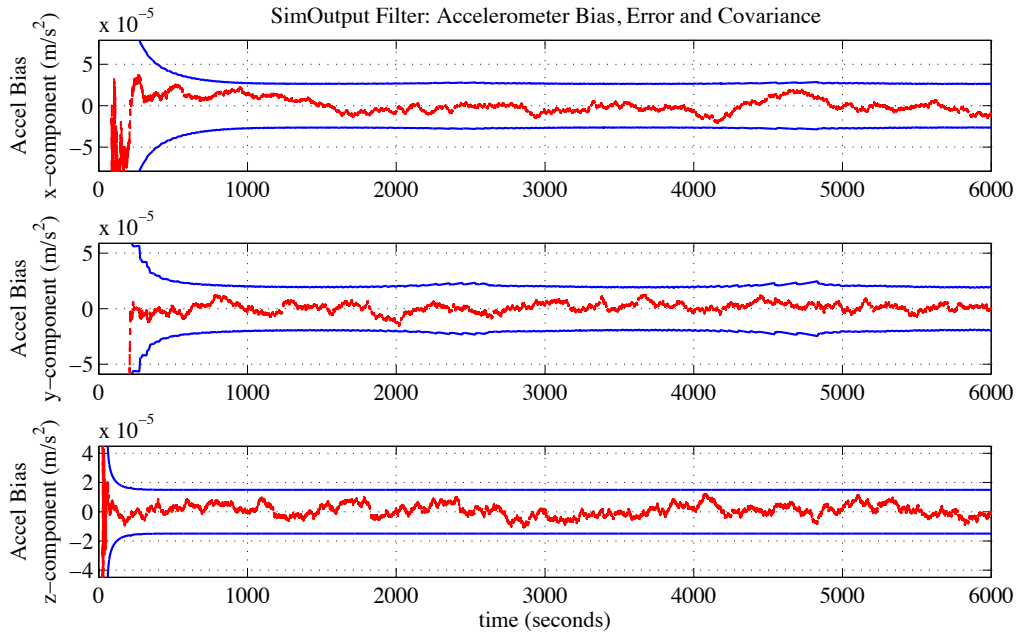


Fig. C.25: 50 mm/s ΔV Perfect nav: Accel bias error and covariance (3σ).

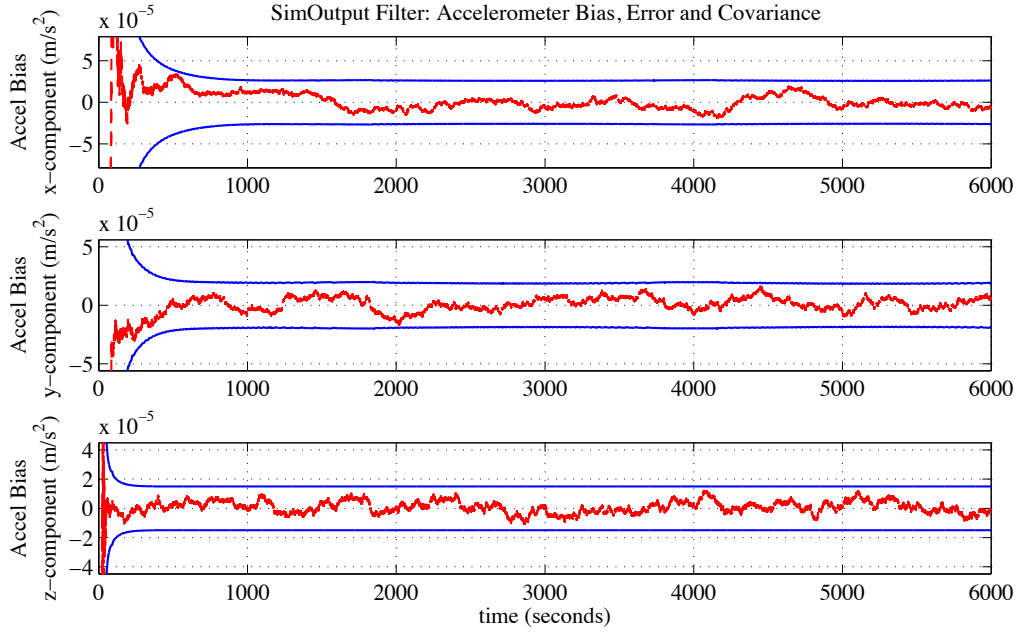


Fig. C.26: 50 mm/s ΔV Estimated nav: Accel bias error and covariance (3σ).

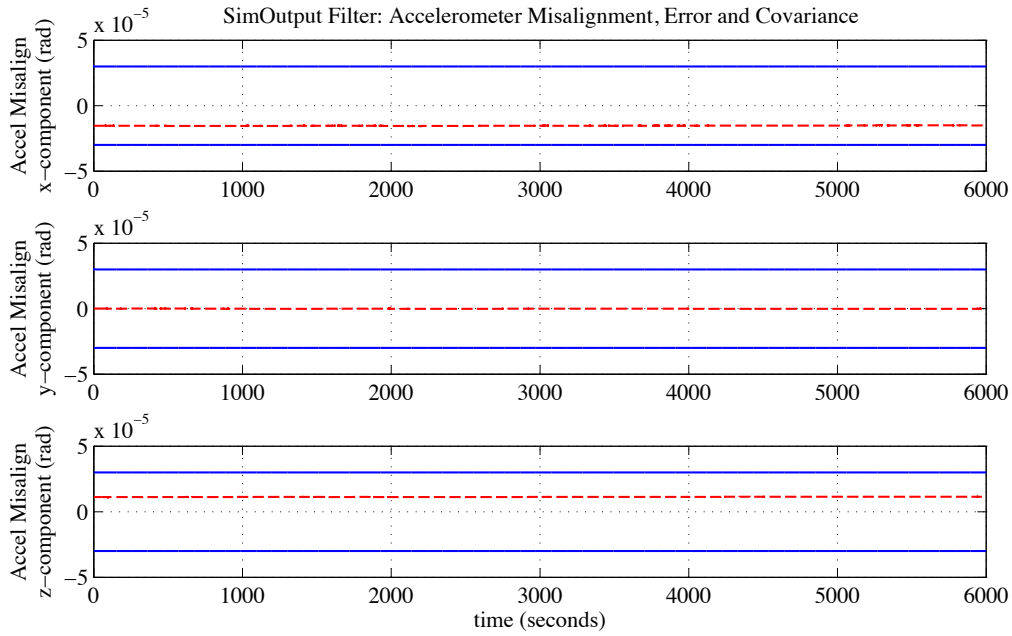


Fig. C.27: 50 mm/s ΔV Perfect nav: Accel misalignment error and covariance (3σ).

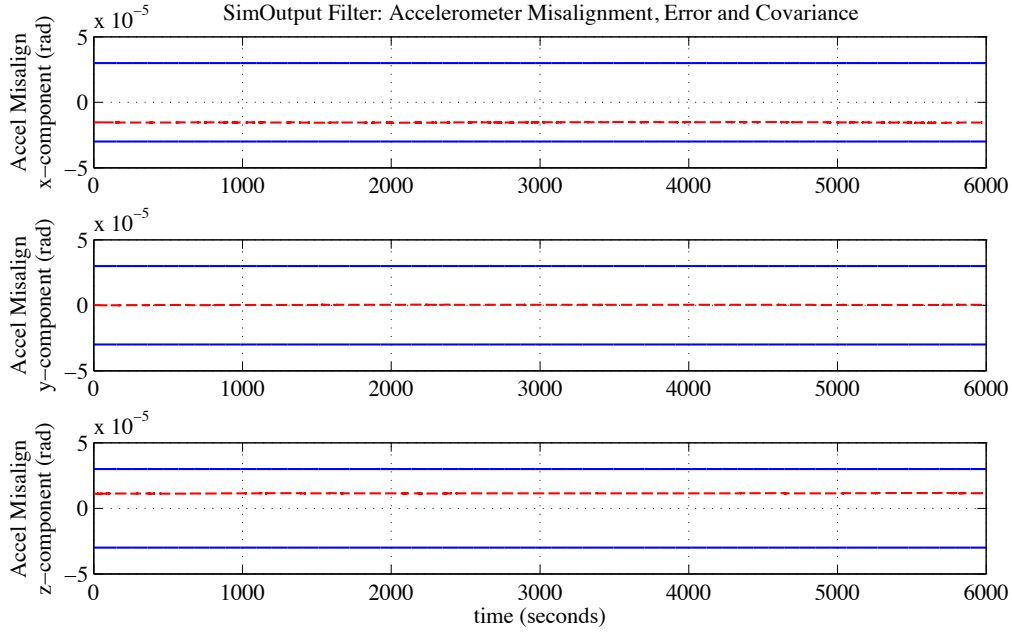


Fig. C.28: 50 mm/s ΔV Estimated nav: Accel misalignment error and covariance (3σ).

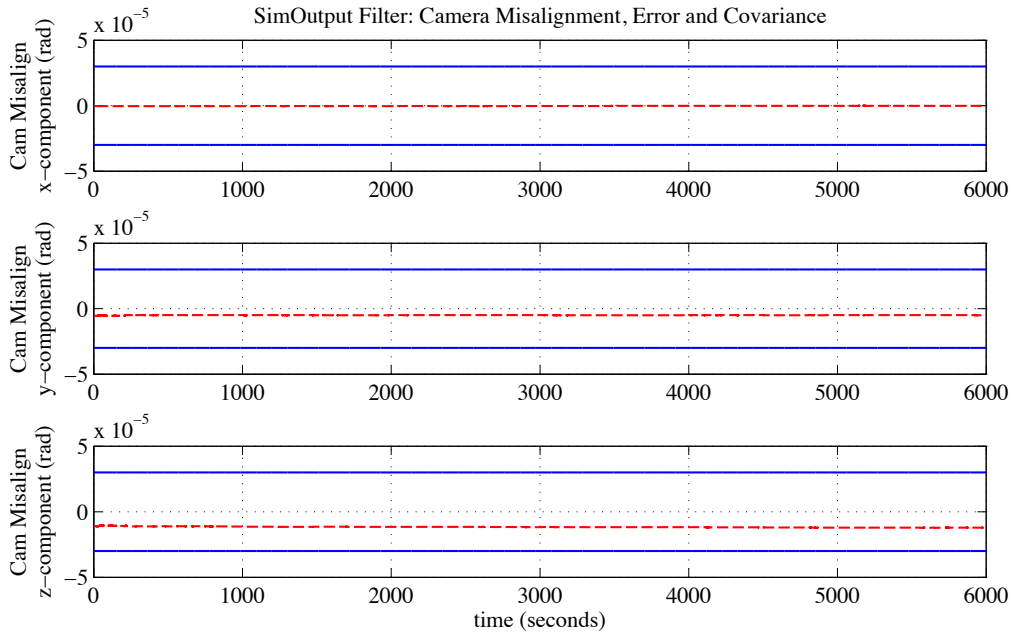


Fig. C.29: 50 mm/s ΔV Perfect nav: Accel misalignment error and covariance (3σ).

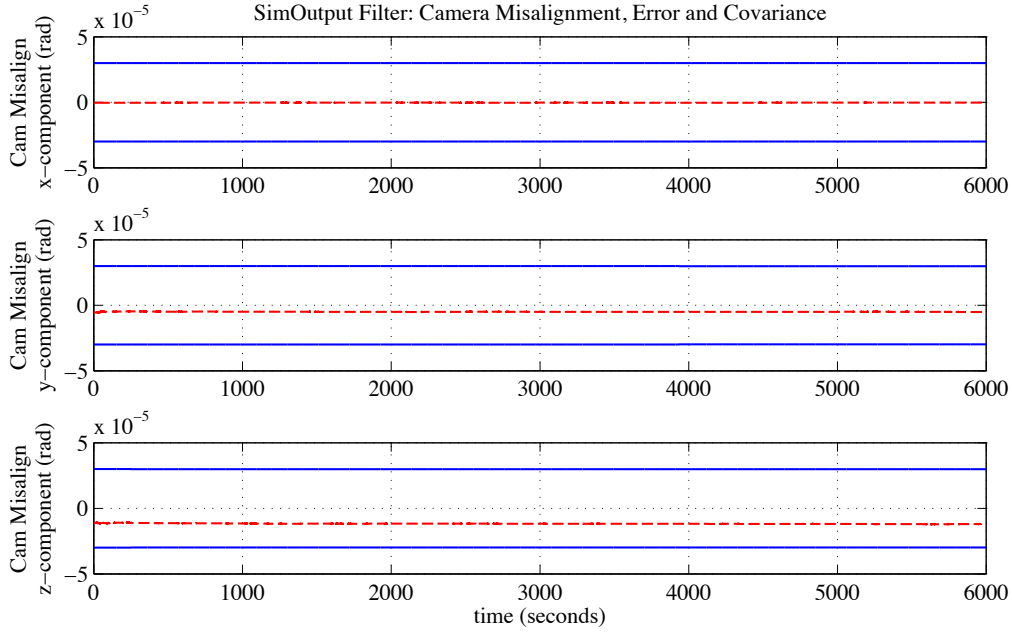


Fig. C.30: 50 mm/s ΔV Estimated nav: Accel misalignment error and covariance (3σ).

Appendix D

Toy Relative Kalman Filter Codes

This appendix includes the Simulink implementations of the Linear conventional, Potter, UD factorization filters in figures D.1 through D.3. The m-code setup file with filter models and initial conditions is found in Section D.2.

D.1 Linear Kalman Filters in Simulink

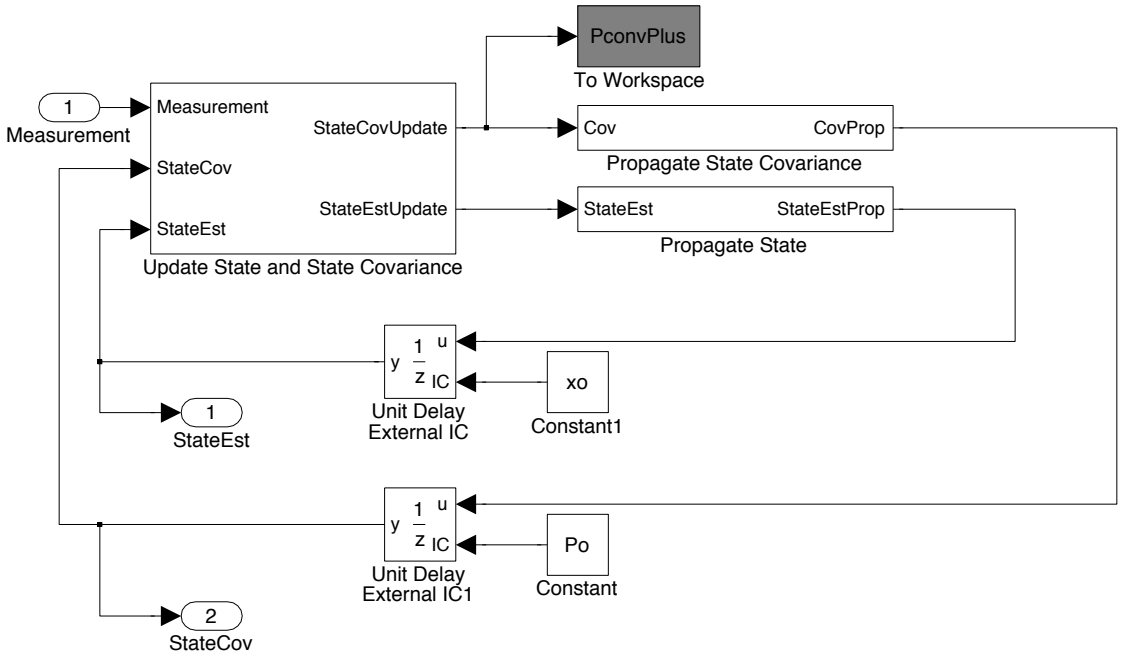


Fig. D.1: Conventional linear Kalman filter implementation.

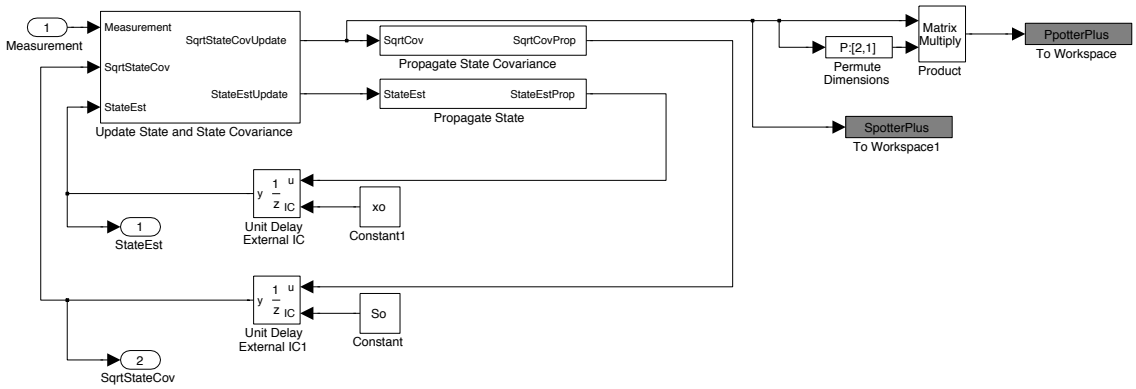


Fig. D.2: Potter linear Kalman filter implementation.

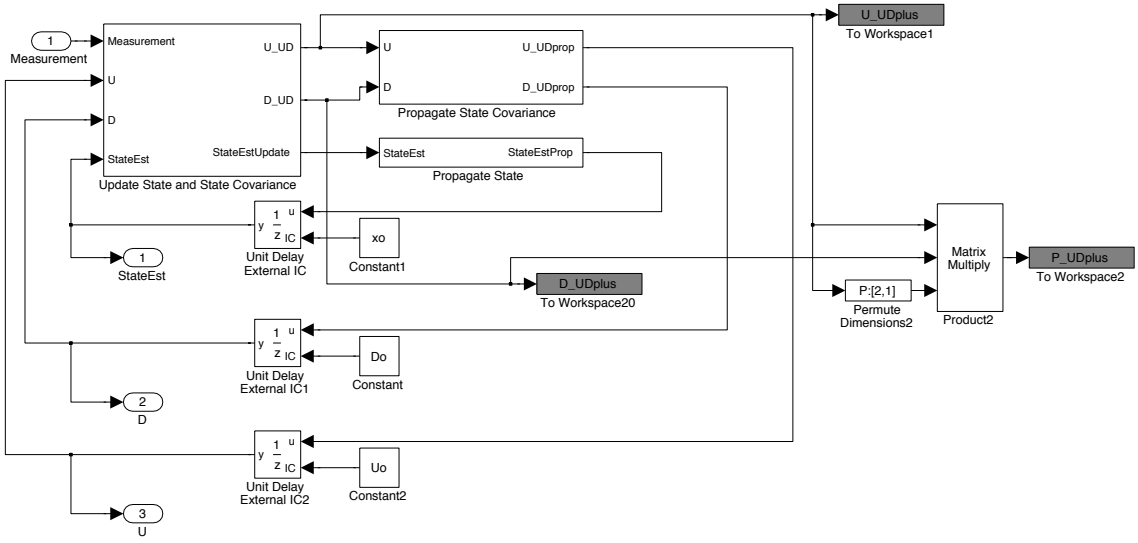


Fig. D.3: UD factorization linear Kalman filter implementation.

D.2 Toy Relative Case Setup

3/9/10 9:35 PM /Users/jasonschmidt/Documents.../Run_RelativeCase.m 1 of 2

```
%% Simple Relative-State Filter
clear all; close all; clc;
savePDF=0;
saveFIG=0;

%% GENERATE DATA FOR WORKSPACE

%% Initial State and State Covariance
%first rotating shaft
r1o = 0; %deg
v1o = 0; %deg/sec
%second rotating shaft
r2o = 1; %deg
v2o = 0; %deg/sec

xo = [r1o; v1o; r2o; v2o]

%Initial Covariance
Po = diag([1 .01 1 .01]);%*1e-10
So = chol(Po,'lower'); %Such that S0*S0' = P0
[Uo, Do]=udu(Po);      %Such that U0*D0*U0' = P0

%% Measurement Info
%measurement sensitivity matrix
H=[-1 0 1 0]
%Measurement noise
v = sqrt(eps)*1e2;
%Measurement covariance matrix
R = diag(v.*v)

%% State Transition Matrix
%integration stepsize
stepsize = 1;
%Linearized Dynamics
F = [0 1; 0 0];
F = [F zeros(2); zeros(2) F]
%state transition matrix
Phi = eye(4)+ F*stepsize

%% Process Noise
%Process Noise
w = [eps; eps; eps; eps]*1e4;%.*eps;%
%noise strength
Q = diag(w.*w)
%discrete process noise
Qd = Q*stepsize;
Wd = chol(Qd,'lower'); %such that Wd*Wd' = Qd
%Noise input matrix
B = eye(4)

%% Matrix to convert absolute covariance to relative covariance
REL = [-eye(2) eye(2)];

%% RUN SIMULATION

sim('RelativeCase.mdl',[0 5000])
```

Appendix E

Extended Kalman Filter Codes

The Simulink implementations of the Non-linear conventional and Joseph, Potter and Carlson, and UD factorization filters are shown in figures E.1 through E.3. The m-code implementation of the state dynamic model and integration code is in Section E.2. The "Extended" protions of the codes are covered in sections E.3 and E.4. The Potter, Carlson, and UD factorization "Update" codes are covered in sections E.5 through E.7. The Potter/Carlson and UD covariance propagation codes are shown in sections E.8 and E.9.

E.1 Extended Kalman Filters in Simulink

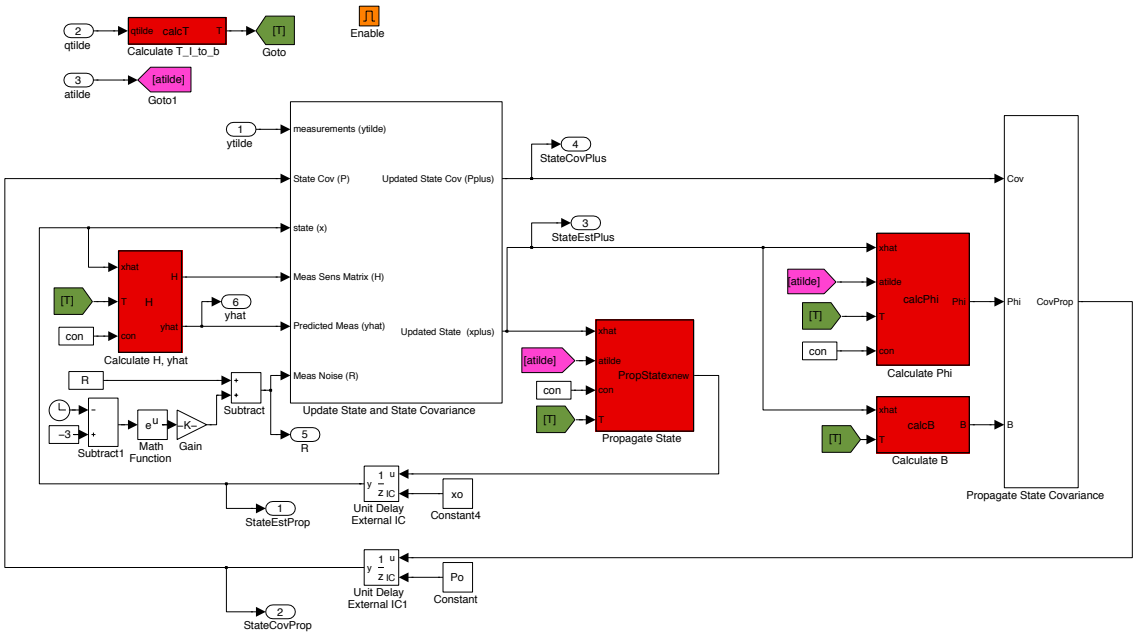


Fig. E.1: Conventional and Joseph extended Kalman filter implementation.

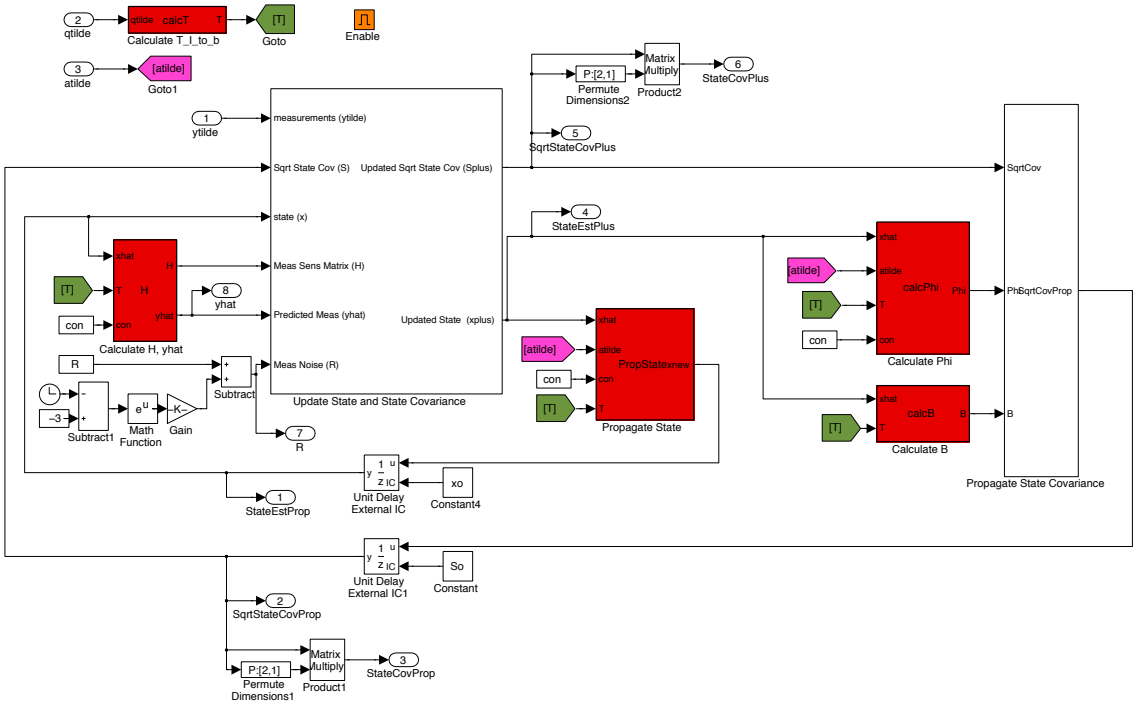


Fig. E.2: Potter and Carlson extended Kalman filter implementation.

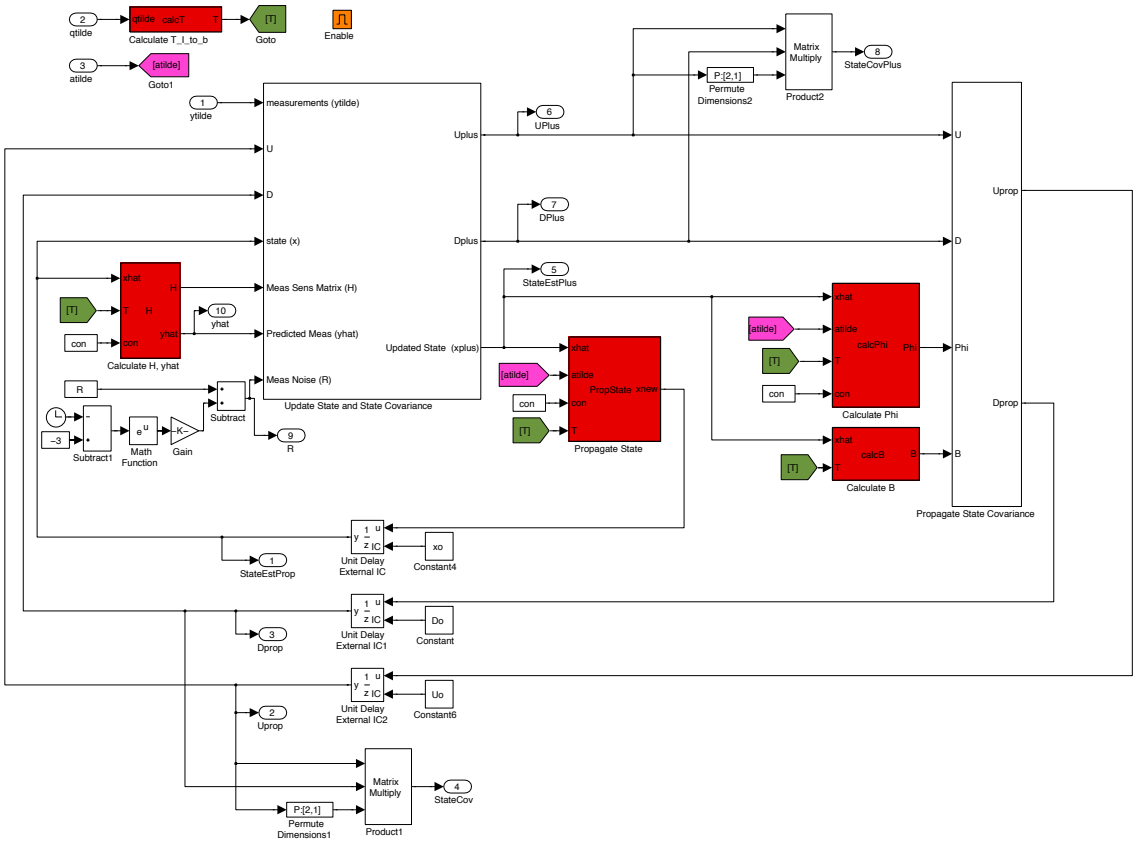


Fig. E.3: UD factorization extended Kalman filter implementation.

E.2 State Dynamics and Integration

2/16/10 9:41 PM Block: SatelliteSIM/Satellite/F.../Propagate State 1 of 2

```
function xnew =PropState(xhat,atilde,con,T)                                %↵
<<====  PropState
dt  = con(10);

y=xhat;
x=0;
dydx=getxdot(y,x,atilde,con,T);

h=dt;
hh=h/2;
h6=h/6;
xh=x+hh;

yt=y+hh*dydx;

dyt=getxdot(yt,xh,atilde,con,T);
yt=y+hh*dyt;

dym=getxdot(yt,xh,atilde,con,T);
yt=y+h*dym;
dym=dyt+dym;

dyt=getxdot(yt,x+h,atilde,con,T);
yout=y+h6*(dydx+dyt+2*dym);
xnew=yout;


end

function xdot=getxdot(xhat,t,atilde,con,T)
%% Parse Data

%xhat
rc  = xhat(1:3);
vc  = xhat(4:6);
rt  = xhat(7:9);
vt  = xhat(10:12);
beta= xhat(13:15);
eacc= xhat(16:18);
ecam= xhat(19:21);


%con
mu      = con(1);
J2      = con(2);
Re      = con(3);
nbar    = con(4:6);
tauaccB = con(7);
tauacc  = con(8);
taucam  = con(9);



%% ihat_r
ihat_rc = rc/norm(rc);
ihat_rt = rt/norm(rt);

%% f(xhat) = xdot
xdot = [vc;
         g(rc,mu,J2,Re,nbar,ihat_rc)+T'*(eye(3)+crs(eacc))*(atilde-beta);
         vt;
         g(rt,mu,J2,Re,nbar,ihat_rt);
         -beta/tauaccB;
         -eacc/tauacc;
         -ecam/taucam];



end

%_____________________________________
%         SUBFUNCTIONS
%_____________________________________
function gr = g(r,mu,J2,Re,nbar,ihat_r)↵
%<<====  g

gr=-mu*(r/norm(r)^3)-mu*(J2*Re^2/(2*norm(r)^5))*...
    (6*dot(r,nbar)*nbar+3*r-15*dot(ihat_r,nbar)^2*r);
end

function crsform=crs(vect)
%calculate the cross product form of a vector

crsform = [0        -vect(3)    vect(2)
           vect(3)  0           -vect(1)
           -vect(2) vect(1)     0];
end
```

E.3 Measurement sensitivity matrix (H) and measurement estimate (yhat)

2/16/10 9:40 PM Block: SatelliteSIM/Satellite.../Calculate H, yhat 1 of 1

```
function [H,yhat] = H(xhat,T,con)
% Calculate the measurements sensitivity matrix (H) and the predicted
% measurement (yhat).  This is a function of the states and various
% constants.

% Parse xhat
rc  = xhat(1:3);
rt  = xhat(7:9);
ecam= xhat(19:21);
% Parse con
rcam=con(11:13);


% Calculate Relative state
Rrel = (eye(3)-crs(ecam))*(T*(rt-rc)-rcam);
Rrel=Rrel/norm(Rrel);
Rx=Rrel(1);
Ry=Rrel(2);
Rz=Rrel(3);


yhat = [ Rz/Rx
         Ry/Rx];

dh_dRrel = [-Rz/Rx^2    0    1/Rx;
            -Ry/Rx^2    1/Rx 0];
dRrel_drc = -(eye(3)- crs(ecam))*T;
dRrel_drt = (eye(3)- crs(ecam))*T;
dRrel_decam=crs(T*(rt-rc)-rcam);

H = dh_dRrel*[dRrel_drc'; zeros(3); dRrel_drt'; zeros(9,3); dRrel_decam']';

end

function crsform=crs(vect)
%calculate the cross product form of a vector

crsform = [0        -vect(3)    vect(2)
           vect(3)  0           -vect(1)
           -vect(2) vect(1)     0];
end
```

E.4 Linearized State Dynamics (F) and State Transition Matrix (Phi)

2/16/10 9:45 PM Block: SatelliteSIM/Satellite/Fli.../Calculate Phi 1 of 2

```
function Phi  = calcPhi(xhat,atilde,T,con)
% calculate the transformation matrix form of q

dt      = con(10);

F = getF(xhat,atilde,con,T);
%F=sparse(F);
Phi=expm(F.*dt);
end

%_____________________________________
%         SUBFUNCTIONS
%_____________________________________
function [F] = getF(xhat,atilde,con,T)
%<<====  getF
% Linear form of f function

%% Parse Data

%xhat
rc  = xhat(1:3);

rt  = xhat(7:9);

beta= xhat(13:15);
eacc= xhat(16:18);

%con
mu      = con(1);
J2      = con(2);
Re      = con(3);
nbar    = con(4:6);
tauaccB = con(7);
tauacc  = con(8);
taucam  = con(9);


%% Define Partials
ihat_rc = rc/norm(rc);
ihat_rt = rc/norm(rt);

drdot_dvc = eye(3);
dvdot_drc = A1(rc,mu,J2,Re,ihat_rc)+A2(rc,mu,J2,Re,nbar,ihat_rc) +...
              A3(rc,mu,J2,Re,nbar,ihat_rc);
dvdot_dB = -T'*(eye(3)+crs(eacc));
dvdot_deacc = T'*(crs(beta)-crs(atilde));
drdot_dvt = eye(3);
dvdot_drt = A1(rt,mu,J2,Re,ihat_rt)+A2(rt,mu,J2,Re,nbar,ihat_rt) +...
              A3(rt,mu,J2,Re,nbar,ihat_rt);
dBdot_dB = (-1/tauaccB)*eye(3);
deaccdot_deacc = (-1/tauacc)*eye(3);
decamdot_decam = (-1/taucam)*eye(3);

%% Define F
F=NaN(21,21);
F(:,1:3) = [zeros(3); dvdot_drc; zeros(15,3)];

F(:,4:6) = [drdot_dvc; zeros(18,3)];
F(:,7:9) = [zeros(9,3); dvdot_drt; zeros(9,3)];
F(:,10:12)=[zeros(6,3); drdot_dvt; zeros(12,3)];
F(:,13:15)=[zeros(3); dvdot_dB; zeros(6,3); dBdot_dB; zeros(6,3)];
F(:,16:18)=[zeros(3); dvdot_deacc; zeros(9,3); deaccdot_deacc; zeros(3)];
F(:,19:21)=[zeros(18,3); decamdot_decam];

end

function [A1] = A1(r,mu,J2,Re,ihat_r)
%<<==  A1
    A1 = -mu/norm(r)^3*(eye(3)-3*ihat_r*ihat_r');
end

function [A2] = A2(r,mu,J2,Re,nbar,ihat_r)
%<<==  A2
    A2 = -3*mu*J2*Re^2*nbar/norm(r)^5*(nbar'*eye(3)-5*(ihat_r'*nbar)
*ihat_r')...
        - 3*mu*J2*Re^2/(2*norm(r)^5)*(eye(3)-5*ihat_r*ihat_r');
end

function [A3] = A3(r,mu,J2,Re,nbar,ihat_r)
%<<== A3
    A3 = 7.5*mu*J2*Re^2/norm(r)^5*((ihat_r'*nbar)^2*eye(3)...
        -5*(ihat_r'*nbar)^2*ihat_r*ihat_r' +...
        r*ihat_r'*nbar*nbar'*eye(3)/norm(r)*(eye(3)-ihat_r*ihat_r'));
end

function crsform=crs(vect)
%calculate the cross product form of a vector

crsform = [0        -vect(3)    vect(2)
           vect(3)  0           -vect(1)
           -vect(2) vect(1)     0];
end
```

E.5 Potter Update

2/16/10 9:47 PM Block: SatelliteSIM/Sa.../Embedded MATLAB Function 1 of 1

```
function [xPotter, Spotter] = fcn(x,S,H,R,ytilde,yhat)
% This block supports the Embedded MATLAB subset.
% See the help menu for details.
numMeas = size(R,1);

for i=1:numMeas
    Htemp = H(i,:);
    Rtemp = R(i,i);
    ytildetemp=ytilde(i);

    a = S'*Htemp';
    b=1/(a'*a+Rtemp);
    gamma = 1/(1+sqrt(b*Rtemp));
    K = b*S*a;
    xnew=x+K*(ytildetemp-yhat(i));
    Snew=S-gamma*K*a';

    x=xnew;
    S=Snew;
end

%output answer
xPotter=x;
Spotter=S;
```

E.6 Carlson Update

2/16/10 9:50 PM Block: SatelliteSIM/Sa.../Embedded MATLAB Function 1 of 1

```
function [xCarlson, SCarlson] = fcn(x,S,H,R,ytilde,yhat)
% This block supports the Embedded MATLAB subset.
% See the help menu for details.
numMeas = size(R,1);
n=max(size(S)); %get size of S matrix
Snew = NaN(size(S));

e=zeros(n,1);
d=R(1,1);
xnew = x;
for j=1:numMeas
    Htemp=H(j,:);
    Rtemp=R(j,j);
    ytildetemp=ytilde(j);

    %Initialize d0,e0 and a

    d0=Rtemp;
    e0=zeros(n,1);
    a=S'*Htemp';

    %%
    for i=n:-1:1 %1:n  %go backwards to maintain lower triangular nature of
S matrix
        d=d0+a(i)^2;
        b=sqrt(d0/d);
        c=a(i)/sqrt(d0*d);
        e=e0+S(:,i)*a(i);
        Snew(:,i)=S(:,i)*b-e0*c;

        d0=d;
        e0=e;
    end

    xnew=x+e*((ytildetemp-yhat(j))/d);

    x=xnew;
    S=Snew;
end

%output answer
xCarlson=xnew;
SCarlson=Snew;
```

E.7 UD factorization Update

2/16/10 9:52 PM Block: SatelliteSIM/Sa.../Embedded MATLAB Function 1 of 1

```
function [xUD, U_UD, D_UD] = update(x,U0,D0,H,R,ytilde,yhat)
% This block supports the Embedded MATLAB subset.
% See the help menu for details.
numMeas = size(R,1);
n=max(size(D0)); %get size of S matrix

% assume the measurements are uncorrelated
v=NaN(n,1);
b=NaN(1,n);
p=NaN(1,n);
a=NaN(1,1);
xnew=x;
U=U0;
for l=1:numMeas
    %Make measurement scalar
    Htemp=H(l,:);
    Rtemp=R(l,l);
    ytildetemp=ytilde(l);

    %initialize
    f=U0'*Htemp';
    for j=1:n %n:-1:1%
        v(j) = D0(j,j)*f(j);
    end
    a0=Rtemp;

    %run iterations
    U=U0;
    for k=1:n%n:-1:1 %
        a = a0+f(k)*v(k);
        D0(k,k) = D0(k,k)*a0/a;
        b(k)=v(k);
        if k>1
            p(k)=-f(k)/a0;
            for j=1:k-1 %k-1:-1:1%
                U(j,k)=U0(j,k)+b(j)*p(k);
                b(j)=b(j)+U0(j,k)*v(k);
            end
            U0=U;
        end
        a0=a;
    end

    %update state
    K=b'/a;
    xnew=x+K*(ytildetemp-yhat(l));

    x=xnew;
end

xUD=xnew;
U_UD  = U;
D_UD = D0;
%P_UD=U*D0*U'
```

E.8 Potter/Carlson Covariance Propagation

2/16/10 9:49 PM Block: SatelliteSIM/Sa.../Embedded MATLAB Function 1 of 1

```
function SPotterProp  = fcn(S0,Phi,B,Wd)
% This block supports the Embedded MATLAB subset.
% See the help menu for details.
S_modified = [Phi*S0 B*Wd];

[q,Sprop_trans]=MGS(S_modified');

SPotterProp = Sprop_trans';

end

function [Q,R]=MGS(A)
% Modified Gram-Schmidt orthoginalization process
% Generates m by n matrix Q and n by n matrix R such that
% A = QR;
%
% INPUT
% A = matrix where columns are independent vectors
% OUTPUT
% Q = Orthonormal basis for vector space spaned by the columns of A
% R = Upper triangular matrix whos columns are a basis for vector space
% spaned by the columns of A
%
% EXAMPLE
% [Q,R] = MGS(A)
%
% Written 8/18/09 by Jason Schmidt based on gramschmidtmat.pdf

[m,n]=size(A);
Q=zeros(m,n);
R=zeros(n,n);

for j=1:n
    v=A(:,j);                       %v begins as column j of A
    for i=1:j-1
        R(i,j)=Q(:,i)'*v;  %A(:,j); %replacing A(:,j) with v yields modified
Gram-Schmidt
        v=v-R(i,j)*Q(:,i);          %subtract the projection (qi^T aj)qi =
(qi^T v)qi
    end
    R(j,j)=norm(v);
    Q(:,j)=v/R(j,j);                %normalize v to be the next unit vector
q
end

end
```

E.9 UD factorization Covariance Propagation

2/16/10 9:54 PM Block: SatelliteSIM/Sa.../Embedded MATLAB Function 1 of 1

```
function  [U_UDprop, D_UDprop] = prop(U0,D0,Phi,B,Qd)
% Note that this propagation is identical to the Potter Filter
% This block supports the Embedded MATLAB subset.
% See the help menu for details.
eml.extrinsic('blkdiag')
eml.extrinsic('ldl')
[m,n]=size(D0);
%[o,p]=size(Qd);
D_modified=NaN(2*m,2*n);
u=NaN(m,m);
Qdnew=NaN(m,m);
p=NaN(m,m);
%Propagate UDU covariance matrix
[u,Qdnew,p]=ldl(B*Qd*B','upper');
Bnew=p'\u';
%[Bnew,Qdnew]=udu(B*Qd*B');
Y=[Phi*U0 Bnew];
D_modified = blkdiag(D0,Qdnew);
[D_UDprop,U_UDprop]=MWGS(Y',D_modified,D0,U0);
%Pprop_UD = Uprop_UD*Dprop_UD*Uprop_UD'
end


function [D,U]=MWGS(A,D_mod,D,U)
% Modified Weighted Gram-Schmidt orthoginalization process
% based on Maybeck section 7.7 page 397
% Written 8/20/09 by Jason Schmidt

n=size(A,2);
c=NaN(size(A,1),n);
d=NaN(size(A,1),n);
for k=n:-1:1
    for j=1:1:size(A,1)
        c(j,k) = D_mod(j,j)*A(j,k);
    end
    D(k,k)=A(:,k)'*c(:,k);
    if k>1
        d(:,k)=c(:,k)/D(k,k);
        for j=1:1:k-1
            U(j,k)=A(:,j)'*d(:,k);
            A(:,j)=A(:,j)-U(j,k)*A(:,k);
        end
    end
end

end
```