A Monte-Carlo Performance Analysis of Kalman Filter and Targeting Algorithms for Autonomous Orbital Rendezvous

by

Andrew Thomas Vaughan

B.S. Aerospace Engineering, Virginia Tech, Blacksburg, VA, 2002

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Master of Science in Aeronautics and Astronautics
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2004

Author . . . .
Department of Aeronautics and Astronautics
May 7, 2004

Certified by . . . .
David K. Geller, Ph.D.
Senior Member of the Technical Staff
The Charles Stark Draper Laboratory, Inc.
Technical Supervisor

Certified by . . . .
Richard H. Battin, Ph.D.
Professor, Department of Aeronautics and Astronautics
Thesis Advisor

Accepted by . . . .
Edward M. Greitzer, Ph.D.
H.N. Slater Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

[This page intentionally left nearly blank.]

A Monte-Carlo Performance Analysis of Kalman Filter and Targeting Algorithms for Autonomous Orbital Rendezvous

by

Andrew Thomas Vaughan

Submitted to the Department of Aeronautics and Astronautics
on May 7, 2004, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Autonomous orbital rendezvous with an orbiting sample (OS) is seen as an enabling technology for a Mars Sample Return (MSR) mission, so several demonstrations have been planned. With CNES cooperation, a proposed rendezvous demonstration was governed by ITAR restrictions, and a guidance and navigation system was designed using a Precomputed Gain Kalman filter and targeting algorithms. Having lost CNES participation, the opportunity now exists to use a full Extended Kalman filter with onboard targeting algorithms on a new demonstration using the Mars Telecommunications Orbiter (MTO). This creates an impetus to compare the Precomputed Gain system with the Extended system to determine their relative performance.

This thesis aims to compare the Precomputed Gain and Extended Kalman filters and associated targeting algorithms using a Monte-Carlo analysis, and based on quantitative performance metrics including: total change in velocity required, navigation errors, target pointing errors. In addition, other aspects of the algorithms will be studied including: technology readiness level (TRL), data uplink requirements, and complexity and computational burden for the onboard algorithms.

Monte-Carlo analysis will reveal that the Extended system modestly outperforms the Precomputed Gain system in total change in velocity required, navigation error, and target pointing error, with a larger performance envelope. The Extended system will also be found to have a greater technology readiness and require substantially less data uplink. The Precomputed Gain system will be found to be a significantly less complex algorithm for the onboard flight computer.

Technical Supervisor: David K. Geller, Ph.D.
Title: Senior Member of the Technical Staff
The Charles Stark Draper Laboratory, Inc.

Thesis Advisor: Richard H. Battin, Ph.D.
Title: Professor, Department of Aeronautics and Astronautics

[This page intentionally left nearly blank.]

Acknowledgments

In a letter to Robert Hooke in 1675, Sir Isaac Newton once wrote: *"If I have seen further, it is by standing upon the shoulders of giants."*

It is my hope that, after 2 years of hard work at MIT and the Draper Laboratory, I have actually seen just a little bit further. Not just further than I could see two years ago, because that is no doubt true; further, in fact, than anyone else has ever seen, even if only in one particular direction. Perhaps someday I will be fortunate enough to have someone who needs to stand on my shoulders; today, however, I just have a lot of giants that I need to thank.

Foremost, I can't express enough gratitude to my parents, Kit and Don, who have supported me academically, emotionally, and financially, for all these years. Without them, I would have been nothing in the beginning, and probably nothing in the end.

Thanks to my entire family. To my sisters Ali and Becca, eternally supportive of all my adventures, even if they take me hundreds or thousands of miles from home. To my grandmother, and all of my aunts, uncles, and cousins who have been so patient with me hiding in my corner of Cambridge for two straight years.

Amanda, my stars and moon for the last 20 months, has kept me going through my time at MIT, even across 3000 miles of phone lines.

Thanks to Mike and Rob, for friendships so strong that they can be maintained across a continent with no effort, and then resumed for a weekend visit as if they were never gone.

Thanks to Geoff, Steve, and Megan, apartment-mates and good friends. Thanks to Dave W., good friend and equally knowledgeable in matters of work and life in general. Although we all may go different ways, here's to many more years of friendship from wherever we end up.

I appreciate the efforts of all the MIT faculty that have made this a pleasant (and, of course, educational) graduate experience. In particular, Dr. Richard Battin, twice my Astrodynamics instructor and once my thesis advisor, has made learning a pleasure at MIT.

I have been uncommonly fortunate to have David Geller as my research advisor at the Draper Laboratory. Advisor, teacher, colleague, and friend, it has been an honor to spend 2 years working with Dave. I am not alone in wishing him the best of fortunes, wherever his career takes him.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under the MORS Contract #1215138 sponsored by the Jet Propulsion Laboratory.

Andrew Thomas Vaughan ....

[This page intentionally left nearly blank.]

[This page intentionally left nearly blank.]

Contents

[This page intentionally left nearly blank.]

List of Figures

List of Tables

[This page intentionally left nearly blank.]

Chapter 1

Introduction

With the launch of Sputnik on October 4, 1957, the human race forever became a space-faring society. When President John Kennedy responded publicly to the Russian challenge on September 12, 1962 in his visionary and historic fashion, he ushered in a new era of technological innovation that would have permanent effects on American society, and indeed the world.

> *We choose to go to the moon. We choose to go to the moon in this decade and do the other things, not because they are easy, but because they are hard, because that goal will serve to organize and measure the best of our energies and skills, because that challenge is one that we are willing to accept, one we are unwilling to postpone, and one which we intend to win, and the others, too.* [14]

Kennedy's vision brought with it a host of challenges, not least of which was the question of guiding a manned spacecraft while traveling to and from the moon, and landing on its surface. To a large extent, this task would bring the field of spacecraft guidance and navigation into its own, and would thrust the MIT Instrumentation Laboratory, now the Charles Stark Draper Laboratory, into the forefront of the field.

Much progress has been made since 1962. The Instrumentation Laboratory successfully designed and built the Apollo guidance computer. NASA has successfully sent robotic probes to orbit or fly-by every planet but Pluto, and has successfully landed several vehicles on the surface of Mars. The NEAR spacecraft has successfully orbited (and ultimately 'landed' on) the Eros asteroid, while the STARDUST mission has successfully flown through the tail of a comet. The Space Shuttle and Russian Progress re-supply vehicles routinely rendezvous with the International Space Station (ISS). What, then, is the next task to push the limits of modern guidance and navigation?

Having 'conquered' the moon, the next tick-mark on NASA's solar system yardstick seems to be Mars. Indeed, a great deal of time, money, and research is already focused on the ultimate goal of a manned Mars mission. There are many steps that must be taken, many technologies that must be proven, before a manned Mars mission can succeed. NASA must fully understand the nature of the destination, a worthwhile venture in itself that includes geography, climate, and composition. To this end, missions are planned or underway to answer many of these questions, the most ambitious of which is undoubtedly the Mars Sample Return (MSR) mission.

Mars sample return presents problems unlike any that NASA has solved before. A 'single-shot' sample return mission would require the following components to be lifted from the Earth's surface and entered into a Mars transfer orbit: Mars atmosphere entry-interface hardware, possibly including hardware such as a heat shield, parachutes, retro-rockets, or airbags; sample collection hardware, possibly including a robotic arm, drill, or rover for obtaining interesting samples; Mars ascent and Earth transfer hardware, including fuel or the ability to manufacture fuel on the surface; and, finally, Earth entry hardware to protect the sample from a violent re-entry into the atmosphere. When all of the hardware is accounted for, the cost to return just a few kilograms of the Mars surface to Earth is prohibitively expensive in mass, price, and risk. Fortunately, there are alternatives which involve breaking the mission into several stages. A common division in many current approaches is the separation of the Earth return vehicle from the sample collection equipment and Mars ascent vehicle. That is, the sample collection equipment and Mars ascent vehicle are landed on the surface of Mars, while the Earth return vehicle (and the required fuel) remain in orbit for later use. This has a significant advantage over the 'single-shot' approach, in that the Mars ascent vehicle is not required to boost the fuel and thruster hardware for the return trip off of the planet's surface. Such an approach is called for by the Groundbreaking Approach to MSR, proposed by NASA's Jet Propulsion Laboratory (JPL) in late 2002 [10].

Of course, the tacit assumption to this approach is that the Earth return vehicle can safely rendezvous with the newly-launched orbiting sample (OS), and secure it for the return trip. Given the time delay in sending signals to and from Mars, the rendezvous would seem to require some level of autonomy, and ceases to become a trivial task. It is this particular challenge of autonomous rendezvous that the present work deals with.

1.1 Kalman Filtering

While R. E. Kalman might have his name attached to today's form of the least-squares estimator, the idea has been around for more than two hundred years. Motivated by an orbit-prediction problem, Karl Friedrich Gauss developed a theory of least-squares estimation in 1795 in which measurements are made in discrete time. Gauss did not publish his work until 1809, leaving room for Adrien-Marie Legendre to independently develop and publish a similar least-squares estimator in 1806.

It was not for 130 years before significant advances would again be made in the field of optimal estimation. In 1941, Andrey Nikolaevich Kolmogorov developed a technique for discrete-time, linear, minimum mean-square estimation, derived independently by Wiener in 1942 for continuous-time. The Wiener-Kolmogorov theory, though valid for both continuous and discrete problems, was formulated in the frequency domain rather than the "state-space" of the Kalman filter, making it unwieldy for many engineering applications. In addition, Wiener's technique was limited in application to statistically stationary processes, unless extended by cumbersome and computationally intensive techniques developed by others in two decades of work after 1942.

Finally, in a third instance of near-simultaneous discovery that seems to plague the field, Rudolph Emil Kalman and Peter Swerling developed the modern, discrete time, recursive, mean-square filter that is prevalent today. Kalman's seminal paper, published in 1960, is generally regarded as the cornerstone of modern optimal estimation [7]. Fundamentally, Kalman's work can be seen as a recursive solution to Gauss's least-squares estimator, published 160 years earlier. Still, the recursive estimator lends itself marvelously to implementation on the digital computer, which was beginning to come of age about this time. The Kalman filter became quite popular and has since found application in orbit determination, global positioning, gravity field modeling, chemical process modeling, atmospheric density profiling, and spacecraft navigation, among many others.

Much research has been dedicated to the Kalman filter since 1960, however the basic technique remains largely the same. Some work, such as the recently developed particle filter, has gone beyond the basic Kalman filter. The rest has created a huge knowledge base surrounding optimal least-squares estimation dealing with subjects such as: handling divergence, preventing instabilities, reducing computational burden, and using simple random variables to model complex processes. Variations on the Kalman filter, such as the so-called "extended Kalman filter," have been developed to reduce approximation errors resulting from the linearization of the system being modeled. The extended Kalman filter, used in this work, will be discussed in more detail in Section 6.7.

1.2 Spacecraft Navigation

Gauss was motivated to develop his optimal estimator by an orbit determination problem, and it has similarly been the fate of Kalman filters to find widespread application in the astronautical sciences. Since the days of the Apollo missions to the moon, recursive estimators have found their way into the guidance computers of a variety of spacecraft.

One particularly interesting use for the Kalman filter in spacecraft navigation is in the autonomous ("closed-loop") rendezvous of two spacecraft in orbit about a third body. In this problem a chaser vehicle uses instruments such as Light Detection and Ranging (LIDAR) equipment or optical navigation cameras to take range and angle measurements to the target vehicle. A Kalman filter is used to process the noisy measurements and generate an estimate of the inertial position and velocity of one or both vehicles, allowing a targeting algorithm to accurately perform the necessary rendezvous maneuvers.

A promising example of autonomous rendezvous comes from the budding Mars program on NASA's agenda, the MSR mission described above. Recognizing that the best way to analyze the Martian surface is to process samples in sophisticated Earth-based laboratories, NASA began many years ago to asses the required technologies to make Mars sample return possible. Likewise, in March of 1998 the French Space Agency (CNES) resolved to engage in a program of Mars exploration that would include orbital observation, surface-based science, preparation for sample return, and sample return. The 2007 launch window was targeted for the first PREMIER (Programme de Retour d'Echantillons Martiens et Installation d'Expériences en Réseau) orbiter, designed (among other objectives) to demonstrate the rendezvous phase of a sample return mission. Cooperation between NASA and CNES led to a contract for JPL to provide onboard guidance and navigation capabilities for the PREMIER-07 orbiter. Interestingly, during a design iteration on the PREMIER-07 orbiter, the guidance and navigation system lost the use of a dedicated processor, so JPL could no longer use a stand-alone computer and was forced instead to deal in algorithms to place on the primary computer. With this new strategy, JPL fell under restrictions from the International Trafficking in Arms (ITAR) regulations, preventing them from giving CNES the full algorithms previously designed for the PREMIER orbiter.

To satisfy ITAR restrictions, a clever guidance and navigation scheme was developed at JPL that merely required the onboard computer to perform matrix multiplication to obtain state updates and maneuver corrections. The solution was to use a *Precomputed Gain* Kalman filter that was linearized about a *nominal reference trajectory.* Thus, all of the required matrices could be calculated on the ground and uploaded to the spacecraft, which could then perform the rendezvous as long as the vehicles stayed near the predetermined reference trajectory. This system was tested exhaustively and proved to be sufficiently robust for the MSR rendezvous phase.

Subsequent to the development and testing of the Precomputed Gain navigation and targeting algorithms, the CNES PREMIER-07 mission was canceled. Fortunately for those involved, Mars Sample Return and the rendezvous demonstration were deemed sufficiently important by NASA to continue the project despite the lack of CNES participation. Without the French involvement, however, there was no longer a need for the Precomputed Gain system used to satisfy ITAR restrictions. The question then arises: what type of filter and targeting algorithms should be used for the rendezvous phase of the MSR mission demonstration? Each system has advantages and disadvantages, in terms of comparative metrics such as onboard computational intensity, complexity, technology readiness level (TRL), and performance envelope. This thesis seeks to answer that question in a very quantitative way, comparing the Precomputed Gain and extended implementations in simulation, and evaluating the results in terms of performance metrics described in Section 2.2. Using this analysis, the reader should be able to make an informed decision between the Precomputed Gain and Extended implementations of the Kalman filter for use in his or her particular rendezvous problem.

Chapter 2

Scope, Objectives, and Notation

The goal of this chapter is to describe in detail what this thesis aims to cover. Section 2.1 defines the scope of the thesis, including the types of algorithms that will be used, the baseline mission parameters, and the type of analysis. Section 2.2 outlines the primary objective of this thesis: to quantitatively evaluate the two flight computer algorithms based on several well-defined performance metrics. Finally, Section 2.3 explains the system of notation that will be used throughout the work.

2.1 Scope

The problem of autonomous rendezvous is a wonderfully interesting challenge with a multitude of possible angles of study, using any of several unique analysis tools, applied to a near infinite set of practical orbital elements and rendezvous trajectories. The range of potential research topics is so vast that it is essential to properly define a scope and keep this thesis manageable. Scoping begins with the navigation and targeting algorithms that will be studied in this work.

As described in the background material in Section 1.2, two primary implementations of the Kalman filter have been considered for the MSR mission. The first is a *Precomputed Gain Kalman filter using a corrective maneuver targeting strategy.* This implementation of the Kalman filter uses measurement, maneuver, and state transition matrices calculated around a reference rendezvous trajectory to perform guidance and navigation in a manner that satisfies ITAR restrictions. The full rationale, motivation, and implementation for this strategy are described in Chapter 8. The second implementation considered is an *extended Kalman filter with onboard targeting algorithms.* This strategy is not ITAR-friendly, but it is expected to be a more robust algorithm that can be used in the absence of international participation. The rationale, motivation, and implementation for this strategy are also described in Chapter 8. Considering the need to limit scope, and that these are the two implementations of current interest to the MSR mission, they are the only ones that will be studied in this work.

It is quite possible to study the two algorithms in question using a variety of analysis tools, such as Linear Covariance or Monte-Carlo techniques. The analysis tool that will be used in this study is the Monte-Carlo simulator described in detail in Chapter 5. While the results from the simulator will be evaluated using several performance metrics, this is the only analysis tool that will be used to test the algorithms above.

In order for this thesis to have the most practical application possible, the orbital elements selected for study will be those of the proposed Mars Telecommunications Orbiter (MTO), as detailed in Section 3.3.1. At the time of this writing, MTO is the most likely candidate to first perform a demonstration of the rendezvous phase of a MSR mission. It is assumed that the expected sensor package for the MSR mission will drive the MTO demonstration sensor package, so the physical parameters required in the analysis will be derived from these instruments. Accordingly, the simulation is configured to use an optical camera to take range and angle measurements to the orbiting sample, as is expected from the real mission. The orbiting sample size and shape, attitude sensor errors, camera alignment errors, camera measurement noise, and maneuver execution error all take their nominal values from those anticipated for the MTO/MSR missions. The numerical values used in the analysis can be found in Chapter 9.

When performing orbital rendezvous there are several basic trajectories that are often combined to create a nominal rendezvous trajectory (see Section 3.2). These basic building blocks can be combined in a large variety of ways to create many different and practical rendezvous trajectories. For this analysis, a single reference trajectory, hereafter called the *baseline trajectory*, was used for every Monte-Carlo trial. This baseline trajectory was selected after evaluating the author's prior research into trajectory designs for the MTO mission. The baseline trajectory can be found in Section 3.3, with appropriate justification for its use.

2.2 Objectives

The objective of this study is to compare the algorithms above on the basis of several meaningful performance metrics that will provide a useful comparison between the two algorithms. The following is a list of the metrics that are used to compare the algorithms, including a brief motivation and description for each. Detailed descriptions can be found in the results section of this thesis, Chapters 9 through 12.

2.2.1 Performance Envelope

The most important and quantitative metric that is used is the *performance envelope*. Given the two algorithms under consideration, what are the mean and variance of the total Δv required to perform the rendezvous? How large is the navigation error as a function of time? How robust are the algorithms when exposed to larger than expected values of execution error or unmodeled accelerations? How do the algorithms perform in non-circular orbits? These kinds of questions can be answered statistically using the results from large Monte-Carlo trials.

It should be noted that, using a deterministic analysis tool such as Monte-Carlo simulation with multiple tunable parameters, the size of the analysis can grow very quickly to unmanageable proportions. Even with the problem scoped as described in Section 2.1 (two algorithms, one set of orbital elements, one baseline reference trajectory), the Monte-Carlo analysis can still become unwieldy. The performance envelope is thus considered in the following ways, divided as the author sees it into *input strategies* and *output strategies*.

Input Though the simulator has many tunable parameters as described in Chapter 5, the two that are considered of greatest importance are the maneuver execution error and unmodeled acceleration terms. These two parameters are thus considered the primary 'design knobs' when performing the analysis. With that in mind, input strategies for the Monte-Carlo simulator were designed in three principal ways.

First, a set of *nominal inputs* were derived from the actual expected mission values of all tunable parameters. This creates a base case that represents a mission where all design parameters fall within the range expected by the engineers. Second, a set of *off-nominal inputs* were created by selecting a set of unmodeled accelerations and execution errors that are plausible, but either worse or better than the values expected by the mission engineers. In this case, like the nominal inputs, the navigation algorithms are configured to expect the off-nominal values of unmodeled acceleration or execution error. These inputs represent, for example, a case where engineers discover in the final testing of the spacecraft that the thrusters are not as consistent as expected, and a larger execution error will be injected than originally anticipated. In this scenario, the navigation algorithm can be modified to expect this error before launch. Finally, a set of *stress case inputs* were created by combining off-nominal environmental inputs with nominal filter values. For example, it might happen that the actual magnitude of unmodeled accelerations seen in flight is larger than the value the navigation algorithm was programmed at launch to expect. This would stress the capabilities of the navigation algorithms and their response to this stress is an important quality to study. Numerical values for these various input strategies can be found in Chapter 9.

Output Driven by the input strategies listed above, the Monte-Carlo tool will simulate the orbital rendezvous many, many times, resulting in a large quantity of data for each input strategy. The question still remains: by what *metrics* will the performance envelope of these algorithms be measured? Several quantitative and qualitative measures of performance have been identified and are used to evaluate the results of the Monte-Carlo cases. Though briefly listed here, these metrics are described in detail in Chapter 9, as they appear.

> The following quantitative metrics will be used to evaluate the performance envelope of the two sets of navigation and targeting algorithms: mean and standard deviation of the required $\Delta \boldsymbol{v}$; magnitude of position and velocity navigation error, particularly at specific epoch times such as closest approach; position dispersions at epoch times such as maneuver times and closest approach; and, target pointing error at epoch times, such as when the target leaves eclipse. Several qualitative representations will be presented as well, including Monte-Carlo "hair" plots and the passive-abort outlook for a given case.

One final metric is used to address the performance envelope of these algorithms. In general, the orbits of both spacecraft (but particularly the orbiting sample) have been assumed to be very nearly circular. Since the actual vehicle orbits are not likely to be exactly circular, a discussion is included of how the algorithms perform in slightly eccentric orbits, for example $0 < e < 0.04$. It is believed that this metric, combined with those listed above, provides a thorough analysis of the performance envelope that will be useful to the reader in judging the merits of the Precomputed Gain and Extended Kalman filters.

2.2.2 Technology Readiness Level

When selecting a navigation and targeting algorithm, it is important to consider several things besides the performance envelope. In particular, heritage is a prime factor that is considered when choosing space-flight hardware and software. For software algorithms, 'heritage' is most easily quantified in terms of Technology Readiness Levels (TRLs), which are used by NASA and other organizations to asses a technology's maturity. In Chapter 10, the various TRLs will be defined and the algorithms under consideration will be placed, based on their heritage and current state of development.

2.2.3 Complexity and Computational Burden

In the early history of digital computers on spacecraft, processor power was at a premium and onboard computational burden was the subject of much analysis and refinement. Today, spacecraft processors have become so powerful that, in most applications, the computational intensity of guidance and navigation algorithms is not a primary design constraint. Still, the complexity and computational burden of these algorithms do have an effect on the overall design and merit some measure of serious study.

The *complexity* of a given set of navigation and targeting algorithms is measured by the approximate number of lines of code required in the onboard system. Increasingly complex systems have consequences in terms of larger storage and memory usage, as well as greater chances for human error in the programming. Simple systems do not require much in the way of computer resources, and are easy to debug. While the number of lines of code can be used as a quantitative assessment of complexity, there must be some allowance made qualitatively for the ground-based portion of the algorithm. For example, it is known that the Precomputed Gain algorithm requires only matrix multiplication onboard the spacecraft and so likely has a much lower onboard computational intensity. This system requires, however, that the Kalman filter be essentially split in half, with much of the work performed on the ground and then pieces uploaded to the spacecraft. It could be argued that this system is indeed *more complex* than simply putting the Extended filter on the spacecraft.

The *computational burden* imposed by a given algorithm is measured either experimentally by the required process time, or analytically by evaluating (or approximating) the required number of computations to complete a cycle. Only the onboard portion of each algorithm is considered, as the ground-based piece can be performed with sophisticated tools on very powerful computers. The analysis and detailed descriptions for complexity and computational burden can be found in Chapter 11.

2.2.4 Data Uplink Requirements

There is a definite disparity between the data uplink requirements of the two algorithms under consideration. The Precomputed Gain system requires a large periodic upload of data, and would have no navigational abilities beyond the current experiment if communication is lost. The Extended algorithm has no such requirement and would typically need only basic trajectory parameters to perform the rendezvous. This disparity could be a significant consideration in some missions, and so is quantified for the Precomputed Gain and Extended algorithms in Chapter 12.

2.3 Notation

Notational standards are many and varied. It was left to the author to pick a suitable system and be consistent in its use throughout this work. The following pages will be filled with equations containing scalars, vectors, matrices, quaternions, partial derivatives, and subscripted and indexed variables. The following typesetting styles will be used as consistently as possible to distinguish these terms.

Scalars Scalar variables such as range, r, and azimuth, α, are represented by italicized lower case characters. For scalar random variables, the convention in literature is to use italicized upper case characters, not in bold face. For the few random variables in this work, such as R and S, this convention will be used.

Vectors Vector variables such as position, $\boldsymbol{r}$, and velocity, $\boldsymbol{v}$, are represented by bold and italicized lower case characters. Unit vectors are indicated by a hat, as in the case of a unit Euler axis, $\hat{\boldsymbol{u}}$. In Kalman filtering, it is also standard notation to use a hat to represent an "estimated" quantity, such as the estimated state, $\hat{\boldsymbol{x}}$. The distinction between unit vectors and estimated quantities will be apparent in context.

Matrices Matrix variables such as the Kalman gain matrix, $\boldsymbol{K}$, and state transition matrix, $\boldsymbol{\Phi}$, are represented by bold capital characters, italicized if the font permits. Matrix inversion is indicated by a superscript -1, such as $\boldsymbol{P}^{-1}$. Likewise, matrix transposes are indicated by a superscript T, such as $\boldsymbol{H}^T$.

Quaternions Although vector-like in appearance, quaternions are not represented by bold characters. To distinguish quaternions from standard vectors, they are typeset as scalars but reserved to the character q. Typically, quaternions are distinguished by subscripts to indicate their function, such as $q_{I\rightarrow B}$ to represent the inertial to body transformation.

Time Derivatives Standard dot notation is occasionally used as a convenient way to represent time derivatives. Each dot represents a single time derivative, such as $\frac{d\boldsymbol{r}}{dt} = \dot{\boldsymbol{r}}$, or $\frac{d^2\boldsymbol{r}}{dt^2} = \ddot{\boldsymbol{r}}$.

Indexing It is often necessary to represent indexed variables, such as the current time step, t_i, the next time step, t_{i+1}, or the previous time step, t_{i-1}. These variables are represented by subscripted indices as shown.

Coordinate Frames Several coordinate frames are used throughout this work, including the inertial frame, the body frame, and the local-horizontal local-vertical (LVLH) frame. When it is necessary to specify the coordinate frame for a specific variable, the frame is represented by a descriptive capital superscript. For example, the relative position vector in the body frame is represented as $\boldsymbol{r}_{rel}^B$.

In general, angular quantities are typically represented by Greek symbols, while rectilinear quantities are represented by standard Latin characters. Subscripts are used to provide additional information about a particular variable. For example, $\boldsymbol{r}_{OS}$ could be used to represent the position of the orbiting sample. In some cases, notational consistency cannot be achieved due to variable naming schemes with significant precedent. These exceptions are noted as they arise. Note also that, in this work, the terms "Orbiting Sample" (OS) and "target" are used interchangeably, as are "Orbiter" (Orb) and "chaser". For the MSR mission and MTO demonstration, these ideas are equivalent.

Chapter 3

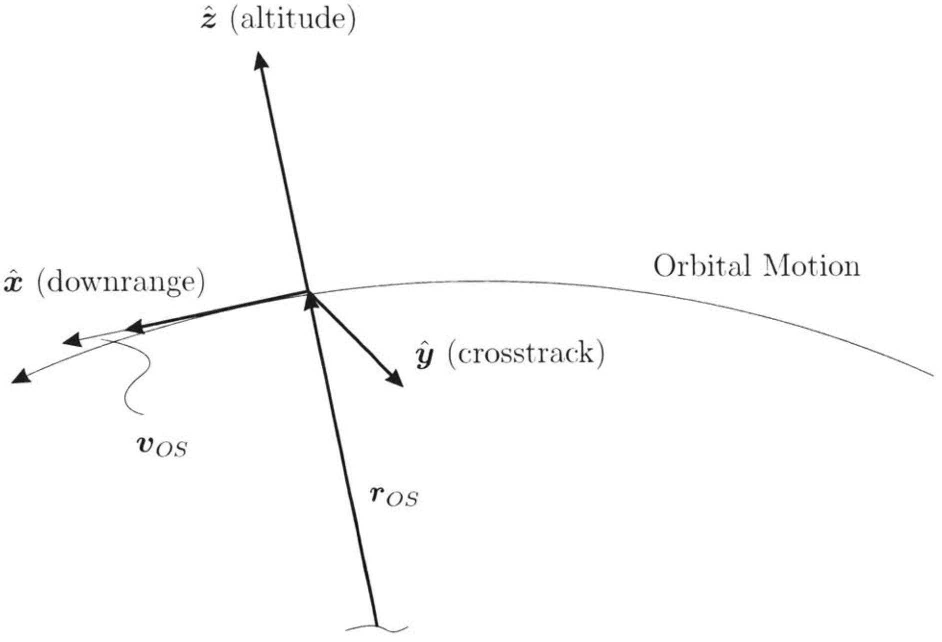
Reference Trajectory Design

In Section 2.1 it is mentioned that a baseline trajectory would be used for every Monte-Carlo trial in this work. In this chapter, that baseline trajectory is defined, and justification is presented for its use. Before the baseline is introduced, however, it is necessary to describe how relative motion is treated in this work, specifically the relative coordinate system that dominates the results. In addition, it will prove convenient to describe the standard 'building blocks' that comprise just about every rendezvous trajectory, and which were used to create the baseline trajectory for this thesis. Finally, Section 3.3.1 details the orbital elements for the MTO mission under investigation, and Section 3.3.3 describes the baseline trajectory itself.

3.1 Coordinate Systems

The inertial coordinate system used in the implementations of the Precomputed Gain and Extended Kalman filters is the "Mars-centered Mars Mean Equator and IAU-vector of Epoch J2000" system. This coordinate system is characterized by a z-axis in the direction of the rotational north pole and the x-axis in the direction of the IAU-vector. The IAU-vector is in the direction of the point where Mars's equator ascends through the Earth's equatorial plane at the standard J2000 Epoch [15]. The y-axis completes the right-hand set. Since the present analysis assumes a spherical, non-rotating Mars unless otherwise noted, the definition of the inertial coordinate

Figure 3-1: LVLH coordinate system definition


$\hat{\boldsymbol{z}}$ (altitude)
$\hat{\boldsymbol{x}}$ (downrange)
Orbital Motion
$\hat{\boldsymbol{y}}$ (crosstrack)
$\boldsymbol{v}_{OS}$
$\boldsymbol{r}_{OS}$

system is only relevant when finding the direction to the Sun in the Mars-centered frame (see Section 5.2.4).

The relative coordinate system used in this work is called the *local vertical-local horizontal*, or LVLH, coordinate system. The system is centered on the target and defined with the z-axis in the direction of the inertial position vector of the OS, the y-axis in the direction of the angular momentum vector of the OS, and the x-axis completing the right-hand set. This coordinate system is shown in Figure 3-1. Intuitively, the x-, y-, and z-axes of the LVLH relative frame can be considered the 'downrange', 'crosstrack', and 'altitude' directions with respect to the target, also indicated on the figure. Note that this coordinate system is rotating with mean angular rate $\boldsymbol{\omega}$ as shown in Equation 3.1.

$$\boldsymbol{\omega} = \sqrt{\frac{\mu}{a_{OS}^3}} \frac{\boldsymbol{r}_{OS} \times \boldsymbol{v}_{OS}}{|\boldsymbol{r}_{OS} \times \boldsymbol{v}_{OS}|} \tag{3.1}$$

Here a_{OS} is the semi-major axis of the OS and μ is the gravitational parameter for Mars. The LVLH coordinate system used in this thesis is actually a curvilinear coordinate system, meaning that the curvature of the orbit is removed when relative position is plotted on rectangular axes. For example, if the target and chaser vehicles were in identical orbits but the chaser vehicle was leading the target by a quarter of a period, a plot of downrange versus altitude would show a very large downrange component but no altitude component at all. For the types of downrange distances encountered in this thesis ($\lesssim$ 6 km), the variation is minor.

3.1.1 Coordinate Transformations

In this analysis there is a frequent need to transform vectors or matrices between two coordinate systems. A general discussion is provided here on coordinate transformations, followed by application to the main two transformations required in this work: inertial to LVLH, and inertial to body.

For an arbitrary position vector $\boldsymbol{r}$ expressed in two unique reference frames F_I and F_B (represented as $\boldsymbol{r}^I$ and $\boldsymbol{r}^B$, respective), there exists a rotation matrix that will map one vector to the other as shown in Equation 3.2.

$$\boldsymbol{r}^B = \boldsymbol{T}_{I \to B} \boldsymbol{r}^I \tag{3.2}$$

If the unit vectors describing the coordinate axes of F_B (expressed in F_I) are given by $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, and $\hat{\boldsymbol{z}}$, then the vector $\boldsymbol{r}^B$ can be expressed in terms of $\boldsymbol{r}^I$ as shown in Equation 3.3.

$$\boldsymbol{r}^B = \begin{bmatrix} \boldsymbol{r}^I \cdot \hat{\boldsymbol{x}} \\ \boldsymbol{r}^I \cdot \hat{\boldsymbol{y}} \\ \boldsymbol{r}^I \cdot \hat{\boldsymbol{z}} \end{bmatrix} \tag{3.3}$$

That is, each term of $\boldsymbol{r}^B$ is the component of $\boldsymbol{r}^I$ in the direction of the corresponding principle axis of F_B. Comparing Equation 3.2 with Equation 3.3, it can be seen that $\boldsymbol{T}_{I \to B}$ is composed of the vectors describing the axes of F_B, but represented in terms of F_I. Symbolically this shown in Equation 3.4.

$$\boldsymbol{T}_{I\rightarrow B} = \begin{bmatrix} \hat{\boldsymbol{x}} \\ \hat{\boldsymbol{y}} \\ \hat{\boldsymbol{z}} \end{bmatrix} \tag{3.4}$$

Allowing F_I to represent the inertial reference frame and F_B to represent the LVLH frame, Equation 3.4 will yield the *inertial to LVLH transformation* matrix, $\boldsymbol{T}_{I\rightarrow LVLH}$, provided that $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, and $\hat{\boldsymbol{z}}$ are defined as follows:

$$\begin{aligned} \hat{\boldsymbol{x}} &= \hat{\boldsymbol{y}} \times \hat{\boldsymbol{z}} \\ \hat{\boldsymbol{y}} &= \frac{\boldsymbol{r} \times \boldsymbol{v}}{|\boldsymbol{r} \times \boldsymbol{v}|} \\ \hat{\boldsymbol{z}} &= \frac{\boldsymbol{r}_{OS}}{|\boldsymbol{r}_{OS}|} \end{aligned} \tag{3.5}$$

Allowing F_I to represent the inertial reference frame and F_B to represent the body frame, Equation 3.4 will yield the *inertial to body transformation* matrix, $\boldsymbol{T}_{I\rightarrow B}$, provided that $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, and $\hat{\boldsymbol{z}}$ are defined as follows:

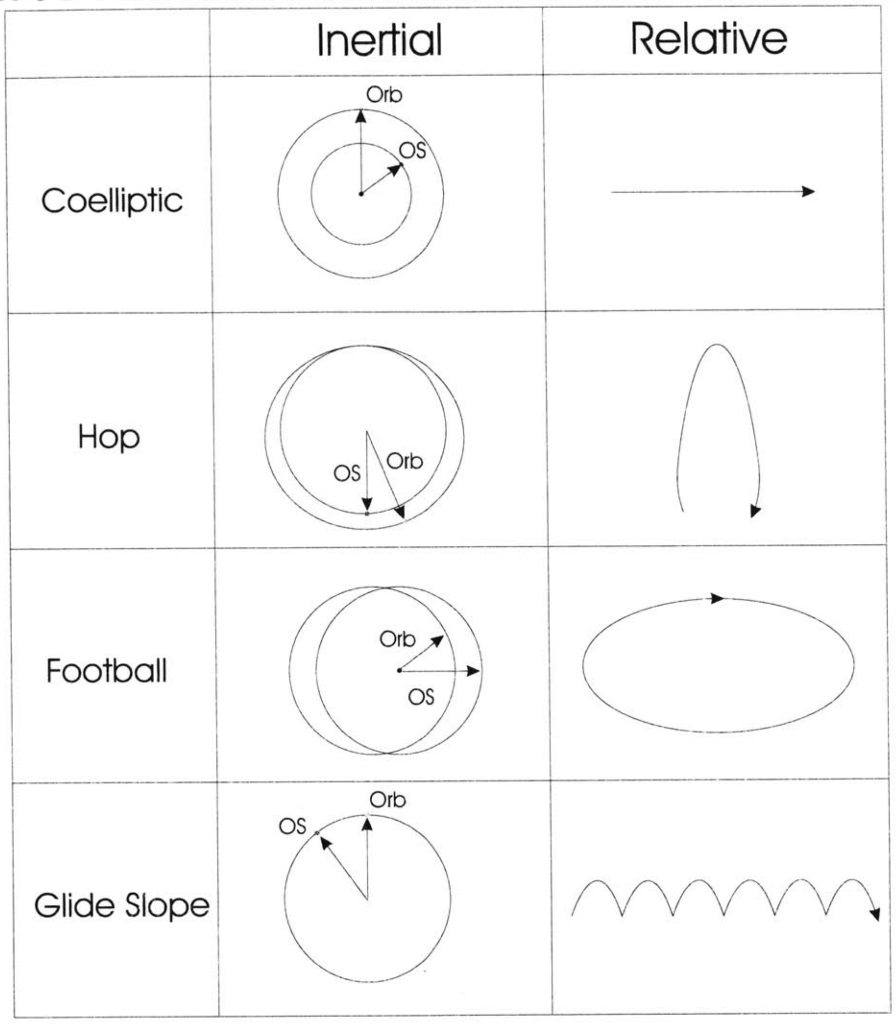
$$\begin{aligned} \hat{\boldsymbol{x}} &= \frac{\boldsymbol{r}_{rel}}{|\boldsymbol{r}_{rel}|} \\ \hat{\boldsymbol{y}} &= \frac{\hat{\boldsymbol{x}} \times \boldsymbol{\omega}}{|\hat{\boldsymbol{x}} \times \boldsymbol{\omega}|} \\ \hat{\boldsymbol{z}} &= \hat{\boldsymbol{x}} \times \hat{\boldsymbol{y}} \end{aligned} \tag{3.6}$$

Here $\boldsymbol{r}_{rel}$ is the relative position vector between the target and the chaser vehicles and $\boldsymbol{\omega}$ is the chaser vehicle's angular velocity vector in inertial coordinates. The relative position vector is defined as $\boldsymbol{r}_{rel} = \boldsymbol{r}_{Orb} - \boldsymbol{r}_{OS}$.

3.2 Relative Motion

As mentioned in Section 2.1, there are several basic building blocks used to design most full rendezvous trajectories. This section names each of these types of relative

Figure 3-2: Basic rendezvous maneuvers in Inertial and LVLH coordinates



motion and describes their main features, so that they may be used in Section 3.3 to construct the baseline rendezvous trajectory. The following relative trajectories, bearing descriptive names, are discussed in the paragraphs below: coelliptic approach, hop, football orbit, and glide slope. These trajectories are shown in both inertial and LVLH coordinates in Figure 3-2.

The Lambert maneuver is frequently used to generate relative trajectories, but is better classified as a targeting algorithm than a 'relative trajectory.' It is mentioned in the sections below but is described much later as a targeting algorithm, in Section 7.1.

3.2.1 Coelliptic

One of the most basic trajectories in an orbital rendezvous is the coelliptic approach. If the target and chaser vehicles have the same eccentricity but different semi-major axes, they will have different orbital periods and thus the distance between them will change monotonically in time until closest approach (for low eccentricity orbits). In the curvilinear LVLH coordinate system, a plot of the coelliptic approach is a horizontal line representing a changing relative downrange position with a constant relative altitude.

Several useful intuitions about the coelliptic approach can come from the CW/Hill equations of relative motion, particularly their solution for circular or near-circular orbits. Vallado [17] writes the equation for relative altitude, $z(t)$, as (using the notation of Figure 3-1):

$$z(t) = \frac{\dot{z}_0}{\omega}\sin(\omega t) - \left(3z_0 + \frac{2\dot{x}_0}{\omega}\right)\cos(\omega t) + \left(4z_0 + \frac{2\dot{x}_0}{\omega}\right) \tag{3.7}$$

Here $\dot{z}_0$ is the initial velocity in the altitude direction, $\dot{x}_0$ is the initial velocity in the downrange direction, z_0 is the initial altitude position, and ω is the mean orbital rate. Written in terms of the period, P, the mean orbital rate can be expressed as $\omega = 2\pi/P$. It is clear in Equation 3.7 that the two time varying terms will be zero if the following conditions are met:

$$\begin{aligned} \left(3z_0 + \frac{2\dot{x}_0}{\omega}\right) &= 0 \\ \frac{\dot{z}_0}{\omega} &= 0 \end{aligned} \tag{3.8}$$

Thus, constant altitude in time can be obtained when the initial relative radial velocity is zero, and when the initial relative downrange velocity has the value shown in Equation 3.9.

$$\dot{x}_0 = -\frac{3}{2}\omega z_0 = -\frac{3\pi z_0}{P} \tag{3.9}$$

It can therefore be seen that the closing rate between the chaser and target vehicles is $3\pi\Delta h/rev$, where Δh is the difference between the target and chaser semi-major axes (the relative altitude, or the difference in radius for circular orbits).

When performing a maneuver (considered in the inertial frame) to establish a coelliptic trajectory, the calculations for the required $\Delta\boldsymbol{v}$ are fairly straightforward. Here it is assumed that the chaser vehicle is already at the desired height to enter the coelliptic, and that the two vehicles are in coplanar orbits. If this is not the case, an additional maneuver is required to change the relative altitudes and/or remove the crosstrack position error (or a coelliptic approach can be initiated with a periodic out-of-plane motion). For the target spacecraft in a near circular orbit, the desired chaser velocity, $\boldsymbol{v}_{des}$, is simply the circular speed at the desired altitude, in the direction normal to the position and angular momentum vectors (the direction tangent to the new circular orbit):

$$\boldsymbol{v}_{des} = \sqrt{\frac{\mu}{a_{Orb}}} \frac{\boldsymbol{h}_{Orb} \times \boldsymbol{r}_{Orb}}{|\boldsymbol{h}_{Orb} \times \boldsymbol{r}_{Orb}|} \tag{3.10}$$

Here $\boldsymbol{h}$ represents the angular momentum vector, formed as shown below in Equation 3.11.

$$\boldsymbol{h}_{Orb} = \boldsymbol{r}_{Orb} \times \boldsymbol{v}_{Orb} \tag{3.11}$$

The required $\Delta\boldsymbol{v}$ is then the difference between the desired velocity and the current velocity. If the target is in an elliptical orbit, then the required calculation is less straightforward. For elliptical orbits, the definition of the coelliptic changes slightly, as an approach at a fixed relative Δh, or difference in altitude, becomes impossible. Instead, the goal is to force the difference in altitude at both apoapse and periapse to be the same. Enforcing this condition yields the result shown in Equation 3.12.

$$e_{OS}a_{OS} = e_{Orb}a_{Orb} \tag{3.12}$$

Here e is the eccentricity and a is the semimajor axis.

Coelliptic approaches are the cornerstone of most rendezvous trajectories, used as the primary means to close distance between the two vehicles. By controlling the relative altitude, the closing rate can be varied from very rapid (removing bulk distance between the vehicles) to very slow (controlled observational approach). In general, coelliptic approaches provide a favorable *passive abort* capability, meaning that the chaser has a very low probability of striking the target should control of the vehicle be lost.

3.2.2 Hop

Another basic rendezvous maneuver used to close distance between the target and chaser is shown in Figure 3-3. Based on its appearance, this maneuver is appropriately called a 'hop', as one arch is completed per orbit (two hops are shown in Figure 3-3).

In general a hop is initiated by a burn parallel to the chaser vehicle's velocity vector, which changes the energy of the chaser orbit and therefore its semi-major axis and period. If the burn is truly in the direction of the chaser velocity vector (not in the opposite direction), then energy is 'added' (energy becomes less negative), the chaser semi-major axis gets larger, and the chaser period gets longer. The chaser is then moving slower than the target in a mean-motion sense, so the target "catches up" (assuming that the chaser is initially in front of the target) to the orbiter by a prescribed distance every orbit. This behavior is shown in Figure 3-3. If the burn is performed in the direction opposite to the chaser velocity vector, the hop has a negative altitude component, and the net motion is away from the target vehicle (again assuming that the chaser is initially in front of the target).

Like in the case of the coelliptic maneuver, there is intuition to be gained by examining the CW/Hill solution for circular orbits as applied to the hop. Vallado [17] writes the equation for relative downrange component, $x(t)$ as:

$$x(t) = \left(6z_0 + \frac{4\dot{x}_0}{\omega}\right)\sin(\omega t) + \frac{2\dot{z}_0}{\omega}\cos(\omega t) - (6\omega z_0 + 3\dot{x}_0)t + \left(x_0 - \frac{2\dot{z}_0}{\omega}\right) \quad (3.13)$$

As before, $\dot{z}_0$ is the initial velocity in the altitude direction, $\dot{x}_0$ is the initial velocity in the downrange direction, z_0 is the initial altitude position, and ω is the mean orbital rate. Knowing from observation that the hopping motion is recurrent with the orbital period, Equation 3.13 can be evaluated at $t = P = 2\pi/\omega$ and equated to zero to find the downrange distance traveled each orbit. Assuming no initial altitude, as is common and shown in Figure 3-3, the resulting downrange motion in one orbit is:

$$x_0 = \frac{6\pi}{\omega}\dot{x}_0 \tag{3.14}$$

Perhaps of even more interest, the required initial relative downrange velocity (that is, the required $\Delta\boldsymbol{v}$ magnitude in the direction of the chaser velocity vector) can be found by rearranging Equation 3.14:

$$\Delta v = \dot{x}_0 = \frac{x_0\,\omega}{6\pi} \tag{3.15}$$

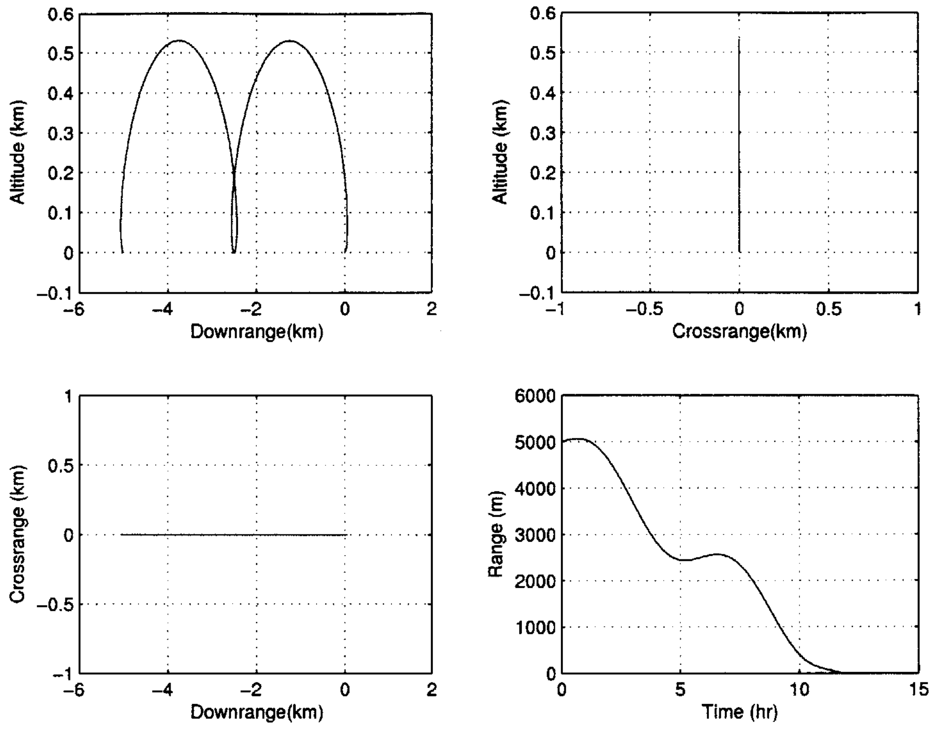
The resulting $\Delta\boldsymbol{v}$ magnitude is of course in the LVLH frame, but a transformation can be performed (easily, since the required burn direction is known in an inertial sense) if hop initial conditions are desired in inertial coordinates.

Contrary to the coelliptic approach and some of the maneuvers to follow, the hop has a relatively poor passive abort outlook. Since the hop trajectory touches the horizontal axis (the 'downrange' axis, hereafter referred to as the *v-bar*) once per orbit, it is possible for the chaser to collide with the target vehicle if control is lost in mid-course. This is particularly true in a 'real' rendezvous scenario, where knowledge errors and maneuver execution errors can cause the hop to be off-nominal. In this case, the trajectory might actually cross the v-bar twice per orbit, creating the possibility of a strike (particularly if the maneuver was targeted for close approach). For this reason, the hop did not find a place in the baseline trajectory, but is included here for completeness.

3.2.3 Football Orbit

Not every rendezvous trajectory 'building block' results in an average non-zero relative downrange velocity. Figure 3-4 shows a *football orbit*, named for the distinctly football-shaped relative path that is traced out every orbit. Often referred to as a 'holding football orbit,' this trajectory is frequently used to initialize full rendezvous

Figure 3-3: Hop approach from 5km downrange



trajectories because of its very stable nature and low control requirement.

In general a football orbit is initiated by a burn parallel to the chaser vehicle's position vector, when the target and chaser originally have the same semi-major axis. Because the change in velocity is typically *very* small compared to the orbital velocity of the chaser, and is applied in a normal direction, the change in the orbital energy is negligible. Consequently, the semi-major axis and period of the chaser orbit are unchanged, and there is no net change in position with respect to the target in a given orbit. The football orbit burn does change the *shape* of the chaser orbit, however, adding a small eccentricity and resulting in the relative trajectory shape shown in Figure 3-4.

Recognizing that the football orbit is characterized by equal energy between the target and chaser vehicles, it is not necessary to enter the football orbit as described above. At any point in the relative trajectory, an energy-matching (semi-major axis matching) burn can be performed to enter a holding football orbit at that point.

$$a_{Orb,des} = a_{OS} \tag{3.16}$$

$$\boldsymbol{v}_{des} = \frac{\boldsymbol{v}_{Orb}}{|\boldsymbol{v}_{Orb}|}\sqrt{\frac{2\mu}{|\boldsymbol{r}_{Orb}|} - \frac{\mu}{a_{Orb,des}}} \tag{3.17}$$

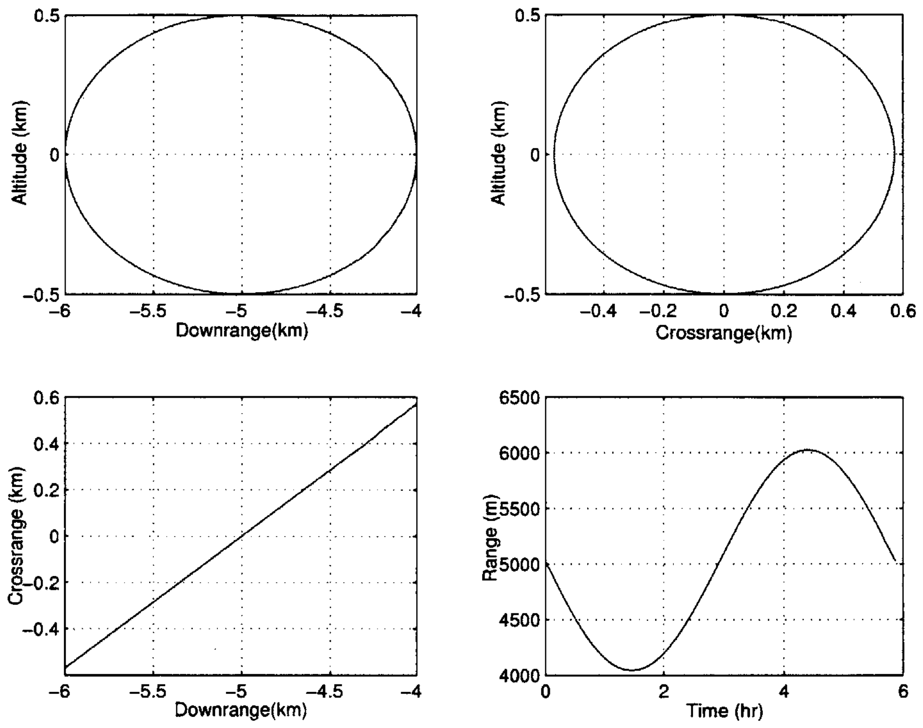
In terms of passive abort, football orbits are the most valuable of the 'building blocks.' Because a properly designed football orbit requires the OS and orbiter periods to be the same, the orbiter retains its net downrange spacing from the target. Given that the football was originally designed such that it did not intersect the target's position, there can be no collision in the nominal case. In 'real' rendezvous situations, when maneuvers cannot be performed to precisely match OS and orbiter energy, it is possible for the football orbit to slowly drift towards the target, eventually resulting in the possibility of a collision. A quick glance at the CW/Hill equations for relative motion (see Ref [17]) will again prove useful, however, as it is often noted that the equations for relative crosstrack motion are 'decoupled' from the downrange and altitude equations, and periodic in nature. Thus, it is possible to introduce a periodic crosstrack component to any football orbit without disturbing the original "in-plane" two-dimensional shape. This crosstrack component is shown in the lower-left subplot of Figure 3-4. When a small amount of crosstrack is introduced to the football orbit, the passive abort outlook becomes ideal.[1] Even if the football drifts towards the target, the chaser will circumnavigate the target in a slow helix, and the possibility of a strike becomes exceedingly small for reasonably sized orbits. Football orbits with a crosstrack component are called *offset football orbits* and are very desirable for the reasons listed above.

3.2.4 Glide Slope

The final basic building block of rendezvous trajectories that will be discussed is the *glide slope*. The glide slope is actually the result of a series of maneuvers, rather

[1]The crosstrack component must be properly phased, such that minimum crosstrack occurs at the maximum and minimum altitude, as shown in Figure 3-4.

Figure 3-4: Football orbit centered at 5km downrange

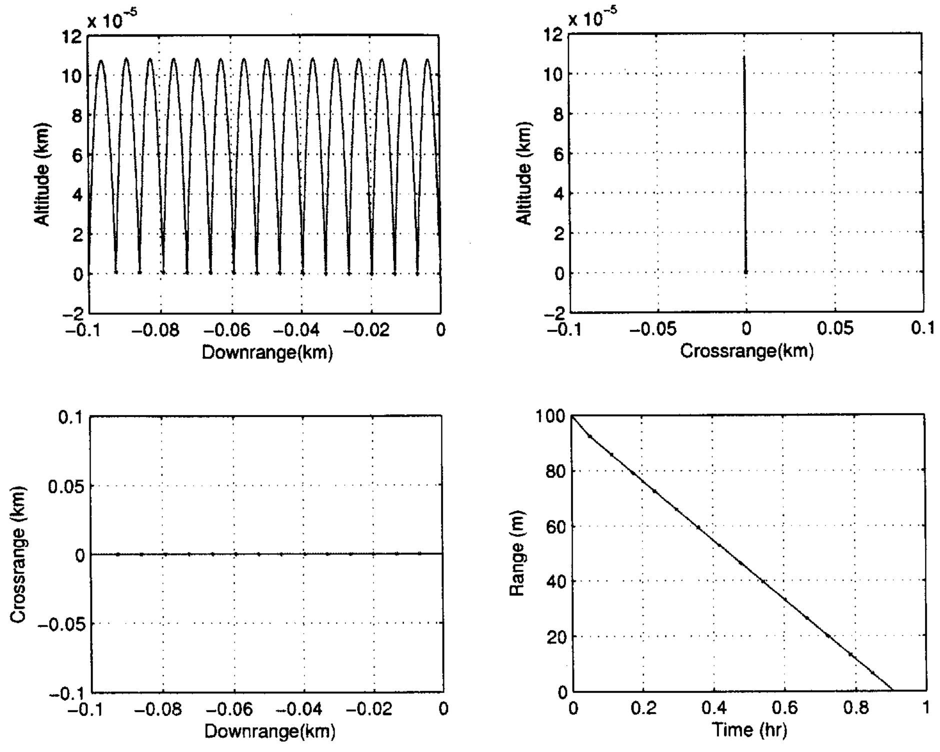


than an initialization maneuver and simple propagation as described for the hop, coelliptic, and football orbit trajectories. Figure 3-5 shows a sample zero-degree glide slope, where the dots indicate required maneuvers.

The primary function of a glide slope is to approximate a linear approach to the target, for purposes of observation or other reasons. On a ballistic approach, such as a hop or targeted Lambert maneuver, orbital dynamics dictate the approach trajectory, and no straight path exists that terminates at the target spacecraft. Coelliptic approaches become increasingly (and unalterably) slow as their altitude approaches zero, and there is no relative motion for a target initially stationary on the v-bar. The solution is to use a series of small maneuvers to force the chaser to follow the desired path along the v-bar or other linear approach; the simplest way to do this is to use a series of Lambert maneuvers to target the next position in the desired amount of time. In this way, a desired relative approach velocity can be obtained.

It is interesting to note that the glide slope approach on the v-bar has excellent passive abort properties, since active maneuvers are required to keep the orbiter

Figure 3-5: Zero-degree glide slope approach from 100m downrange



on the path to the target. If the chaser vehicle becomes disabled during the glide slope approach, the next burn is missed and the chaser enters a significant trajectory hop with negative altitude, which results in an increasing distance between the two vehicles. It is also interesting to note that, while the simplest way to implement the glide slope might be through Lambert targeting, the nominal maneuvers required to maintain the glide slope are all radial in direction. This is indicative of a need to redirect the chaser vehicle but not to change its energy or semi-major axis once the glide slope is established.[2]

3.3 Baseline Trajectory

By combining the basic elements described above and analyzing the resulting rendezvous trajectories using the Monte-Carlo tool, a great deal was learned about how

[2]The required radial burns can also be thought of as canceling a centrifugal force, caused by the chaser vehicle moving faster than the speed dictated by orbital mechanics for that semi-major axis.

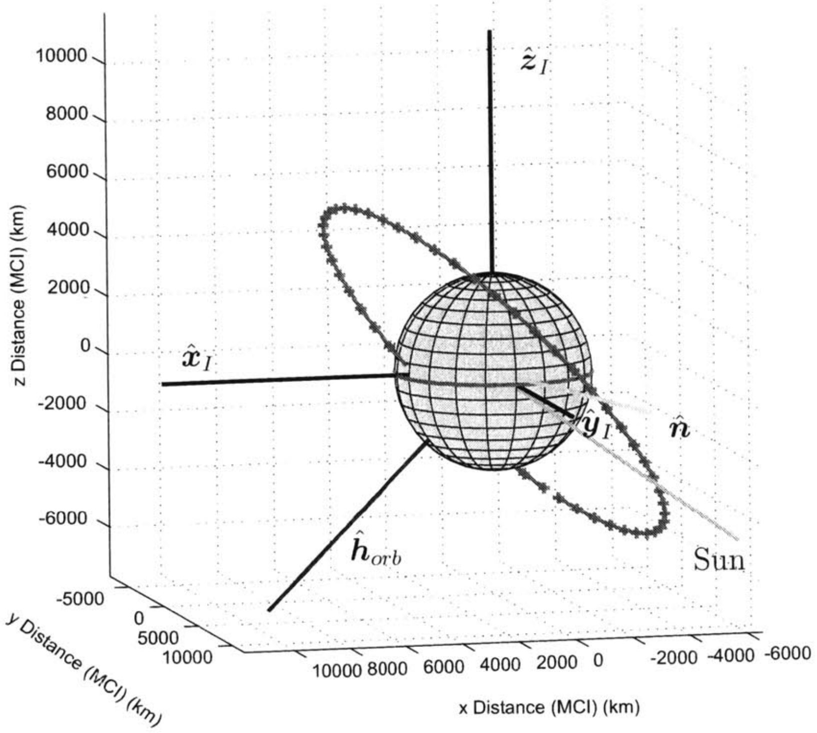
to (and how not to) approach the orbiting sample in the MTO orbit. Although this discussion is outside the scope of this document, it is valuable to present the final trajectory design and to justify its main features and less intuitive parts. This section will describe that baseline trajectory, first by presenting the details of the MTO orbit that the chaser vehicle is assumed to be in, and then by discussing the baseline rendezvous trajectory itself. The history of the trajectory design, although interesting in its own right, will not be presented.

3.3.1 Orbital Elements

The background of NASA's MSR program was discussed in Chapter 1 in some detail. It was noted in Chapter 2 that the MTO mission currently seems the most likely candidate for a Mars rendezvous demonstration, motivating the use of the MTO orbital elements in this analysis. In this section those orbital parameters will be presented with discussion of any important characteristics.

The raw orbital elements for the MTO mission can be found in Table 3.1. These elements have been dubbed the "4450 worst-case eclipse" elements by JPL, as they describe an orbit with an altitude of approximately 4450 km that spends a near maximum amount of time in eclipse. Figure 3-6 is a graphical representation of the MTO orbit, shown for visualization purposes. In this figure, the orbit and planet are to scale, making plain that this is a very large orbit. The orbital period is easily calculated from the semi-major axis as 351.7 min, or 5.86 hours. While the actual MSR mission will certainly have a smaller orbit, most likely with a period on the order of 2 hours, the MTO spacecraft is a communication satellite that benefits from a higher, slower orbit. The author has verified that the analysis presented here is equally applicable to a smaller orbit, given appropriate adjustments to the baseline trajectory. As the mission is currently scheduled to launch during the 2009 window, the epoch time used in this analysis is April 1, 2011, just after midnight on March 31. This date is tentatively scheduled for the rendezvous demonstration, allowing time for the transfer to Mars, MTO check-out and beginning operations, and favorable Mars-Earth communication geometry. The epoch time is summarized in Table 3.2.

Figure 3-6: Three-dimensional visualization of MTO orbit



Parameter	Value
a	$7847km$
e	0.0
i	131 deg
Ω	104 deg
ω	0 deg

Table 3.1: Orbital elements for MTO mission

Parameter	Value
Year	2011
Month	4
Day	1
Hour	0
Minute	0
Second	0

Table 3.2: Epoch time for MTO mission

It might be noted that only 5 orbital elements are provided in Table 3.1, while it takes 6 parameters to fully characterize the orbit. The mean anomaly (or, equivalently, the true anomaly) is considered a free parameter in this analysis, adjusted as required in a given reference trajectory to position the eclipse constraint at a desirable time during the run. It is therefore assumed that the target and chaser vehicle will be at the desired mean anomaly at epoch; while this is perhaps not strictly under the control of mission designers, it is noted that, in a real mission, the epoch time could be delayed until optimal conditions prevailed. The mean anomaly ultimately used for the baseline trajectory is 241 deg.

In the graphical orbit representation shown in Figure 3-6, the direction to the Sun is indicated by a vector that is in the orbit plane. It is this property of the orbit that makes it a "worst-case eclipse" orbit. If the orbit were at all inclined with respect to the Sun vector, the planet would present less area to shadow the target (or none at all for some orientations). In addition, this configuration of the Sun vector in the orbital plane results in frequent violation of the 'Sun angle[3] constraint' as described in Section 5.2.4, since the chaser's camera can easily be pointed at or near the Sun, depending on the actual chaser/target relative motion. Again, any inclination of the orbit with respect to the Sun vector would improve this situation. By using these worst-case orbital elements to generate the baseline trajectory in this thesis, it is assured that the actual performance will be equivalent or improved from this design.

3.3.2 Trajectory Constraints

The standard building blocks of a rendezvous trajectory have been described, along with the parameters of the MTO orbit under consideration. The following constraints have partially governed the shape of the baseline rendezvous trajectory:

Human Endurance The MTO rendezvous is a technology demonstration, and as such it will be very closely monitored by personnel on Earth. It is desirable to create a rendezvous trajectory that can be completed in a single shift, to

[3]That is, the angle the camera sees between the Sun and the target.

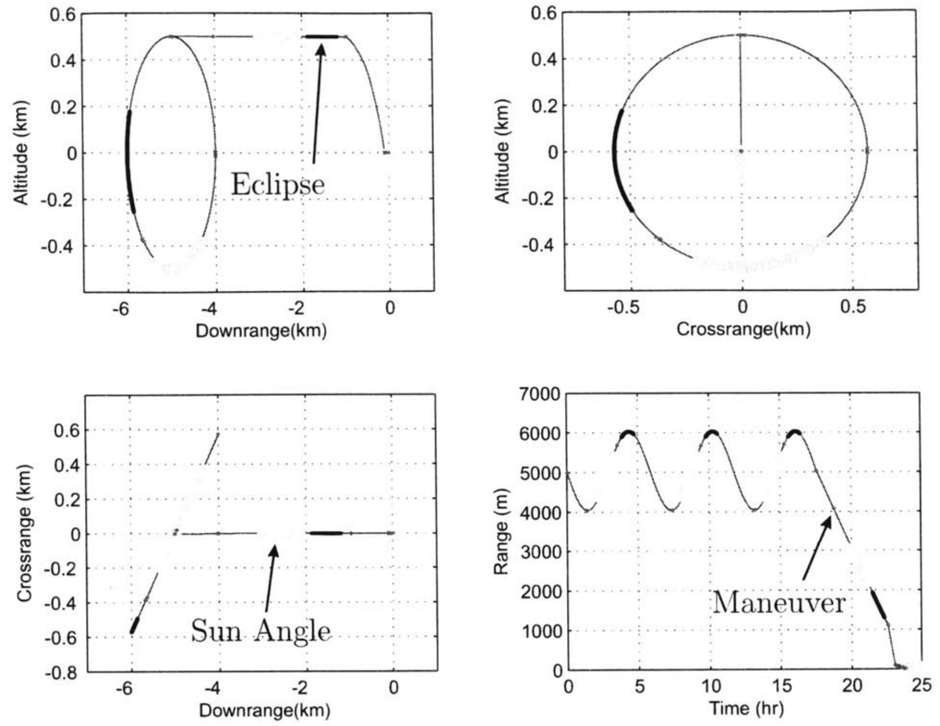
avoid a need to keep the operators alert for extended periods of time. A twelve-hour time constraint has been placed on the rendezvous operation, in hopes of reducing fatigue and therefore human error.

Sun Angle Constraint Section 5.2.4 describes the simulated camera model and its ability to calculate a *Sun angle constraint*. Physically, this constraint arises from lens and CCD abberations due to scattered sunlight that can occur if the camera is pointed too close to the Sun, and no meaningful measurements can be acquired at these times. Sun angle constraints are indicated on Figure 3-7 and must be taken into consideration when designing rendezvous trajectories.

Typically, when the chaser spacecraft is unable to take measurements, the flight computer propagates the state estimate until measurements are again available. When the first measurement is about to be made after the constraint is released, the spacecraft uses the current estimate of the target and chaser vehicles to point the camera at the target. If the lapse in measurements was too long, or if the state estimate was too far in error upon entering the constraint, the target might not be in the field of view of the camera for that first measurement. This can (and will) cause problems and delays in image processing, as the spacecraft must then be commanded to search for the target by taking an ordered series of pictures covering a larger field of view (hereafter called *mosaicing*). In this analysis, a 'mosaic mode' requirement is considered a failed run, and this situation was avoided to the extent possible. In particular, this means not performing maneuvers in (or immediately prior to) a Sun angle constraint, since maneuvers inject unwanted navigational uncertainty into the velocity state.

Target Shadowing Constraint Section 5.2.4 also describes the ability of the simulated camera model to calculate an *target shadowing constraint*. Physically, this constraint arises when the target passes into eclipse and is unobservable to the chaser. Aside from the source of the constraint, the consequences are identical to the Sun angle constraint, and an attempt is made to avoid maneuvers in (or immediately prior to) a shadowing constraint.

Figure 3-7: Baseline rendezvous trajectory: full trajectory



3.3.3 Reference Trajectory

Based on the limitations imposed by the constraints above, the baseline trajectory was designed and extensively tested. This trajectory will be used in the rest of this analysis and is shown in Figure 3-7. In the terms of Section 3.2 the baseline trajectory is comprised of the following pieces:

Holding Football Orbit The trajectory begins in a 2×1 km holding football orbit with about ± 600 m of out-of-plane motion. The trajectory calls for three revolutions around the football orbit, each taking one orbital period of 351.7 min to complete. The goal of this long holding pattern is to simulate the beginning of an actual mission, where the orbiter is likely to be established in a holding football orbit at a large downrange distance, taking measurements to refine the flight computer's estimate of the state. The holding football orbit allows for a check-out of the spacecraft hardware and navigational software at a safe distance, providing excellent passive abort capabilities as described in

Section 3.2.3, in case of hardware or software failure.

Coelliptic Approach After three revolutions around the football orbit, the chaser vehicle nominally enters a Δh = 500 m coelliptic to cover the large distance between the two vehicles. On a Δh = 500 m coelliptic, the chaser closes about 4700 m per orbit towards the target. The chaser is nominally on the coelliptic for a total of 300 min, or 5 hours. A large block of Sun angle and eclipse constraints are navigated at the end of the coelliptic, while the range to the target is still large (meaning that small estimated position errors result in small angle errors).

Lambert Fast Transfer The Δh = 500 m coelliptic is terminated at 1 km downrange by a Lambert maneuver targeted to the v-bar at 100 m downrange. The transfer time for this maneuver is 30 minutes, giving rise to the name 'fast transfer,' as opposed to a full half-hop that would take half a period. The reason for the fast transfer is two-fold. First, the transfer time is short to aid in completing the rendezvous before the next Sun constraint. Since the rest of the trajectory is along a glide slope where maneuvers are frequent, it is essential to make the rendezvous before the next constraint begins. The second reason for the fast transfer is because it creates a maneuver that is large compared to the spherical component of the maneuver execution error, resulting in a relatively accurate maneuver and a clear funneling of the dispersions as the chaser approaches the v-bar.

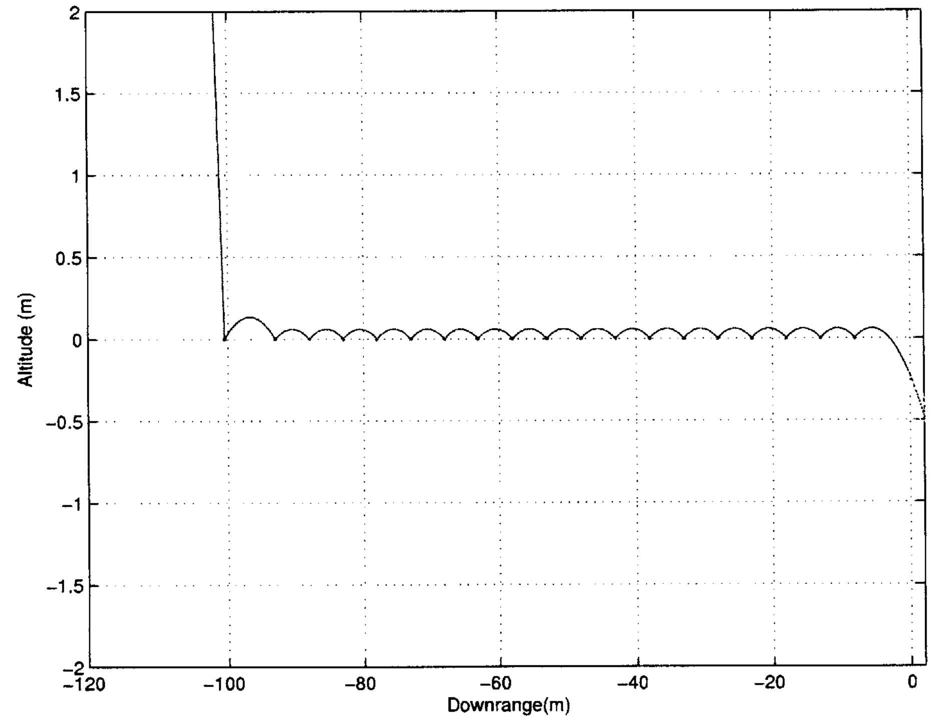
Glide Slope The final 100 m approach is performed on a zero-degree glide slope along the v-bar. Figure 3-8 shows a detailed view of of the glide slope, where the dots represent maneuvers and the circles represent a Sun angle constraint. When the fast transfer terminates on the v-bar the glide slope does not immediately begin in the expected uniform fashion, but there is a large "bounce" instead. Because the maneuver to terminate the fast transfer is fairly large, there is a large component of maneuver execution (compared to the rest of the glide slope maneuvers) error that injects velocity uncertainty into the navigation state.

The large bounces before the glide slope allow the filter to take measurements uninterrupted by maneuvers (and additional velocity uncertainty) and reduce the navigation error before beginning the final approach. In addition, the size of the bounce prevents the first maneuver on the true glide slope from being 'behind' the intersection of the fast transfer with the v-bar.[4]

Once on the uniform glide slope trajectory, the chaser closes the distance to the target at approximately 3 cm/s, a value estimated by the author to be reasonable for a final capture phase. Since the MTO rendezvous is a demonstration only, it will be noted that the trajectory passes through a point 3 m downrange from the target and then ceases to do maneuvers, falling away in a large negative altitude hop. The 3 m offset is a somewhat arbitrary target spot, designed to prove that the rendezvous was successful without actually making contact between the vehicles (the MTO orbiter will not even be equipped with capture hardware).

[4]It will be seen that this is not an issue for the Extended Kalman filter algorithms, but an important consideration for the Precomputed Gain system.

Figure 3-8: Baseline rendezvous trajectory: zero-degree glide slope



[This page intentionally left nearly blank.]

Chapter 4

Discrete Kalman Filter Design

In the words of William M. Lear [8], the *Kalman Filter* is "a computer algorithm that is used to process error corrupted measurement data." More technically, the Kalman filter is a recursive, optimal, linear, least-squares estimator that is used in many modern applications. By this long string of adjectives the following definitions are intended. *Recursive*: the Kalman filter does not require all previous measurements to be stored, as the information from these measurements is already incorporated into the filter. *Optimal*: describing the Kalman filter as 'optimal' means that it in fact optimizes some chosen performance criteria. The selected criteria varies from source to source but the Kalman filter remains optimal when evaluated with nearly any reasonable criteria. An example of a common optimality condition is to minimize the mean square estimation error. Put more plainly, optimality means that the Kalman filter "incorporates all information that can be provided to it [11]." *Linear*: the Kalman filter formulation assumes that the system in question is described by a linear model, and that the state estimate is obtained through linear operations on the measurement data. While most real systems are in fact nonlinear, the Kalman filter can still be applied with reasonable success by linearizing the model about some expected state.

The goal of this chapter is to introduce the "Standard" Discrete Kalman filter as a background for later chapters. It is not the intention of the author to supplant a whole shelf of books and papers that exist on estimation theory and the Kalman filter (see

Parameter	Description
$\boldsymbol{x}$	True state vector
$\hat{\boldsymbol{x}}$	Estimated state vector
$\tilde{\boldsymbol{x}}$	State error vector
$\boldsymbol{x}^*$	Nominal state vector
$\boldsymbol{\delta x}$	Perturbation of state vector
$\widehat{\boldsymbol{\delta x}}$	Estimated perturbed state
$\boldsymbol{z}$	True measurement vector
$\hat{\boldsymbol{z}}$	Estimated measurement vector
$\boldsymbol{\Phi}$	State transition matrix
$\boldsymbol{w}$	Process noise vector
$\boldsymbol{v}$	Measurement noise vector
$\boldsymbol{H}$	Measurement sensitivity matrix
$\boldsymbol{Q}$	Process noise covariance matrix
$\boldsymbol{R}$	Measurement noise covariance matrix
$\boldsymbol{P}$	Error covariance matrix
$\boldsymbol{K}$	Kalman gain (optimal gain) matrix

Table 4.1: Discrete Kalman filter notational conventions

Refs [4] [7] [8] [11] [16]). The goal is rather to provide a summary of the main theory and equations behind the basic Kalman filter to serve as a reference for this work. The theory in this chapter largely follows the derivation provided by Dr. William M. Lear in his report entitled "Kalman Filtering Techniques," as the author finds this to be an intuitive engineering perspective on the Kalman filter [8]. Lear's notation has been changed to match convention more frequently found in the literature. The discussion in this chapter will be kept generic; see Chapters 6 through 8 for details of the actual implementations, including filter states, measurement partials, etc.

4.1 Definitions

As mentioned above, this chapter will largely follow the Kalman filter derivation as presented by Lear [8]. The notational convention, however, will mostly be that of Gelb [4], Maybeck [11], and Siouris [16], as it is clearly prevalent in literature. Table 4.1 summarizes the variables that will appear in the following sections.

4.1.1 Measurements and State

In order to discuss the details of the discrete Kalman filter, two primary relationships must be defined. The first is the state vector and its dynamics. In general, defining the state vector is not a trivial task, and proper definition is essential to the smooth functioning of the filter. The key is to include in the state all information required to predict future measurements. Additionally, it may be necessary to include states that are known sources of error in the model, even if they are not directly used in the measurement calculation or propagation of the vehicle dynamics. The resulting state at time t_i is denoted $\boldsymbol{x}_i$. Secondly, it is important to know the measurement equation, or the expression to derive the measurement $\boldsymbol{z}_i$ from the state $\boldsymbol{x}_i$.

The state dynamics and measurement vectors can be formulated as follows for discrete time:

$$\boldsymbol{x}_{i+1} = \boldsymbol{g}(\boldsymbol{x}_i, t_i, \Delta t) + \boldsymbol{w}_i \tag{4.1}$$

$$\boldsymbol{z}_i = \boldsymbol{h}(\boldsymbol{x}_i, t_i) + \boldsymbol{v}_i \tag{4.2}$$

Here $\boldsymbol{w}_i$ is the process noise vector and $\boldsymbol{v}_i$ is the measurement noise vector. Note that the measurement equation $\boldsymbol{h}(\boldsymbol{x}_i, t_i)$ has been left as an undefined function of the state variables; if the measurements are a linear combination of the elements of the state, then Equation 4.3 holds and can be substituted into Equation 4.2.

$$\boldsymbol{h}(\boldsymbol{x}_i, t_i) = \boldsymbol{H}_i \boldsymbol{x}_i \tag{4.3}$$

The process noise vector, $\boldsymbol{w}_i$, is assumed to have zero mean and variance $\boldsymbol{Q}_i$, and is uncorrelated with the measurement noise vector. The matrix $\boldsymbol{Q}_i$ is called the process noise covariance matrix.

$$\begin{aligned} E[\boldsymbol{w}_i] &= \boldsymbol{0} \\ E[\boldsymbol{w}_i \boldsymbol{v}_i^T] &= \boldsymbol{0} \end{aligned} \tag{4.4}$$

$$\begin{aligned} E[\boldsymbol{w}_i \boldsymbol{w}_j^T] &= \mathbf{0} \qquad i \neq j \\ &= \boldsymbol{Q}_i \qquad i = j \end{aligned} \tag{4.5}$$

The measurement noise vector, $\boldsymbol{v}_i$, is assumed to have zero mean and variance $\boldsymbol{R}_i$, where the matrix $\boldsymbol{R}_i$ is called the measurement noise covariance matrix.

$$\begin{aligned} E[\boldsymbol{v}_i] &= \mathbf{0} \\ E[\boldsymbol{v}_i \boldsymbol{v}_j^T] &= \mathbf{0} \qquad i \neq j \\ &= \boldsymbol{R}_i \qquad i = j \end{aligned} \tag{4.6}$$

If the measurement noise is known to have a nonzero mean, then the bias part of that noise should be added to the state.

It will also be useful to define the state vector, $\boldsymbol{x}$, in terms of a nominal state, $\boldsymbol{x}^*$, and a perturbation from the nominal, $\boldsymbol{\delta x}$. Using this definition the state vector can be expressed:

$$\boldsymbol{x} = \boldsymbol{x}^* + \boldsymbol{\delta x} \tag{4.7}$$

Now let $\hat{\boldsymbol{x}}_i$ be an unbiased estimate of the state $\boldsymbol{x}$. Then the state estimate can also be expressed as a sum of the nominal state and an estimated perturbation, $\widehat{\boldsymbol{\delta x}}$:

$$\hat{\boldsymbol{x}} = \boldsymbol{x}^* + \widehat{\boldsymbol{\delta x}} \tag{4.8}$$

These equations will find use in the linearization process required for the derivations below.

4.1.2 Error Covariance

The state error vector, $\tilde{\boldsymbol{x}}_i$, can then be defined in terms of the state and state estimate, and, equivalently, the perturbed state and its estimate:

$$\tilde{\boldsymbol{x}}_i = \hat{\boldsymbol{x}}_i - \boldsymbol{x}_i \tag{4.9}$$

$$\tilde{\boldsymbol{x}}_i = \widehat{\boldsymbol{\delta x}}_i - \boldsymbol{\delta x}_i \tag{4.10}$$

These are fundamental equations that will be referenced frequently in the remainder of this chapter. It is assumed that the measurement noise vector and process noise vector are uncorrelated with the state error $\tilde{\boldsymbol{x}}_i$. That is:

$$E[\tilde{\boldsymbol{x}}_i\, \boldsymbol{v}_i^T] = \boldsymbol{0} \tag{4.11}$$

$$E[\tilde{\boldsymbol{x}}_i\, \boldsymbol{w}_i^T] = \boldsymbol{0} \tag{4.12}$$

The state error covariance matrix, $\boldsymbol{P}_i$, can now be defined using the expectation function:[1]

$$\boldsymbol{P}_i = E[\tilde{\boldsymbol{x}}_i\, \tilde{\boldsymbol{x}}_i^T] \tag{4.13}$$

In Equation 4.1, the state dynamics have been presented as a general function of state and time. No restrictions are thus placed on whether $\boldsymbol{g}(\boldsymbol{x}_i, t_i, \Delta t)$ is a linear or nonlinear function, perhaps suggesting that either could be used with equal success. In fact, formulating an estimator based on nonlinear dynamics is very difficult, and the most common approach is to linearize the dynamics. Significantly, there are two trajectories that are frequently used in this linearization: a nominal trajectory, and the filter's best estimate of the actual trajectory. This bifurcation will be explored in detail in Chapter 6.

4.2 Propagate State and Covariance

In the discrete Kalman filter it is necessary to propagate the state estimate and error covariance matrix forward from one measurement time to the next. This section describes the methods and equations that can be used to propagate the state and covariance matrix ahead one time step.

[1]The state error covariance matrix is symmetric and positive semi-definite, and it is essential that the matrix remains symmetric during numerical calculations. Often only the lower triangular portion of the covariance matrix is calculated, and the rest completed through symmetry; this saves processing time and memory and ensures that $\boldsymbol{P}$ remains symmetric. Alternatively, the 'poor-man' guarantee of symmetry can be coded as follows, after all calculations that alter the error covariance matrix:

$$\boldsymbol{P}_i = (\boldsymbol{P}_i + \boldsymbol{P}_i^T)/2$$

4.2.1 State

The discrete true state is propagated ahead one time step using the formula shown earlier in Equation 4.1.

$$\boldsymbol{x}_{i+1} = \boldsymbol{g}(\boldsymbol{x}_i, t_i, \Delta t) + \boldsymbol{w}_i \tag{4.1}$$

Again $\boldsymbol{w}_i$ is the process noise, and $\boldsymbol{g}(\boldsymbol{x}_i, t_i, \Delta t)$ is a function or algorithm that uses the state at time t_i and the time step, Δt, to compute the state at the next time, t_{i+1}. If the state dynamics are linear then Equation 4.14 describes the transition from x_i to x_{i+1}, where $\boldsymbol{\Phi}_{i+1,i}$ is appropriately named the *state transition matrix* from t_i to t_{i+1}.

$$\boldsymbol{g}(\boldsymbol{x}_i, t_i, \Delta t) = \boldsymbol{\Phi}_{i+1,i}\,\boldsymbol{x}_i \tag{4.14}$$

For most real systems, the state dynamics are *not* linear and a numerical integrator must be used to propagate the state.[2] In this case $\boldsymbol{g}(\boldsymbol{x}_i, t_i, \Delta t)$ is truly an algorithm, and not simply an equation. Process noise is added to the state propagation because it is very unlikely that the mathematical model of the system is a perfectly accurate representation.

The state estimate is propagated in a similar fashion, where now the algorithm $\boldsymbol{g}(\boldsymbol{x}_i, t_i, \Delta t)$ works on the state estimate instead of the true state, and the noice term $\boldsymbol{w}$ is not included since $\hat{\boldsymbol{w}} = 0$.

$$\hat{\boldsymbol{x}}_{i+1} = \boldsymbol{g}(\hat{\boldsymbol{x}}_i, t_i, \Delta t) \tag{4.15}$$

4.2.2 Error Covariance

The error covariance matrix must also be propagated forward in time between measurements. While the final propagation equation is relatively straightforward, the derivation is somewhat longer than for the state propagation equation. By substituting Equation 4.7 into Equations 4.1 and 4.15, the true and estimated state updates

[2]Lear points out that Kalman filters require a *self-starting* integrator, or one that only uses the current state to propagate ahead, and does not need any past states [8].

can be found in terms of the nominal state and the perturbed state vector:

$$\begin{aligned} \boldsymbol{x}_{i+1} &= \boldsymbol{g}(\boldsymbol{x}_i^* + \boldsymbol{\delta x}_i, t_i, \Delta t) + \boldsymbol{w}_i & (4.16) \\ \hat{\boldsymbol{x}}_{i+1} &= \boldsymbol{g}(\boldsymbol{x}_i^* + \hat{\boldsymbol{\delta x}}_i, t_i, \Delta t) & (4.17) \end{aligned}$$

If the perturbation is indeed small, a first order Taylor series expansion can be used to separate these equations into a more useful form:

$$\begin{aligned} \boldsymbol{x}_{i+1} &= \boldsymbol{g}(\boldsymbol{x}_i^*, t_i, \Delta t) + \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_i}\right|_{\boldsymbol{x}_i^*} \boldsymbol{\delta x}_i + \emptyset(\boldsymbol{\delta x}_i^2) + \boldsymbol{w}_i & (4.18) \\ \hat{\boldsymbol{x}}_{i+1} &= \boldsymbol{g}(\boldsymbol{x}_i^*, t_i, \Delta t) + \left.\frac{\partial \boldsymbol{g}}{\partial \hat{\boldsymbol{x}}_i}\right|_{\boldsymbol{x}_i^*} \hat{\boldsymbol{\delta x}}_i + \emptyset(\hat{\boldsymbol{\delta x}}_i^2) & (4.19) \end{aligned}$$

Subtracting Equation 4.18 from Equation 4.19 and utilizing Equation 4.9, the following form is obtained for the propagation of the state error (neglecting terms of order $\boldsymbol{\delta x}_i^2$ or higher):

$$\begin{aligned} \hat{\boldsymbol{x}}_{i+1} - \boldsymbol{x}_{i+1} &= \boldsymbol{\Phi}_{i+1,i} (\hat{\boldsymbol{\delta x}}_i - \boldsymbol{\delta x}_i) - \boldsymbol{w}_i \\ \tilde{\boldsymbol{x}}_{i+1} &= \boldsymbol{\Phi}_{i+1,i} \tilde{\boldsymbol{x}}_i - \boldsymbol{w}_i & (4.20) \end{aligned}$$

Here $\boldsymbol{\Phi}_{i+1,i}$ is the state transition matrix as defined in Equation 4.21, a generalization of Equation 4.14 for linear or nonlinear dynamics when the state perturbation is small.

$$\boldsymbol{\Phi}_{i+1,i} = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_i}\right|_{\boldsymbol{x}_i^*} = \left.\frac{\partial \boldsymbol{g}}{\partial \hat{\boldsymbol{x}}_i}\right|_{\boldsymbol{x}_i^*} = \left.\frac{\partial \boldsymbol{x}_{i+1}}{\partial \boldsymbol{x}_i}\right|_{\boldsymbol{x}_i^*} \qquad (4.21)$$

Note that if the state dynamics are linear as in Equation 4.14, then there are no derivatives of the dynamics beyond the first order, and the Taylor series approximation in Equation 4.18 is exact. Otherwise it is dependent on the true and estimated state perturbations being small. By using the definition of the error covariance matrix from Equation 4.13, an equation for the propagated covariance matrix can be formed:

$$\begin{aligned}
\boldsymbol{P}_{i+1} &= E\left[\tilde{\boldsymbol{x}}_{i+1}\,\tilde{\boldsymbol{x}}_{i+1}^T\right] \\
&= E\left[(\boldsymbol{\Phi}_{i+1,i}\,\tilde{\boldsymbol{x}}_i - \boldsymbol{w}_i)(\boldsymbol{\Phi}_{i+1,i}\,\tilde{\boldsymbol{x}}_i - \boldsymbol{w}_i)^T\right] \\
&= E\left[\boldsymbol{\Phi}_{i+1,i}\,\tilde{\boldsymbol{x}}_i\,\tilde{\boldsymbol{x}}_i^T\boldsymbol{\Phi}_{i+1,i}^T - \boldsymbol{w}_i\,\tilde{\boldsymbol{x}}_i^T\boldsymbol{\Phi}_{i+1,i}^T - \boldsymbol{\Phi}_{i+1,i}\,\tilde{\boldsymbol{x}}_i\,\boldsymbol{w}_i^T + \boldsymbol{w}_i\boldsymbol{w}_i^T\right]
\end{aligned} \tag{4.22}$$

Since the state error and process noise are uncorrelated as shown above in Equation 4.12, the middle terms drop out leaving the following simplified form:

$$\boldsymbol{P}_{i+1} = \boldsymbol{\Phi}_{i+1,i}\,\boldsymbol{P}_i\,\boldsymbol{\Phi}_{i+1,i}^T + \boldsymbol{Q}_i \tag{4.23}$$

This is the final error covariance matrix propagation equation.

4.3 Measurement Update

The key piece that is still missing from this Kalman filter algorithm is updating the state and the covariance when measurement data is available. As was mentioned in the introduction to this chapter, the Kalman filter does this in an *optimal* fashion, so a fair amount of effort will be expended to find the optimal solution to the update.

The only external information that is available to make a state update are the measurements. Thus, it seems reasonable to try to improve the state estimate by using feedback from the measurement data, and the measurement update equation will be formulated as follows:

$$\hat{\boldsymbol{x}}_i^+ = \hat{\boldsymbol{x}}_i^- + \boldsymbol{K}_i(\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i) \tag{4.24}$$

In words, Equation 4.24 states that the new estimate of the state will be the old estimate of the state, plus a correction directly related to the difference between the actual measurement data and what the filter expected that data to be. The matrix $\boldsymbol{K}_i$ is a weighting matrix that specifies how much weight will be given to each measurement, with respect to each state. This gain matrix is currently unspecified, but ultimately the gain will be found to optimize the state update. The estimated measurement can be found by operating the measurement equation on the state estimate:

$$\hat{\boldsymbol{z}}_i = \boldsymbol{h}(\hat{\boldsymbol{x}}_i, t_i) \tag{4.25}$$

By substituting Equations 4.7 and 4.8 into Equations 4.2 and 4.25, respectively, the true and estimated measurements can be found in terms of the nominal state and the perturbed state vector:

$$\begin{aligned} \boldsymbol{z}_i &= \boldsymbol{h}(\boldsymbol{x}_i^* + \boldsymbol{\delta x}_i, t_i) + \boldsymbol{v}_i & (4.26) \\ \hat{\boldsymbol{z}}_i &= \boldsymbol{h}(\boldsymbol{x}_i^* + \hat{\boldsymbol{\delta x}}_i, t_i) & (4.27) \end{aligned}$$

If the perturbation is indeed small then these equations can be expanded in a first order Taylor series, resulting in the following:

$$\begin{aligned} \boldsymbol{z}_i &= \boldsymbol{h}(\boldsymbol{x}_i^*, t_i) + \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_i}\right|_{\boldsymbol{x}_i^*} \boldsymbol{\delta x}_i + \emptyset(\boldsymbol{\delta x}_i^2) + \boldsymbol{v}_i & (4.28) \\ \hat{\boldsymbol{z}}_i &= \boldsymbol{h}(\boldsymbol{x}_i^*, t_i) + \left.\frac{\partial \boldsymbol{h}}{\partial \hat{\boldsymbol{x}}_i}\right|_{\boldsymbol{x}_i^*} \hat{\boldsymbol{\delta x}}_i + \emptyset(\hat{\boldsymbol{\delta x}}_i^2) & (4.29) \end{aligned}$$

Subtracting Equation 4.29 from Equation 4.28 and rearranging results in the following (neglecting terms of order $\boldsymbol{\delta x}_i^2$ or higher):

$$\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i = -\boldsymbol{H}_i \tilde{\boldsymbol{x}}_i + \boldsymbol{v}_i \tag{4.30}$$

Here the partial derivative from Equation 4.28 is defined to be the *measurement sensitivity matrix*, or the measurement partials matrix, $\boldsymbol{H}_i$:

$$\left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_i}\right|_{\boldsymbol{x}_i^*} = \boldsymbol{H}_i \tag{4.31}$$

Note that if the measurements are a linear combination of the elements of the state, as in Equation 4.3, then there are no higher order derivatives of the measurement equation than Equation 4.31 and the first order Taylor expansion in Equation 4.28

is exact. By multiple substitution of Equation 4.9 into Equation 4.24 it can also be shown that:

$$\tilde{\boldsymbol{x}}_i^+ = \tilde{\boldsymbol{x}}_i^- + \boldsymbol{K}_i(\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i) \tag{4.32}$$

Then by substituting Equation 4.30 into Equation 4.32 and rearranging, the error update equation is revealed:

$$\tilde{\boldsymbol{x}}_i^+ = (\boldsymbol{I} - \boldsymbol{K}_i\boldsymbol{H}_i)\tilde{\boldsymbol{x}}_i^- + \boldsymbol{K}_i\boldsymbol{v}_i \tag{4.33}$$

Now again by using the definition of the error covariance matrix from Equation 4.13, the covariance update equation can be found:

$$\begin{aligned} \boldsymbol{P}_i^+ &= E\left[\tilde{\boldsymbol{x}}_i^+\tilde{\boldsymbol{x}}_i^{+T}\right] \qquad (4.34)\\ &= E\left[\left((\boldsymbol{I} - \boldsymbol{K}_i\boldsymbol{H}_i)\tilde{\boldsymbol{x}}_i^- + \boldsymbol{K}_i\boldsymbol{v}_i\right)\left((\boldsymbol{I} - \boldsymbol{K}_i\boldsymbol{H}_i)\tilde{\boldsymbol{x}}_i^- + \boldsymbol{K}_i\boldsymbol{v}_i\right)^T\right] \end{aligned}$$

Expanding and simplifying the expectation function, and remembering that the state error and measurement noise are uncorrelated as shown in Equation 4.11, gives the final covariance matrix update equation for an arbitrary measurement gain, $\boldsymbol{K}_i$:

$$\boldsymbol{P}_i^+ = (\boldsymbol{I} - \boldsymbol{K}_i\boldsymbol{H}_i)\boldsymbol{P}_i^-(\boldsymbol{I} - \boldsymbol{K}_i\boldsymbol{H}_i)^T + \boldsymbol{K}_i\boldsymbol{R}_i\boldsymbol{K}_i^T \tag{4.35}$$

Equation 4.35 is only exact if the measurements are a linear function of the state. Otherwise, its validity is dependent on the assumption that the true and estimated perturbed states are small. This 'linearity' assumption is important to note, as it can be stretched in some implementations of the Kalman filter.

At this point, a covariance update equation has been found for an arbitrary $\boldsymbol{K}_i$, the measurement gain. The optimal gain can be found by choosing it such that the updated state estimate will be a minimum variance estimate of the state. The goal of this approach is then to minimize the quadratic form associated with the error covariance matrix. That is, to minimize:

$$s = \boldsymbol{a}^T \boldsymbol{P}_i^+ \boldsymbol{a} \tag{4.36}$$

Here $\boldsymbol{a}$ is an arbitrary non-zero vector with the length of the state vector. By allowing $\boldsymbol{a}$ to be arbitrary, the scalar s will be minimized for any symmetric weighting of the elements of $\boldsymbol{P}$.[3] A simple way to minimize s by taking its variation, δs. That is, for $s = \boldsymbol{a}^T \boldsymbol{P} \boldsymbol{a} = f(\boldsymbol{K})$, note that:

$$s + \delta s = f(\boldsymbol{K} + \delta \boldsymbol{K}) \tag{4.37}$$

Substituting Equation 4.35 into Equation 4.36 and then expanding the variation as suggested by Equation 4.37, the following terms result after some algebraic manipulation:

$$\begin{aligned} s + \delta s = \boldsymbol{a}^T \Big\{ & \big[(\boldsymbol{I} - \boldsymbol{K}_i \boldsymbol{H}_i) \boldsymbol{P}_i^- (\boldsymbol{I} - \boldsymbol{K}_i \boldsymbol{H}_i)^T + \boldsymbol{K}_i \boldsymbol{R}_i \boldsymbol{K}_i^T\big] \\ & + \big[- \delta \boldsymbol{K}_i \boldsymbol{H}_i \boldsymbol{P}_i^- (\boldsymbol{I} - \boldsymbol{K}_i \boldsymbol{H}_i)^T - (\boldsymbol{I} - \boldsymbol{K}_i \boldsymbol{H}_i) \boldsymbol{P}_i^- \boldsymbol{H}_i^T \delta \boldsymbol{K}_i^T \\ & + \delta \boldsymbol{K}_i \boldsymbol{R}_i \boldsymbol{K}_i^T + \boldsymbol{K}_i \boldsymbol{R}_i \delta \boldsymbol{K}_i^T \big] + \big[\delta \boldsymbol{K}_i \boldsymbol{H}_i \boldsymbol{P}_i^- \boldsymbol{H}_i^T \delta \boldsymbol{K}_i^T + \delta \boldsymbol{K}_i \boldsymbol{R}_i \delta \boldsymbol{K}_i^T\big] \Big\} \boldsymbol{a} \end{aligned} \tag{4.38}$$

Note that the two terms in the last set of brackets are pre- and post-multiplied by $\delta \boldsymbol{K}_i$, resulting in a second order term that can be neglected for small variations in $\boldsymbol{K}_i$. Noting also that the first term in brackets can be simplified using Equation 4.35, a little more algebra reduces this equation to:

$$\begin{aligned} s + \delta s = \boldsymbol{a}^T \big[& \boldsymbol{P}_i^+ - \delta \boldsymbol{K}_i \boldsymbol{H}_i \boldsymbol{P}_i^- (\boldsymbol{I} - \boldsymbol{K}_i \boldsymbol{H}_i)^T \\ & - (\boldsymbol{I} - \boldsymbol{K}_i \boldsymbol{H}_i) \boldsymbol{P}_i^- \boldsymbol{H}_i^T \delta \boldsymbol{K}_i^T + \delta \boldsymbol{K}_i \boldsymbol{R}_i \boldsymbol{K}_i^T + \boldsymbol{K}_i \boldsymbol{R}_i \delta \boldsymbol{K}_i^T \big] \boldsymbol{a} \end{aligned} \tag{4.39}$$

[3]A common approach in the literature is to minimize the sum of mean-square state errors by minimizing the cost function $J_i = Trace\,[\boldsymbol{P}]$. The approach used by Lear is a more general formulation, and it can be shown that for a particular vector $\boldsymbol{a}$, $s = \boldsymbol{a}^T \boldsymbol{P} \boldsymbol{a} = Trace\,[\boldsymbol{P}]$. Specifically, let $\boldsymbol{P}'$ be a diagonalized form of the covariance matrix, $\boldsymbol{P}' = \boldsymbol{T}\boldsymbol{P}\boldsymbol{T}^T$. Then in this frame, the appropriate weighting vector $\boldsymbol{a}'$ is a vector of ones, such that $Trace\,[\boldsymbol{P}'] = \boldsymbol{a}'^T \boldsymbol{T}\boldsymbol{P}\boldsymbol{T}^T \boldsymbol{a}'$. Knowing that $Trace\,[\boldsymbol{P}'] = Trace\,[\boldsymbol{P}]$, it is clear that the desired weighting vector is $\boldsymbol{a} = \boldsymbol{T}^T \boldsymbol{a}'$.

Now, making use of Equation 4.36, the expression for s can be removed, leaving only the variation, δs. This can be further simplified by a little factorization, and then noticing that the two resultant terms are scalars (and thus the transpose is inconsequential).

$$\begin{aligned}\delta s &= \boldsymbol{a}^T(-\delta \boldsymbol{K}\boldsymbol{H}\boldsymbol{P}(\boldsymbol{I}-\boldsymbol{K}\boldsymbol{H})^T \\ &\qquad -(\boldsymbol{I}-\boldsymbol{K}\boldsymbol{H})\boldsymbol{P}\boldsymbol{H}^T\delta\boldsymbol{K}^T+\delta\boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^T+\boldsymbol{K}\boldsymbol{R}\delta\boldsymbol{K}^T)\boldsymbol{a} \\ &= \boldsymbol{a}^T(-(\boldsymbol{I}-\boldsymbol{K}\boldsymbol{H})\boldsymbol{P}\boldsymbol{H}^T+\boldsymbol{K}\boldsymbol{R})\delta\boldsymbol{K}^T\boldsymbol{a} \\ &\qquad +(\boldsymbol{a}^T(-(\boldsymbol{I}-\boldsymbol{K}\boldsymbol{H})\boldsymbol{P}\boldsymbol{H}^T+\boldsymbol{K}\boldsymbol{R})\delta\boldsymbol{K}^T\boldsymbol{a})^T \\ &= 2\boldsymbol{a}^T\left[(-(\boldsymbol{I}-\boldsymbol{K}\boldsymbol{H})\boldsymbol{P}\boldsymbol{H}^T+\boldsymbol{K}\boldsymbol{R})\delta\boldsymbol{K}^T\right]\boldsymbol{a} \qquad (4.40)\end{aligned}$$

Thus s will be minimized by setting interior terms of Equation 4.40 to zero. Rearranging the inner term finally reveals the optimal gain, $\boldsymbol{K}_i$:

$$\boldsymbol{K}_i = \boldsymbol{P}_i\boldsymbol{H}_i^T(\boldsymbol{H}_i\boldsymbol{P}_i\boldsymbol{H}_i^T+\boldsymbol{R}_i)^{-1} \qquad (4.41)$$

Interestingly, the optimal gain from Equation 4.41 can be substituted back into Equation 4.35 to get a more streamlined form of the covariance update equation:

$$\boldsymbol{P}_i^+ = (\boldsymbol{I}-\boldsymbol{K}_i\boldsymbol{H}_i)\boldsymbol{P}_i^- \qquad (4.42)$$

Using the covariance update equation from Equation 4.42 and the optimal gain solution from Equation 4.41 in the state update equation (Equation 4.24), a full update can be performed with measurement data. These equations are summarized in Section 4.4.

4.4 Summary

The sections above outline the theory and equations behind the basic discrete Kalman filter. This section is provided to highlight the key equations used in the filter loop.

$\hat{\boldsymbol{x}}_{i+1} = \boldsymbol{g}(\hat{\boldsymbol{x}}_i, t_i, \Delta t)$	(4.15)
$\boldsymbol{g}(\boldsymbol{x}_i, t_i, \Delta t) = \boldsymbol{\Phi}_{i+1,i}\, \boldsymbol{x}_i$	(4.14)
$\boldsymbol{P}_{i+1} = \boldsymbol{\Phi}_{i+1,i}\, \boldsymbol{P}_i\, \boldsymbol{\Phi}_{i+1,i}^T + \boldsymbol{Q}_i$	(4.23)
$\boldsymbol{\Phi}_{i+1,i} = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_i}\right\rvert_{\boldsymbol{x}_i^*} = \left.\frac{\partial \boldsymbol{x}_{i+1}}{\partial \boldsymbol{x}_i}\right\rvert_{\boldsymbol{x}_i^*}$	(4.21)

Table 4.2: Discrete Kalman filter propagation equations

Table 4.2 lists the main equations involved in propagating the state estimate and covariance matrix between measurement times. The state estimate is propagated using Equation 4.15, where typically the function $\boldsymbol{g}(\boldsymbol{x}_i, t_i, \Delta t)$ is a self-starting numerical integrator, but in the case of linear dynamics it can simplify to Equation 4.14. The covariance is propagated using Equation 4.23, where the state transition matrix $\boldsymbol{\Phi}_{i+1,i}$ is calculated according to Equation 4.21. If the state dynamics are linear, then the state transition matrix comes directly from the dynamics.

Table 4.3 lists the main equations involved in updating the state estimate and covariance matrix with a measurement. The state estimate is updated according to Equation 4.24, where the optimal gain is computed according to Equation 4.41. The estimated measurement, $\hat{\boldsymbol{z}}_i$, is calculated by operating the measurement equation on the state estimate as in Equation 4.25. If the measurements are a linear function of the states, then Equation 4.3 holds as the measurement equation. Finally, the covariance update is calculated according to Equation 4.42.

$\hat{\boldsymbol{x}}_i^+ = \hat{\boldsymbol{x}}_i^- + \boldsymbol{K}_i(\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i)$	(4.24)
$\hat{\boldsymbol{z}}_i = \boldsymbol{h}(\hat{\boldsymbol{x}}_i, t_i)$	(4.25)
$\boldsymbol{h}(\boldsymbol{x}_i, t_i) = \boldsymbol{H}_i\boldsymbol{x}_i$	(4.3)
$\boldsymbol{K}_i = \boldsymbol{P}_i\boldsymbol{H}_i^T(\boldsymbol{H}_i\boldsymbol{P}_i\boldsymbol{H}_i^T + \boldsymbol{R}_i)^{-1}$	(4.41)
$\boldsymbol{P}_i^+ = (\boldsymbol{I} - \boldsymbol{K}_i\boldsymbol{H}_i)\boldsymbol{P}_i^-$	(4.42)

Table 4.3: Discrete Kalman filter measurement update equations

Chapter 5

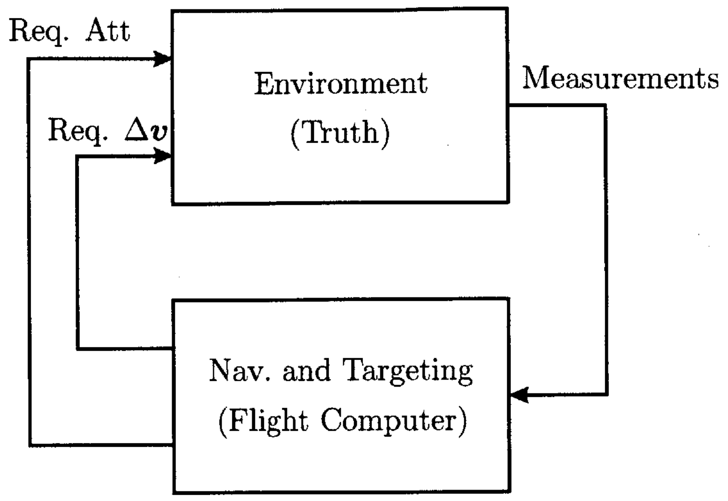
Monte-Carlo Environment Simulator

Comparing two sets of navigation and targeting algorithms in simulation requires an *environment* model that can be used to propagate the true state of the vehicles and derive "actual" instrument measurements. In general, an environment model simulates a real vehicle's response to inputs such as: gravitational forces, atmospheric drag, solar radiation pressure, commanded changes in attitude, commanded maneuver executions, and requests for sensor data. Accordingly, an environment simulator was created for this research that contained an attitude model, a maneuver model, a dynamics model, and a sensor model. For this work, the sensor studied was an optical camera capable of making a range measurement to a known target, by way of subtended angle, as well as azimuth and elevation angles from the boresight. This chapter describes the details of the Monte-Carlo environment simulator, and how it was configured to mate to the two sets of navigation and targeting algorithms under consideration.

5.1 Overview and Interface

At the highest level, the Monte-Carlo simulator is quite simple, as is demonstrated by the flow diagram shown in Figure 5-1. The environment model integrates the

Figure 5-1: Top-level flow diagram for Monte-Carlo simulator

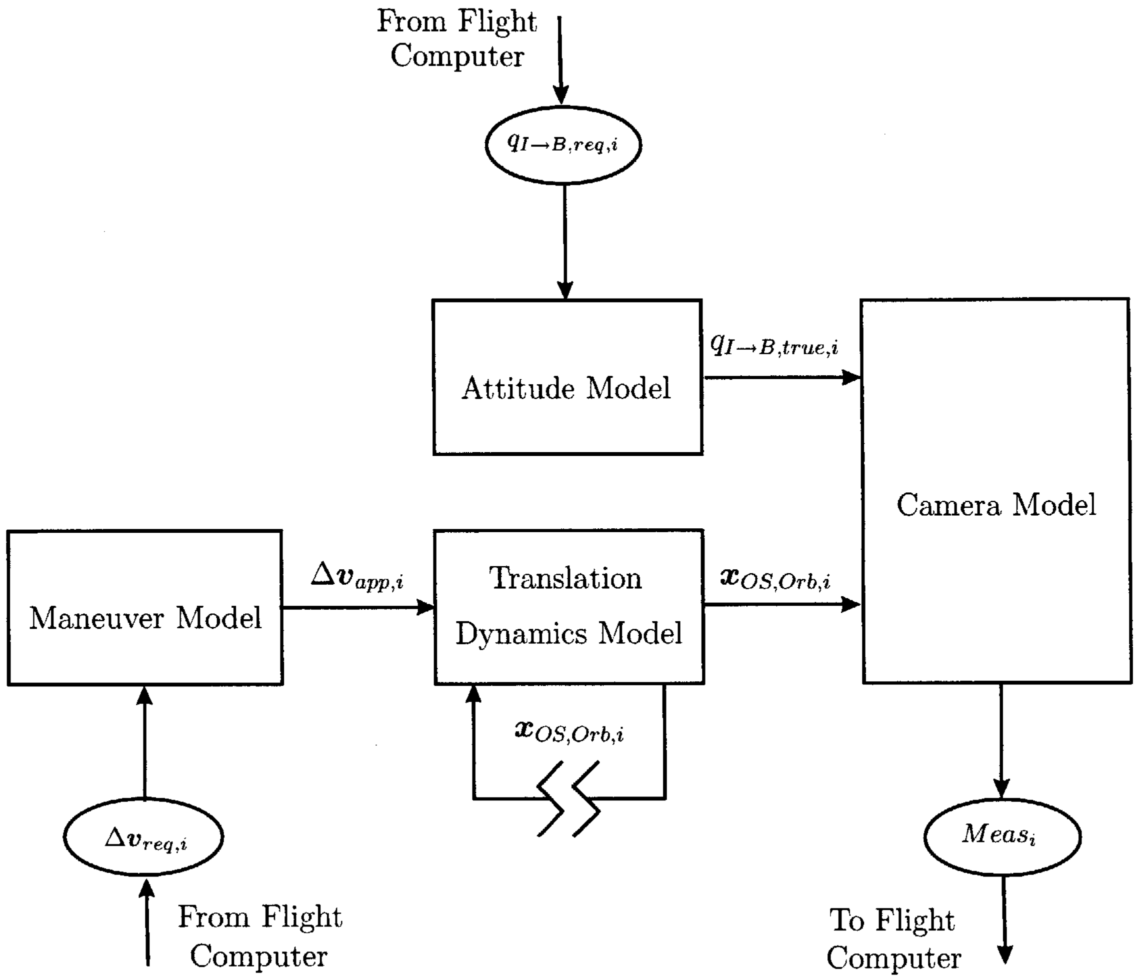

Req. Att
Req. $\Delta \boldsymbol{v}$
Environment (Truth)
Measurements
Nav. and Targeting (Flight Computer)

two-body equation of motion for both the target and chaser vehicles, incorporating required maneuvers for the chaser, and generates simulated measurements based on vehicle positions and the chaser vehicle's attitude. The flight computer accepts the simulated measurements and uses the onboard filter algorithm (Precomputed Gain , extended, or other) to calculate an estimate of the vehicle states. From the state estimate the flight computer can then compute the desired pointing attitude, and the change in velocity required to keep the rendezvous on course. Each set of navigation and targeting algorithms might make these computations in different ways (for example, the Precomputed Gain implementation uses maneuver gain matrices to calculate the required $\Delta \boldsymbol{v}$, while the Extended implementation uses onboard targeting algorithms), however the entire calculation can be contained in the "flight" module and thus be easily traded. The flight computer outputs the required spacecraft attitude and required $\Delta \boldsymbol{v}$, which the environment model then uses for the next time step.

5.2 Environment

Figure 5-2 shows the flow of the environment module and its basic components. It can be seen that the environment module is divided into four primary models:

Figure 5-2: Flow diagram for environment module



the attitude model, the dynamics model, the maneuvering model, and the camera model. The following sections describe these models and the constraints and sources of variability that exist in each.

5.2.1 Attitude Model

The rendezvous simulator is a pseudo six degree of freedom (6-DOF) system for the chaser, meaning that the translation and rotation for the chaser are fully modeled (6-DOF) but there are no attitude dynamics.[1] For this reason, the attitude is consid-

[1]For this study, the target is assumed to be an inert uniform sphere with center of mass at its center. Thus, it is not necessary to model target attitude, as it has no effect on the geometry, and the simulation is 3-DOF for the target.

ered separately from the vehicle dynamics. The function of the attitude model is to accept the required attitude from the flight computer and simulate the uncertainty that is always associated with space vehicle attitude knowledge. With no attitude dynamics, the assumption is that the required attitude, $q_{I \to B,req}$, is achieved to within the accuracy of the attitude sensors and the attitude determination system.

The chaser vehicle's required attitude is passed to the environment model as a quaternion describing the required inertial to body transformation for boresight pointing. The inertial attitude error is applied directly to this "required" quaternion to form the "true" quaternion before a measurement is simulated. The error is generated using a First Order Markov Process with variance ${\sigma_A}^2$ and time constant τ_A to create a vector of three small rotation angles, $\boldsymbol{\Delta\phi}$. This vector is then converted into a quaternion and applied to the required inertial attitude, resulting in the *true* spacecraft attitude as shown in Equations 5.1 and 5.2.

$$q_{I \to B,true} = q_{I \to B,req} \otimes q_{B,req \to B,true} \tag{5.1}$$

$$q_{B,req \to B,true} = f_{vector \to quat}(\Delta\phi_{req \to true}) \tag{5.2}$$

Here $\boldsymbol{\Delta\phi}_{req \to true}$ is the vector of three angles generated by the Markov process, and $f_{vector \to quat}$ is a function that derives a quaternion from a vector of small rotation angles, as shown below.

$$q = \begin{bmatrix} \hat{\boldsymbol{u}} \sin \frac{\psi}{2} \\ \\ \cos \frac{\psi}{2} \end{bmatrix} \approx \begin{bmatrix} \hat{\boldsymbol{u}} \frac{\psi}{2} \\ \\ 1 \end{bmatrix} \tag{5.3}$$

The left half of Equation 5.3 shows the description of a quaternion, q, in terms of an Euler axis, $\boldsymbol{u}$, and an Euler angle, ψ. For small ψ, the sine and cosine can be reduced using the small angle approximation, and the quaternion is simplified as shown in the right half of the equation. The vector of three small rotation angles, $\boldsymbol{\Delta\phi}$, generated by the Markov Process can then be considered the product of a small Euler angle and a random unit Euler axis. The conversion $q_{b,req\rightarrow b,true} = f_{vector\rightarrow quat}(\boldsymbol{\Delta\phi})$ can be written simply as shown in Equation 5.4 below.

$$\begin{aligned} q' &\approx \begin{bmatrix} \frac{\boldsymbol{\Delta\phi}}{2} \\ \\ 1 \end{bmatrix} \\ q_{b,req\rightarrow b,true} &= \frac{q'}{|\,q'|} \end{aligned} \tag{5.4}$$

Here $\boldsymbol{\Delta\phi}$ is the vector of small rotation angles, and q' is normalized to obtain the final quaternion. Equation 5.1 requires quaternion multiplication. Two quaternions q and q' can be multiplied as shown in Equation 5.5, using a matrix formulation for ease of coding [18].

$$q'' = q \otimes q' = \begin{bmatrix} q''_1 \\ q''_2 \\ q''_3 \\ q''_4 \end{bmatrix} = \begin{bmatrix} q'_4 & q'_3 & -q'_2 & q'_1 \\ -q'_3 & q'_4 & q'_1 & q'_2 \\ q'_2 & -q'_1 & q'_4 & q'_3 \\ -q'_1 & -q'_2 & -q'_3 & q'_4 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{5.5}$$

Note that all quaternions used in this study are "right-handed" with the scalar term last.

5.2.2 Maneuver Model

At each maneuver time, the flight computer calculates the maneuver required to drive back to the nominal trajectory. This required change in velocity, $\Delta\boldsymbol{v}_{req}$, is passed to the maneuver model as shown in Figures 5-1 and 5-2. The function of the maneuver model is to accept the required $\Delta\boldsymbol{v}$ from the flight computer (this could be considered the "commanded" change in velocity) and use this to simulate a physically

Maneuver Size	Magnitude Error	Direction Error
< 10 cm/s	$\lvert\Delta \boldsymbol{v}\rvert R(0, {\sigma_R}^2) + S(0, {\sigma_S}^2)$	$\lvert\Delta \boldsymbol{v}\rvert T(0, {\sigma_T}^2)$
> 10 cm/s	$\lvert\Delta \boldsymbol{v}\rvert U(0, {\sigma_U}^2) + V(0, {\sigma_V}^2)$	$\lvert\Delta \boldsymbol{v}\rvert W(0, {\sigma_W}^2)$

Table 5.1: Standard execution error model

realistic maneuver execution. The applied $\Delta \boldsymbol{v}$ will never be identical to the required value, due to physical properties of the system such as fuel flow variations, thruster manufacturing variations, thruster misalignments, cut-off errors, and pointing errors.

In the maneuver model, an execution error is applied any time a non-zero maneuver is requested by the flight computer. The execution error model is modular and can be easily replaced. The current model is a subset of one provided by CNES in a memo [3] dated 9/27/02, with the error strategy presented in Table 5.1. Here the variables R, S, T, U, V, and W are unique normally distributed random variables with a mean of zero and variance of σ_R^2, σ_S^2, σ_T^2, σ_U^2, σ_V^2, and σ_W^2, respectively, provided by the user. The magnitude error is applied in the direction of the desired $\Delta \boldsymbol{v}$ burn, and the direction error is applied in a random direction normal to the desired $\Delta \boldsymbol{v}$ burn.

5.2.3 Dynamics Model

The translational dynamics of both vehicles are calculated by integrating the 2-body vector equation of motion shown in Equation 5.6.

$$\ddot{\boldsymbol{r}} = \dot{\boldsymbol{v}} = -\frac{\mu}{r^3}\boldsymbol{r} \tag{5.6}$$

Here μ is the gravitational parameter for the central body (i.e. Mars) and $\boldsymbol{r}$ is the position vector to the spacecraft. It is relatively easy to include a more sophisticated gravity model that accounts for the central body's oblateness (i.e. the J2 effect), however this capability has not been used for the present study. The J2 effect is small for a single satellite, and the relative J2 effect between two spacecraft in similar orbits is small enough to be neglected. This is particularly true for short integration times, and the trajectories used for this study are completed in approximately 24 hours.

In order to be an accurate model of true space vehicle dynamics, several components must be added to this basic integration. Many small perturbations are not modeled by the 2-body equation, such as n-body effects, solar radiation pressure, atmospheric drag, fuel slosh, and outgassing. To allow for these small but important perturbations, an unmodeled acceleration term is added during each integration time step. The unmodeled accelerations in the simulation are characterized by a "strength" or "power" term, k_u, with units of m^2/s^3. The unmodeled acceleration is injected into the integration at every time step as a normally distributed variable with mean zero and variance as shown in Equation 5.7.

$$\sigma_u{}^2 = \frac{k_u}{\Delta t} \tag{5.7}$$

Here Δt is the simulation time step and $\sigma_u{}^2$ has units of m^2/s^4, appropriate for the variance of an acceleration. This formulation for the variance results in an unmodeled acceleration that is independent of the time step [8]. Distinct values of k_u can be assigned for the target and chaser vehicles.

In a real rendezvous mission, the initial position and velocity of both vehicles would only be known within a given uncertainty. For this reason, in the simulation both vehicles are initialized with position and velocity dispersions on their initial states. Perturbed initial states are supplied to the environment model as initial conditions to the two-body equation of motion integration, and are unique to each trial in a Monte-Carlo run.

Assuming that correlations are known between the position and velocity states of one vehicle,[2] it is desirable to account for this when initializing the states. The uncertainty in the states is given as a generic initial covariance matrix, $\boldsymbol{P}$, with the position and velocity variances on the main diagonal and possible correlated terms

[2]Correlations are in fact known, see Section 6.3. This discussion will be kept generic.

off of the diagonal. This covariance matrix represents an error ellipse (for two states, or an ellipsoid, hyper-ellipse, etc. for more states) with an orientation described by the elements of the matrix. A diagonal matrix will have the primary axes parallel to the given coordinate system; a matrix with off-diagonal terms will not be perfectly aligned.

One can then visualize that it is possible to define a new diagonal covariance matrix, $\boldsymbol{P}'$, that is represented by the same "shape" as $\boldsymbol{P}$, but has been rotated to line up with the coordinate system.[3] Thus there would exist a rotation matrix $\boldsymbol{T}$, such that:

$$\boldsymbol{P}' = \boldsymbol{T}\boldsymbol{P}\boldsymbol{T}^T \tag{5.8}$$

It is a well-known property [13] that for any real symmetric matrix $\boldsymbol{A}$ there exists an orthogonal matrix $\boldsymbol{D}$ that diagonalizes $\boldsymbol{A}$. The matrix $\boldsymbol{D}$ is created by arranging the unit norm eigenvectors as the columns of the matrix $\boldsymbol{D}$, and then the matrix $\boldsymbol{A}'$ is simply a diagonal matrix of the eigenvectors.

$$\boldsymbol{A}' = \boldsymbol{D}^T\boldsymbol{A}\boldsymbol{D} \tag{5.9}$$

The rotation matrix $\boldsymbol{T}$ and the diagonalized covariance $\boldsymbol{P}'$ can then be formed by solving the eigenvalue problem for the initial covariance matrix $\boldsymbol{P}$. In the diagonalized covariance matrix, all of the off-diagonal terms have been "incorporated" into the variances on the main diagonal, so initial statistical errors in this new frame can be easily generated by multiplying the square root of the variances by a random number with zero mean and variance of 1. That is:

$$\delta x'_j = \sqrt{\boldsymbol{P}'_{j,j}}\, R_j \tag{5.10}$$

Here $\delta\boldsymbol{x}'_j$ is the j-th random initial statistical error and R is a random variable with mean zero and variance 1. Then the vector $\boldsymbol{\delta x}'$ is created for the diagonal covariance

[3]Alternatively, one could visualize rotating the coordinate system to line up with the primary axes of the ellipse.

matrix $\boldsymbol{P}'$ and can be rotated to the desired frame using:

$$\delta \boldsymbol{x} = \boldsymbol{T} \delta \boldsymbol{x}' \tag{5.11}$$

The resulting vector $\delta \boldsymbol{x}$ from Equation 5.11 is the desired initial perturbation to the state vector, accounting for the known correlations in the covariance matrix.[4]

5.2.4 Camera Model

The function of the camera model is to take the current vehicle positions and chaser vehicle attitude, and simulate an optical camera measurement as shown in Figure 5-3. The simulator is designed around optical observations, so the camera model is configured to return measurements of elevation, azimuth, and range from the chaser vehicle to the target vehicle. The model uses the true positions of both vehicles and the true chaser attitude (as defined in Equation 5.1) to calculate the true elevation, azimuth, and range.

Azimuth and elevation are calculated based on the true unit vector in the direction of the target spacecraft, as viewed from the chaser vehicle in the chaser frame (more specifically, in the frame of the camera on the chaser vehicle). The relative position vector can be calculated according to Equation 5.12.

$$\boldsymbol{r}^I_{rel} = \boldsymbol{r}^I_{OS} - \boldsymbol{r}^I_{Orb} \tag{5.12}$$

In the camera model, the inertial position vectors $\boldsymbol{r}^I_{OS}$ and $\boldsymbol{r}^I_{Orb}$ come from the dynamics model, where the state has been propagated as described in Section 5.2.3. The relative position vector as calculated in Equation 5.12 represents the true inertial line-of-sight from the chaser to the target, but does not yet incorporate the simulated attitude to provide a body-fixed line-of-sight.

As detailed in Section 5.2.1, the inertial attitude of the chaser vehicle is stored

[4]Note that the resulting perturbations will be in the same coordinate system as the initial covariance matrix. If the initial covariance is given in LVLH coordinates, as will be seen in Section 6.3, then the vector $\delta \boldsymbol{x}$ should be rotated into inertial coordinates before use as shown in Section 3.1.1.

as a quaternion describing the transformation from the inertial frame to the chaser body frame. The flight computer requests a specific attitude for boresight pointing based on the current state estimate, and then the attitude model uses this required attitude as the basis for a simulated "true" attitude. The resulting quaternion can then be used to transfer the inertial relative position vector, $\boldsymbol{r}^I_{rel}$, into the true chaser vehicle body frame, $\boldsymbol{r}^B_{rel}$, as shown in Equation 5.13.

$$\boldsymbol{r}'^B_{rel} = q^*_{I \rightarrow B\_true}\, \boldsymbol{r}'^I_{rel}\, q_{I \rightarrow B\_true} \tag{5.13}$$

Here the quaternion $q_{I \rightarrow B\_true}$ describes the inertial to body transformation (from the attitude model, see Equation 5.1) and $q^*_{I \rightarrow B\_true}$ is its conjugate. Remember that all quaternions used are "right-handed" with the scalar term last. The vectors $\boldsymbol{r}'^I_{rel}$ and $\boldsymbol{r}'^B_{rel}$ are the relative position vectors represented as quaternions, which are formulated as shown in Equation 5.14.

$$\boldsymbol{r}'^I_{rel} = \begin{bmatrix} \boldsymbol{r}^I_{rel} \\ 0 \end{bmatrix}, \ \boldsymbol{r}'^B_{rel} = \begin{bmatrix} \boldsymbol{r}^B_{rel} \\ 0 \end{bmatrix} \tag{5.14}$$

The result of the transformation in Equation 5.13 is the relative position vector from the chaser to the target, in the chaser vehicle's body frame. Assuming that the camera is mounted perfectly with its axes lined up with the body frame, the desired measurements could now be calculated from the relative position. It is likely, however, that the camera is *not* mounted perfectly to the body frame. Thus the camera model allows for two additional rotations to place $\boldsymbol{r}^B_{rel}$ into the true camera frame: a rotation between the chaser body frame and the nominal camera frame, and a rotation between the nominal camera frame and the "true" camera frame. The difference between the nominal camera frame and the true camera frame is called the static camera alignment error, representing the physical impossibility of mounting the camera on the spacecraft *exactly* as desired. The static camera alignment error is a constant for a given Monte-Carlo trial, but should be normally distributed over all of the Monte-Carlo trials. It is provided to the camera model as a vector $\boldsymbol{\epsilon}$ of three small angle errors with variance ${\sigma_c}^2$ and mean zero.

The transformation from the chaser vehicle body frame to the true camera frame is accomplished according to Equations 5.15, 5.16, and 5.17, where again the prime mark means the quaternion formulation of a vector.

$$\boldsymbol{r}'_{rel,cam\_true} = q^*_{B\rightarrow cam\_true} \otimes \boldsymbol{r}'^{B}_{rel} \otimes q_{B\rightarrow cam\_true} \tag{5.15}$$

$$q_{B\rightarrow cam\_true} = q_{B\rightarrow cam\_nom} \otimes q_{cam\_nom\rightarrow cam\_true} \tag{5.16}$$

$$q_{cam\_nom\rightarrow cam\_true} = f_{vector\rightarrow quat}(\boldsymbol{\epsilon}) \tag{5.17}$$

Here $\boldsymbol{\epsilon}$ is the vector of three small angles, and $f_{vector\rightarrow quat}$ is a function that derives a quaternion from a vector of three small rotation angles, as described in Equation 5.4. The quaternion $q_{B\rightarrow cam\_nom}$ represents the known rotation between the chaser body frame and the nominal camera frame.

Having finally determined $\boldsymbol{r}_{rel,cam\_true}$, the relative position vector in the true camera frame, the simulated elevation and azimuth measurements can be calculated directly. Elevation, e, is defined to be the angle above the camera x-y plane, and azimuth, α, is defined in the x-y plane. This configuration is shown in Figure 5-3. From Figure 5-3 it is clear that elevation and azimuth can be calculated using Equations 5.18, 5.19, and 5.20. The true range measurement, r, is also easily calculated as the magnitude of the relative position vector, as shown in Equation 5.21.

$$\hat{\boldsymbol{i}}_{LOS} = \frac{\boldsymbol{r}_{rel,cam\_true}}{|\,\boldsymbol{r}_{rel,cam\_true}|} = \begin{bmatrix} i_x \\ i_y \\ i_z \end{bmatrix} \tag{5.18}$$

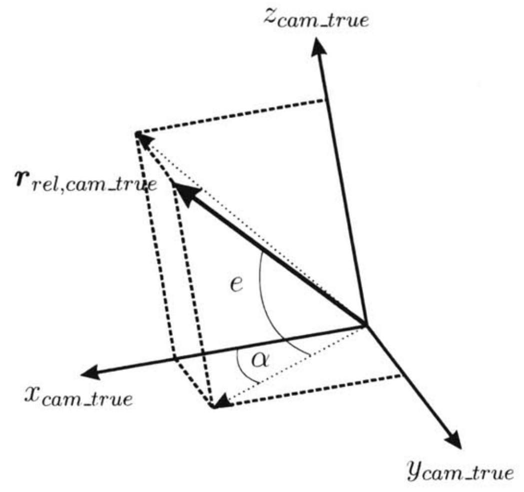
$$e = \sin^{-1}(i_z) \tag{5.19}$$

$$\alpha = \tan^{-1}\left(\frac{i_y}{i_x}\right) \tag{5.20}$$

$$r = |\,\boldsymbol{r}_{rel,cam\_true}| \tag{5.21}$$

Measurement noise and bias values on elevation and azimuth are then added directly

Figure 5-3: Definitions of elevation and azimuth in the camera frame



to the true values as random variables with normal distribution. These random variables have variance ${\sigma_e}^2$ as provided by the user, and mean m_e that corresponds to an angle measurement bias. The angle bias should be constant for a given trial but normally distributed over all of the Monte-Carlo trials.

The variance of the optically determined *range* measurement noise is a function of the angle noise variance, as shown in Equation 5.22.

$$ {\sigma_r}^2 = \left(\frac{r^2}{d}\right)^2 {\sigma_e}^2 \tag{5.22} $$

Here ${\sigma_r}^2$ is the variance of the range measurement, d is the diameter of the target vehicle, and ${\sigma_e}^2$ is the angle variance as described above.

Equation 5.22 can be derived by remembering that all measurements in this study are taken by an optical camera making observations of a well-known target vehicle, the OS. Since the diameter of the OS is known, a range measurement can be calculated based on the apparent size of the OS in the camera frame. Note that, for small d, the relationship between the diameter of the target, d, the range to the target, r, and the angle subtended by the target, θ, is given by Equation 5.23.

$$d = r\theta \tag{5.23}$$

Then the relationship between a small change in range, dr, and a small change in angle, $d\theta$, is shown by Equation 5.24.

$$dr = -\frac{r^2}{d}d\theta \tag{5.24}$$

Equation 5.22 then follows by taking the expected value of dr^2 to find the variance of the range measurement.

$${\sigma_r}^2 = E[dr^2] = E[(-\frac{r^2}{d})^2 d\theta^2] = \left(\frac{r^2}{d}\right)^2 {\sigma_e}^2 \tag{5.25}$$

The camera model is configured to use only one camera at a time, however two cameras with different values for ${\sigma_e}^2$ and m_e can be specified and toggled on a specific value of range. This capability is provided so that a narrow angle camera (NAC) can be used until the target fills its field of view (at some range), and then the spacecraft will switch to a wide angle camera (WAC) for the duration of the rendezvous.

There are three major constraints implemented in the camera model: field of view (FOV) angle constraint, sun angle constraint, and target eclipse constraint. These three constraints are described briefly below:

Field of View Constraint After the camera model calculates the simulated measurements, it checks to make sure the angle values returned would actually be in the field of view of the camera. For example, the NAC has a full cone angle field of view of 1.4 degrees, so any elevation or azimuth angle computed to be larger than 0.7 degrees is not physically possible. If the target craft was one degree away from the camera boresight, the camera would return an empty field and no measurement update could be performed. Thus, the simulation flags the measurement as invalid due to the FOV constraint, and passes this flag to the flight computer instead of the true measurement. The flight computer can then take an appropriate action for the missed measurement, typically by performing no update. Given Equation 5.18 above describing the true LOS unit vector, the FOV angle, γ, can be easily calculated as:

$$\gamma = \cos^{-1}(i_x) \tag{5.26}$$

Sun Angle Constraint To avoid image processing problems in the optical measurements, pictures of the target are not allowed when the angle between the Sun and the boresight is less than some specified value. If the camera is pointed too close to the Sun, the simulation flags the measurement as invalid due to the sun angle constraint, and passes this flag to the flight computer instead of the true measurement. The flight computer can then take an appropriate action for the missed measurement as before.

Knowing the vector from the planet to the Sun in the current inertial coordinate system (taken from ephemeris data based on the epoch time for the simulation), the Sun angle, θ, is easily calculated as:

$$\theta = \cos^{-1} \frac{\boldsymbol{r}_{rel} \cdot \boldsymbol{r}_{Orb \to Sun}}{|\boldsymbol{r}_{rel}||\boldsymbol{r}_{Orb \to Sun}|} \tag{5.27}$$

Here $\boldsymbol{r}_{rel}$ is taken to be the relative position vector from Equation 5.12, and $\boldsymbol{r}_{Orb \to Sun}$, the vector from the chaser to the Sun, is calculated as:

$$\boldsymbol{r}_{Orb \to Sun} = \boldsymbol{r}^I_{Sun} - \boldsymbol{r}^I_{Orb} \tag{5.28}$$

Target Eclipse Constraint Since this simulation was designed for optical measurements, it is necessary for the target to be illuminated by the Sun when taking a picture. If a measurement is required but the target is in eclipse, the simulation flags the measurement as invalid due to the target eclipse constraint, and passes this flag to the flight computer instead of the true measurement. The simulator does allow for the possibility of a "headlight" on the chaser vehicle, which would illuminate the target at some specified range. When the relative distance between the chaser and target is inside this range, the sunlight constraint will never be flagged.

The conditions for target shadowing are shown in Figure 5-4. Here r_{eq} is the equatorial radius of the planet, $\boldsymbol{r}_{OS}$ is the inertial position vector to the target, and $\boldsymbol{r}_{Sun}$ is the inertial position vector to the Sun. The quantity p is the magnitude of the target position vector in the direction of the Sun vector, defined by:

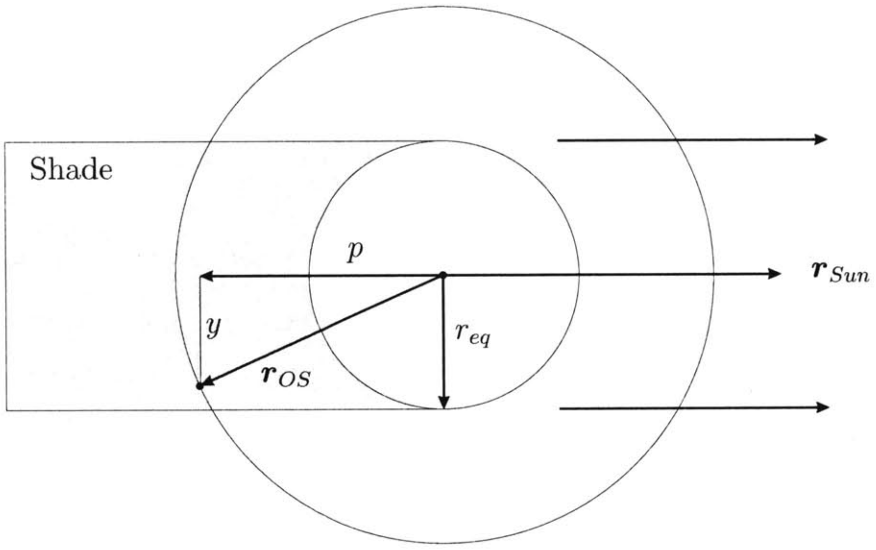
$$p = \frac{\boldsymbol{r}_{Sun}}{|\boldsymbol{r}_{Sun}|} \cdot \boldsymbol{r}_{OS} \tag{5.29}$$

This is a scalar quantity but shown in the figure in the direction opposite the Sun vector. The scalar variable y is calculated as:

$$y = \sqrt{\left(\frac{\boldsymbol{r}_{OS}}{|\boldsymbol{r}_{OS}|}\right)^2 - \left(\frac{\boldsymbol{r}_{Sun}}{|\boldsymbol{r}_{Sun}|} \cdot \boldsymbol{r}_{OS}\right)^2} \tag{5.30}$$

Two things can then be seen from the figure. First, if the dot product of the target position vector and the Sun vector is positive, then clearly the target is not in eclipse. Second, if the dot product is negative and the quantity y is less than the equatorial radius, as shown in the figure, then the target does experience some shading. This algorithm is of course an approximation that does not evaluate partial lighting conditions in the penumbra, etc. It is sufficient for this study to know that lighting conditions are changing when indicated by the algorithm above, and due to the optical nature of the measurements used, no pictures will be taken.

Figure 5-4: Conditions for target shadowing constraint



Chapter 6

Rendezvous Navigation Filters

The Kalman filter was officially conceived in 1960 by R. E. Kalman in his seminal paper on linear filtering and prediction [7]. Since then, the basic Kalman filter has been the subject of a great deal of research, resulting in a wide variety of derivations, modifications, and applications that have been constructed by creative researchers. A generic "standard" Kalman filter[1] was derived in Chapter 4 as a basis for the navigation algorithms studied in this work. The next three chapters will describe the implementations of two such modifications that are used – or could be used – in space flight navigation. These algorithms, called here the Precomputed Gain Kalman filter algorithm and the Extended Kalman filter algorithm, will be presented with a rationale for their use, the distinguishing theory behind the name, and their implementation for the study at hand.

In this chapter, a description is provided of the two rendezvous navigation filters. Despite the different linearization strategies, both filters in this study are based on the same set of states and measurements, so it is useful to present these parameters together. Other similarities such as the initialization of the error covariance matrix and the calculation of the measurement noise covariance matrix are also presented together in this chapter. At the end of the chapter, a section each is devoted to the Precomputed Gain and Extended filter designs, to discuss characteristics that are not common between them.

[1]That is, a standard Kalman filter with no application.

Chapter 7 describes all of the targeting algorithms used in this study, including those used exclusively by the Precomputed Gain or Extended implementations. The basic building blocks of a full rendezvous trajectory have been presented in Section 3.2, and Chapter 7 now discusses exactly how these types of relative motion are initiated and maintained in implementation.

Finally, Chapter 8 details the "flight software" for both the Precomputed Gain and Extended Kalman filters. It includes rationale and motivation for their use, as well as the details of their implementation for this study.

6.1 Filter States and Dynamics

Section 4.1.1 emphasized the importance of properly defining the state vector when designing a Kalman filter. In this implementation, 20 states are used as shown in Equation 6.1.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r}_{OS}^T & \boldsymbol{v}_{OS}^T & \boldsymbol{r}_{Orb}^T & \boldsymbol{v}_{Orb}^T & b_e & b_\alpha & \boldsymbol{\epsilon}^T & \boldsymbol{\Theta}^T \end{bmatrix}^T \tag{6.1}$$

It will also be useful to define a nominal state vector and an estimated state vector for some of the discussions to follow. These are shown in Equations 6.2 and 6.3, respectively:

$$\boldsymbol{x}^* = \begin{bmatrix} \boldsymbol{r}_{OS}^{*T} & \boldsymbol{v}_{OS}^{*T} & \boldsymbol{r}_{Orb}^{*T} & \boldsymbol{v}_{Orb}^{*T} & b_e^* & b_\alpha^* & \boldsymbol{\epsilon}^{*T} & \boldsymbol{\Theta}^{*T} \end{bmatrix}^T \tag{6.2}$$

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{\boldsymbol{r}}_{OS}^T & \hat{\boldsymbol{v}}_{OS}^T & \hat{\boldsymbol{r}}_{Orb}^T & \hat{\boldsymbol{v}}_{Orb}^T & \hat{b}_e & \hat{b}_\alpha & \hat{\boldsymbol{\epsilon}}^T & \hat{\boldsymbol{\Theta}}^T \end{bmatrix}^T \tag{6.3}$$

The first 12 states are the position and velocity of the OS and orbiter, respectively. The variables b_e and b_α represent angle biases in the elevation and azimuth measurements, respectively. The variable $\boldsymbol{\epsilon}$ is a vector representing the three angular components of the static camera alignment error. Finally, the vector $\boldsymbol{\Theta}$ represents the three angular components of the inertial attitude knowledge error.

The following three sections discuss properties of the states that are common to both implementations and will be useful later on in the discussion of state and covariance propagation.

6.1.1 Position and Velocity

The first twelve states in the state vector are inertial position and velocity of both vehicles. It will be necessary in both filter implementations to calculate a state transition matrix, $\boldsymbol{\Phi}_{i+1,i}$, which can be used to propagate position and velocity states of the error covariance matrix from time t_i to time t_{i+1}. The technique used to generate state transition matrices is one provided by Lear [8], and is outlined as follows.[2]

When concerned only with translational vehicle dynamics, the state of a single vehicle contains only its position and velocity as shown in Equation 6.4.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r} \\ \dot{\boldsymbol{r}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{v} \end{bmatrix} \tag{6.4}$$

Recall the two-body equation of motion presented in Section 5.2.3:

$$\ddot{\boldsymbol{r}} = \dot{\boldsymbol{v}} = -\frac{\mu}{r^3}\boldsymbol{r} = g(\boldsymbol{r}) \tag{5.6}$$

It is then clear that the time derivative of the state, $\dot{\boldsymbol{x}}$, can be expressed as a function of the state, $\boldsymbol{x}$, as follows:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{v} \\ -\frac{\mu}{r^3}\boldsymbol{r} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{g}(\boldsymbol{r}) \end{bmatrix} \tag{6.5}$$

Linearizing the above equation around the state $\boldsymbol{x} = \boldsymbol{x}^* + \boldsymbol{\delta x}$ results in the following

[2]This derivation will be performed using the Precomputed Gain linearization. That is, it will be said that $\boldsymbol{x} = \boldsymbol{x}^* + \boldsymbol{\delta x}$, and the resulting matrix $\boldsymbol{A}$ will be evaluated on the nominal trajectory, $\boldsymbol{r}^*$. This is done out of a need to choose a point to linearize about, but note that the derivation would still be valid if linearized about $\boldsymbol{x} = \hat{\boldsymbol{x}} - \tilde{\boldsymbol{x}}$. The result in Equation 6.7 is then $\boldsymbol{A} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{r}}\Big|_{\hat{\boldsymbol{r}}}$ and the state transition matrix can be used to propagate $\tilde{\boldsymbol{x}}_{i+1} = \boldsymbol{\Phi}_{i+1,i}\,\tilde{\boldsymbol{x}}_i$.

form:

$$\dot{\boldsymbol{\delta x}} = \begin{bmatrix} \dot{\boldsymbol{\delta r}} \\ \dot{\boldsymbol{\delta v}} \end{bmatrix} = \begin{bmatrix} 0 & \boldsymbol{I} \\ \boldsymbol{A} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta r} \\ \boldsymbol{\delta v} \end{bmatrix} \tag{6.6}$$

Here the matrix $\boldsymbol{A}$ is defined as shown in Equation 6.7:

$$\boldsymbol{A} = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{r}}\right|_{\boldsymbol{r}^*} \tag{6.7}$$

The fourth-order Runge-Kutta state transition matrix, such that $\boldsymbol{\delta x}_{i+1} = \boldsymbol{\Phi}_{i+1,i}\, \boldsymbol{\delta x}_i$, is shown by Lear [8] to be the matrix in Equation 6.8:

$$\boldsymbol{\Phi}_{i+1,i} = \begin{bmatrix} \boldsymbol{I} + \boldsymbol{T}_1 \frac{\Delta t^2}{6} + \boldsymbol{T}_3 \frac{\Delta t^4}{24} & \boldsymbol{I}\Delta t + \boldsymbol{A}_{i+0.5} \frac{\Delta t^3}{6} \\ (\boldsymbol{T}_1 + \boldsymbol{T}_2)\frac{\Delta t}{6} + (\boldsymbol{T}_3 + \boldsymbol{T}_4)\frac{\Delta t^3}{12} & \boldsymbol{I} + \boldsymbol{T}_2 \frac{\Delta t^2}{6} + \boldsymbol{T}_4 \frac{\Delta t^4}{24} \end{bmatrix} \tag{6.8}$$

Here the matrices $\boldsymbol{T}_1$ through $\boldsymbol{T}_4$ are defined in Equation 6.9. Note from these definitions that three evaluations of the $\boldsymbol{A}$ matrix are required: $\boldsymbol{A}_i$, $\boldsymbol{A}_{i+0.5}$, and $\boldsymbol{A}_{i+1}$.

$$\begin{aligned} \boldsymbol{T}_1 &= \boldsymbol{A}_i + 2\boldsymbol{A}_{i+0.5} \\ \boldsymbol{T}_2 &= 2\boldsymbol{A}_{i+0.5} + \boldsymbol{A}_{i+1} \\ \boldsymbol{T}_3 &= \boldsymbol{A}_{i+0.5}\boldsymbol{A}_i \\ \boldsymbol{T}_4 &= \boldsymbol{A}_{i+1}\boldsymbol{A}_{i+0.5} \end{aligned} \tag{6.9}$$

Using the technique outlined above for both the target and chaser vehicle, a state transition matrix can be calculated at every time step. Later sections will detail the use of the state transition matrices in the simulated flight computers.

When propagating the error covariance matrix in both filter implementations, the process noise covariance matrix will be required for every state. For the vehicle positions and velocities, the process noise covariance is based on the unmodeled accelerations as described in Chapter 5.2.3 (accounting for n-body effects, solar radiation pressure, atmospheric drag, fuel slosh, outgassing, etc., as perturbations to the standard two-body equation). Given this small acceleration added to the equations of motion. Lear [8] derives the following process noise covariance matrix, for one vehicle, whose cumulative effect over the time interval $[0,\ t_f]$ is independent of the time step Δt:

$$\boldsymbol{Q}_u = k_u \begin{bmatrix} \boldsymbol{I}\dfrac{\Delta t^3}{4} & \boldsymbol{I}\dfrac{\Delta t^2}{2} \\ \boldsymbol{I}\dfrac{\Delta t^2}{2} & \boldsymbol{I}\Delta t \end{bmatrix} \tag{6.10}$$

This calculation is performed for each vehicle, where the value of k_u can be different between them.

6.1.2 Markov Processes

The remaining 8 states are modeled in both implementations as First Order Markov processes, which are suitable for modeling the uncertainties of state variables that experience some time variability. These states are: static alignment error, inertial attitude error, and measurement angle biases in elevation and azimuth. Note that only the inertial attitude error has a time constant that is small relative to the total trajectory time; the rest have near infinite time constants, causing them to be constant biases as desired. Still, by modeling all of the errors as Markov processes, the calculations of the state transition matrix and process noise covariance are greatly simplified.

As with the position and velocity states above, it will be necessary in both algorithms to propagate these states and the associated error covariance, and to find the process noise covariance matrix. To do this, start with the equation defining a First Order Markov process in discrete time. Lear presents this equation, which can be written in vector form as:

$$\tilde{\boldsymbol{y}}_{i+1} = e^{-\Delta t/\tau}\tilde{\boldsymbol{y}}_i + \sigma_y\sqrt{1-e^{-2\Delta t/\tau}}\,\boldsymbol{U}_i \tag{6.11}$$

Here $\boldsymbol{U}$ is a normally distributed vector random variable with mean zero and covariance $\boldsymbol{I}$ (identity), and $\tilde{\boldsymbol{y}}_0$ is normally distributed with zero mean and covariance $\sigma_y^2\boldsymbol{I}$. Note that every component of $\tilde{\boldsymbol{y}}$ has the the same value of τ. The covariance matrix associated with the uncertainty modeled by $\tilde{\boldsymbol{y}}_{i+1}$ can then be expressed:

$$\boldsymbol{P}_{y,\,i+1} = E[\tilde{\boldsymbol{y}}_{i+1}\tilde{\boldsymbol{y}}_{i+1}^T] = e^{-2\Delta t/\tau}\boldsymbol{P}_{y,\,i} + \sigma_y^2(1 - e^{-2\Delta t/\tau})\,\boldsymbol{I} \tag{6.12}$$

From the Markov covariance propagation equation shown above, a form analogous to Equation 4.23 can be formed as follows:

$$\begin{aligned} \boldsymbol{P}_{y,\,i+1} &= \left(e^{-\Delta t/\tau}\boldsymbol{I}\right)\boldsymbol{P}_{y,\,i}\left(e^{-\Delta t/\tau}\boldsymbol{I}\right)^T + \sigma_y^2(1 - e^{-2\Delta t/\tau})\,\boldsymbol{I} & (6.13)\\ &= \boldsymbol{\Phi}_{y,\,i+1,\,i}\,\boldsymbol{P}_{y,\,i}\,\boldsymbol{\Phi}_{y,\,i+1,\,i} + \boldsymbol{Q}_{y,\,i} & (6.14)\end{aligned}$$

Thus a state propagation matrix and process noise covariance matrix have been arrived upon for uncertainties expressed as First Order Markov processes, summarized below:

$$\boldsymbol{\Phi}_{y,\,i+1,\,i} = e^{-\Delta t/\tau}\boldsymbol{I} \tag{6.15}$$

$$\boldsymbol{Q}_{y,\,i} = \sigma_y^2(1 - e^{-2\Delta t/\tau})\,\boldsymbol{I} \tag{6.16}$$

6.2 Measurement Model

The sensor package assumed in this study is that of the MTO spacecraft, as discussed in Section 2.1. MTO will navigate solely through the use of an optical camera capable of angle and range measurements to a target of known size and shape. Flexibility has been added to the algorithms to include two unique cameras, however the cameras will never operate simultaneously, so there will always be a maximum of three measurements at any given time.

Three separate 'measurements,' or pieces of relative information between the orbiter and the sample, can be obtained from a single optical image. The onboard camera can directly measure azimuth and elevation angles to the target relative to the camera boresight (the camera's centerline), given the coordinate system definition shown in Figure 5-3. In addition, a range measurement can be derived from the apparent size of the sample in the image, since the actual size and shape of the OS and lighting conditions will be precisely known. Using either the WAC or the NAC, these are the only measurements considered in this study. The elevation, azimuth, and range measurements are summarized in Equation 6.17, respectively.

$$\boldsymbol{z} = \begin{bmatrix} e_m \\ \alpha_m \\ \boldsymbol{r}_m \end{bmatrix} = \boldsymbol{h}(\boldsymbol{x}, t) + \boldsymbol{v} \tag{6.17}$$

Here the elevation angle, e_m, azimuth angle, α_m, and range, $|\boldsymbol{r}_{rel}|_m$ are defined as follows:

$$\begin{aligned} e_m &= e(\boldsymbol{r}_{rel}, \boldsymbol{\epsilon}, \boldsymbol{\Theta}) + e_b + v_e \\ \alpha_m &= \alpha(\boldsymbol{r}_{rel}, \boldsymbol{\epsilon}, \boldsymbol{\Theta}) + \alpha_b + v_\alpha \\ \boldsymbol{r}_m &= r(\boldsymbol{r}_{rel}) + v_r \end{aligned} \tag{6.18}$$

For more detail on the azimuth, elevation, and range measurements, see the camera model in Section 5.2.4.

6.2.1 Measurement Partials

Having now defined the *states* and the *measurements* used in this study, it is possible to discuss the measurement sensitivity matrix, or measurement partials matrix. Recall Equation 4.31 from Section 4.3, derived for the 'standard' Kalman filter:[3]

$$\left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_i}\right|_{\boldsymbol{x}_i^*} = \boldsymbol{H}_i \tag{4.31}$$

Computing this matrix of partial derivatives is not a trivial task. First, the measurements in Equation 6.17 must be formulated in terms of the state variables. The measurements for this implementation are clearly nonlinear, so calculating the partial derivatives themselves is quite tedious. In his Master's Thesis entitled "Trajectory

[3]While the discussion here centers on the measurement sensitivity matrix derived for the Precomputed Gain filter, it will be shown that this is the same matrix used by the Extended filter, merely evaluated at a different point (see Equation 6.43).

Design and Analysis of a Mars-Orbit Rendezvous Flight Experiment," author Geoffrey Huntington does a very thorough job of deriving the partial derivatives for these measurements [5]. In fact, Huntington's derivation includes more states than are used in the present analysis, so only a subset of his sensitivity matrix is presented here. The following is the structure of the sensitivity matrix before substitution of the partial derivatives has been made:

$$\boldsymbol{H} = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^*} = \begin{bmatrix} \frac{\partial e}{\partial \boldsymbol{r}_{OS}} & \frac{\partial \alpha}{\partial \boldsymbol{r}_{OS}} & \frac{\partial |\boldsymbol{r}_{rel}|}{\partial \boldsymbol{r}_{OS}} \\ \frac{\partial e}{\partial \boldsymbol{v}_{OS}} & \frac{\partial \alpha}{\partial \boldsymbol{v}_{OS}} & \frac{\partial |\boldsymbol{r}_{rel}|}{\partial \boldsymbol{v}_{OS}} \\ \frac{\partial e}{\partial \boldsymbol{r}_{Orb}} & \frac{\partial \alpha}{\partial \boldsymbol{r}_{Orb}} & \frac{\partial |\boldsymbol{r}_{rel}|}{\partial \boldsymbol{r}_{Orb}} \\ \frac{\partial e}{\partial \boldsymbol{v}_{Orb}} & \frac{\partial \alpha}{\partial \boldsymbol{v}_{Orb}} & \frac{\partial |\boldsymbol{r}_{rel}|}{\partial \boldsymbol{v}_{Orb}} \\ \frac{\partial e}{\partial b_e} & \frac{\partial \alpha}{\partial b_e} & \frac{\partial |\boldsymbol{r}_{rel}|}{\partial b_e} \\ \frac{\partial e}{\partial b_\alpha} & \frac{\partial \alpha}{\partial b_\alpha} & \frac{\partial |\boldsymbol{r}_{rel}|}{\partial b_\alpha} \\ \frac{\partial e}{\partial \epsilon} & \frac{\partial \alpha}{\partial \epsilon_b} & \frac{\partial |\boldsymbol{r}_{rel}|}{\partial \epsilon_b} \\ \frac{\partial e}{\partial \Theta} & \frac{\partial \alpha}{\partial \Theta} & \frac{\partial |\boldsymbol{r}_{rel}|}{\partial \Theta} \end{bmatrix}^T_{\boldsymbol{x}^*} \tag{6.19}$$

Substituting the measurement partial derivatives as derived in Huntington's work, the following sensitivity matrix is obtained for the filter implementations under study:

$$\boldsymbol{H} = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^*} = \begin{bmatrix} \dfrac{\boldsymbol{p}_e^I}{|\boldsymbol{r}_{rel}|} & \dfrac{\boldsymbol{p}_\alpha^I}{|\boldsymbol{r}_{rel}|\cos^2(e)} & \hat{\boldsymbol{i}}_{LOS}^I \\ 0 & 0 & 0 \\ -\dfrac{\boldsymbol{p}_e^I}{|\boldsymbol{r}_{rel}|} & -\dfrac{\boldsymbol{p}_\alpha^I}{|\boldsymbol{r}_{rel}|\cos^2(e)} & -\hat{\boldsymbol{i}}_{LOS}^I \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ \boldsymbol{p}_e^c \times \hat{\boldsymbol{i}}_{LOS}^c & \dfrac{\boldsymbol{p}_\alpha^c \times \hat{\boldsymbol{i}}_{LOS}^c}{\cos^2(e)} & 0 \\ \boldsymbol{p}_e^c \times \hat{\boldsymbol{i}}_{LOS}^c & \dfrac{\boldsymbol{p}_\alpha^c \times \hat{\boldsymbol{i}}_{LOS}^c}{\cos^2(e)} & 0 \end{bmatrix}_{\boldsymbol{x}^*}^T \tag{6.20}$$

Here $\boldsymbol{p}_e^c$ is the partial derivative of the line of sight (LOS) unit vector with respect to elevation, and $\boldsymbol{p}_\alpha^c$ is the partial derivative of the LOS unit vector with respect to azimuth. Both vectors are expressed in the camera frame, indicated by the letter c in the superscript [5]. The vectors $\boldsymbol{p}_e^I$ and $\boldsymbol{p}_\alpha^I$ in Equation 6.20 are obtained by transforming $\boldsymbol{p}_e^c$ and $\boldsymbol{p}_\alpha^c$ into the inertial frame.

$$\boldsymbol{p}_e^c = \frac{\partial \hat{\boldsymbol{i}}_{LOS}^c}{\partial e} = \begin{bmatrix} -\sin(e-b_e)\cos(\alpha-b_\alpha) \\ -\sin(e-b_e)\sin(\alpha-b_\alpha) \\ \cos(e-b_e) \end{bmatrix} \tag{6.21}$$

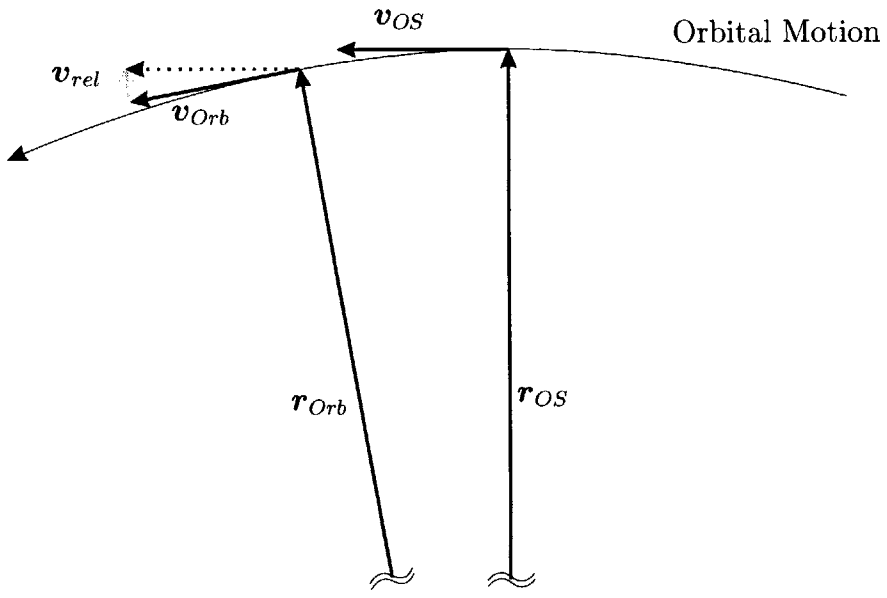
$$\boldsymbol{p}_\alpha^c = \frac{\partial \hat{\boldsymbol{i}}_{LOS}^c}{\partial \alpha} = \begin{bmatrix} -\sin(\alpha-b_\alpha)\cos(e-b_e) \\ \cos(\alpha-b_\alpha)\cos(e-b_e) \\ 0 \end{bmatrix} \tag{6.22}$$

6.3 Covariance Initialization

The state vector used in this study is 20 elements long (see Equation 6.1), resulting in an error covariance matrix that is 20×20 according to Equation 4.13. In both filter implementations, the error covariance matrix must be initialized at time t_0, based on the designer's best estimate of the variance of the elements of the state vector. For the last 8 elements of the state (everything but the position and velocity of both vehicles), the initial covariance is simply a diagonal matrix of the corresponding state variances, with no correlations at all. This signifies that at the initial time, the designer has no information to suggest any correlation between the last 8 elements of the state.

For the vehicles' position and velocity, however, a strong correlation is in fact known. Speaking in the LVLH frame, downrange velocity is correlated with altitude position, and similarly, altitude (radial) velocity is correlated with downrange position. The first can be seen intuitively by considering a coelliptic approach, where the altitude of the approach is directly correlated to the closing speed. The second is best illustrated graphically and is shown in Figure 6-1. The figure depicts the target and chaser vehicles on the same circular orbit and separated by a large downrange component. It is then clear from the graphical representations of the vehicles' velocity vectors that the relative velocity vector is largely in the radial direction, and its magnitude is dependent on the separation distance. Thus, in this implementation, the initial covariance matrix for each vehicle in the LVLH frame is defined by

Figure 6-1: Correlation between radial velocity and relative downrange position



Equation 6.23.

$$P^{LVLH} = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 & 0 & -\rho\,\sigma_x\sigma_{\dot z} \\ 0 & \sigma_y^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_z^2 & -\rho\,\sigma_z\sigma_{\dot x} & 0 & 0 \\ 0 & 0 & -\rho\,\sigma_z\sigma_{\dot x} & \sigma_{\dot x}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\dot y}^2 & 0 \\ -\rho\,\sigma_x\sigma_{\dot z} & 0 & 0 & 0 & 0 & \sigma_{\dot z}^2 \end{bmatrix} \tag{6.23}$$

Here σ represents the standard deviation of the position or velocity component indicated by the subscript, where the subscripts keep to the LVLH convention defined in Figure 3-1. The correlation coefficient ρ is given a value of 0.9, to provide the high level of correlation that is known to exist. Before being used in the full covariance, this matrix must be rotated into inertial coordinates using:

$$P^I = TP^{LVLH}T^T \tag{6.24}$$

Here $\boldsymbol{T}$ is a rotation matrix defined by Equation 6.25, where the body to inertial transformation matrix is the transpose of $\boldsymbol{T}_{I\rightarrow LVLH}$ defined by Equations 3.4 and 3.5.

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{T}_{LVLH\rightarrow I} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{T}_{LVLH\rightarrow I} \end{bmatrix} \tag{6.25}$$

6.4 Covariance Propagation

Combining the state transition matrices and process noise covariance matrices from the Sections 6.1.1 and 6.1.2 above, the error covariance matrix for the entire state used in this implementation can be propagated according to Equation 4.23:

$$\boldsymbol{P}_{i+1} = \boldsymbol{\Phi}_{i+1,\,i}\,\boldsymbol{P}_i\,\boldsymbol{\Phi}^T_{i+1,\,i} + \boldsymbol{Q}_i \tag{4.23}$$

Here the state transition matrix and process noise covariance matrix are formulated as follows for the full state (refer to the definition of the state vector, Equation 6.1):

$$\boldsymbol{\Phi}_{i+1,\,i} = \begin{bmatrix} \boldsymbol{\Phi}_{Orb}\,(6\times6) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi}_{OS}\,(6\times6) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Phi}_{b_e,b_\alpha}\,(2\times2) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Phi}_{\epsilon}\,(3\times3) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Phi}_{\Theta}\,(3\times3) \end{bmatrix}_{i+1,\,i} \tag{6.26}$$

Here $\boldsymbol{\Phi}_{Orb}$ and $\boldsymbol{\Phi}_{OS}$ are the state transition matrices for the position and velocity states of the chaser and target, respectively.

$$\boldsymbol{Q}_i = \begin{bmatrix} \boldsymbol{Q}_{u,OS}\,(6\times6) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{Q}_{u,Orb}\,(6\times6) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{Q}_{b_e,b_\alpha}\,(2\times2) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{Q}_{\epsilon}\,(3\times3) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{Q}_{\Theta}\,(3\times3) \end{bmatrix} \tag{6.27}$$

These terms when used in Equation 4.23 allow the propagation of the covariance matrix at every time step. The actual variance and time constant values used in this study can be found in Chapter 9.

6.5 Measurement Noise Covariance

Recall that the measurement noise covariance matrix is defined as $\boldsymbol{R}_i = E[\boldsymbol{v}_i \boldsymbol{v}_j^T]$, where $\boldsymbol{v}_i$ is the measurement noise vector from Equation 4.2:

$$\boldsymbol{z}_i = \boldsymbol{h}(\boldsymbol{x}_i, t_i) + \boldsymbol{v}_i \tag{4.2}$$

Now for this study the measurements are known to be elevation, azimuth, and range to the target, so the definition of the measurement covariance can be refined as follows:

$$\boldsymbol{R}_i = \begin{bmatrix} \sigma_e^2 & 0 & 0 \\ 0 & \sigma_\alpha^2 & 0 \\ 0 & 0 & \sigma_r^2 \end{bmatrix} \tag{6.28}$$

As described in Section 5.2.4, the variance of the azimuth and elevation measurement noise are defined by the user, σ_α^2 and σ_e^2 respectively, and are typically the same value. The variance of the range measurement noise is based on the angle variance as was demonstrated in Equation 5.22:

$${\sigma_r}^2 = \left(\frac{r^2}{d}\right)^2 {\sigma_e}^2 \tag{5.22}$$

Here ${\sigma_r}^2$ is the variance of the range measurement, d is the diameter of the target vehicle, and ${\sigma_e}^2 = {\sigma_\alpha}^2$ is the angle variance as described above.

6.6 Precomputed Gain Kalman Filter

The preceding sections have discussed parameters and calculations that were common between the Precomputed Gain and Extended implementations of the Kalman filter. This section will describe parameters and calculations that are unique to the Precomputed Gain implementation.

6.6.1 Covariance Update for Maneuvers

When a maneuver is performed on the reference trajectory, a velocity error is introduced into the orbiter state that should be reflected in the covariance matrix. In general, the velocity error is caused by the fact that the maneuver performed is not precisely known, even though a specific maneuver was requested, and the maneuver itself might have been measured. In this study, there are no sensors assumed available to measure the applied change in velocity, so the velocity knowledge error comes from the fact that the performed maneuver is not the same as the commanded maneuver. The statistics of the knowledge error can be estimated by the designer, and this section describes the technique by which the filter is made aware of the velocity error.

When a maneuver is performed by the chaser spacecraft, that change in velocity is described by:

$$\Delta \boldsymbol{v}_{exec} = \Delta \boldsymbol{v}^* + \boldsymbol{C}^* \delta \hat{\boldsymbol{x}} + \Delta \boldsymbol{v}_{ee} \tag{6.29}$$

That is, the executed maneuver, $\Delta \boldsymbol{v}_{exec}$, is equal to the nominal maneuver, $\Delta \boldsymbol{v}^*$, plus the corrective maneuver, $\boldsymbol{C}^* \delta \hat{\boldsymbol{x}}$, plus some execution error, $\Delta \boldsymbol{v}_{ee}$. Now let the knowledge error, $\Delta \boldsymbol{v}_k$, be described by the following formula:[4]

$$\Delta \boldsymbol{v}_k = k_{sf} \Delta \boldsymbol{v}_{exec} + \boldsymbol{k}_{pf} \times \Delta \boldsymbol{v}_{exec} + \Delta \boldsymbol{v}_{quant} \tag{6.30}$$

Here k_{sf} is a *scale factor* which, multiplied by the executed $\Delta \boldsymbol{v}$, represents an error proportional to the commanded burn and in the same direction. The vector coefficient $\boldsymbol{k}_{pf}$ is a *pointing factor* which, crossed with the executed $\Delta \boldsymbol{v}$, represents an error proportional to the commanded burn but in a normal direction. Finally, the term $\Delta \boldsymbol{v}_{quant}$ represents a *quantization* error, typically spherical in nature. Substituting

[4]The terms $\Delta \boldsymbol{v}^*$ and $\boldsymbol{C}^* \delta \hat{\boldsymbol{x}}$ will be discussed in detail in Chapter 7. For now, let them just be the nominal and corrective parts of the executed maneuver.

Equation 6.29 into Equation 6.30 yields the following:

$$\Delta \boldsymbol{v}_k \approx k_{sf}(\Delta \boldsymbol{v}^* + \boldsymbol{C}^* \hat{\delta \boldsymbol{x}}) + \boldsymbol{k}_{pf} \times (\Delta \boldsymbol{v}^* + \boldsymbol{C}^* \hat{\delta \boldsymbol{x}}) + \Delta \boldsymbol{v}_{quant} \tag{6.31}$$

Note that the execution error terms have been neglected, as the small scale factors multiplied by the small execution error result in a second-order quantity that is not significant compared to the other terms. The velocity error covariance can then be found by taking the expectation of $\Delta \boldsymbol{v}_k \Delta \boldsymbol{v}_k^T$, recognizing that the cross product in Equation 6.31 can be expressed as shown in Equation 6.33.

$$\begin{aligned} E\left[\Delta \boldsymbol{v}_k \Delta \boldsymbol{v}_k^T\right] = E\Big[&\Big(k_{sf}(\Delta \boldsymbol{v}^* + \boldsymbol{C}^* \hat{\delta \boldsymbol{x}}) + \boldsymbol{k}_{pf} \times (\Delta \boldsymbol{v}^* + \boldsymbol{C}^* \hat{\delta \boldsymbol{x}}) + \Delta \boldsymbol{v}_{quant}\Big) \\ &\Big(k_{sf}(\Delta \boldsymbol{v}^* + \boldsymbol{C}^* \hat{\delta \boldsymbol{x}}) + \boldsymbol{k}_{pf} \times (\Delta \boldsymbol{v}^* + \boldsymbol{C}^* \hat{\delta \boldsymbol{x}}) + \Delta \boldsymbol{v}_{quant}\Big)^T\Big] \end{aligned} \tag{6.32}$$

$$\boldsymbol{k}_{pf} \times (\Delta \boldsymbol{v}^* + \boldsymbol{C}^* \hat{\delta \boldsymbol{x}}) = \boldsymbol{K}_{pf,\otimes} (\Delta \boldsymbol{v}^* + \boldsymbol{C}^* \hat{\delta \boldsymbol{x}}) \tag{6.33}$$

The variable $\boldsymbol{K}_{pf,\otimes}$ represents the skew-symmetric formulation of the pointing factor vector, formed as shown in Equation 6.34.

$$\boldsymbol{K}_{pf,\otimes} = \begin{bmatrix} 0 & -k_{pf_3} & k_{pf_2} \\ k_{pf_3} & 0 & -k_{pf_1} \\ -k_{pf_2} & k_{pf_1} & 0 \end{bmatrix} \tag{6.34}$$

Expanding Equation 6.32 and using the linearity of the expectation function results in a mess of algebra that will not be presented here. The result of simplification follows:

$$\begin{aligned} \boldsymbol{P}_{\Delta v} = \sigma_{sf}^2(\Delta \boldsymbol{v}^* \Delta \boldsymbol{v}^{*T} + \boldsymbol{C}^* \boldsymbol{P}_{dyn} \boldsymbol{C}^{*T}) + \sigma_q^2 \boldsymbol{I} + \sigma_{pf}^2 \Big(&Tr[\Delta \boldsymbol{v}^* \Delta \boldsymbol{v}^{*T}]\boldsymbol{I} \\ &+ Tr[\boldsymbol{C}^* \boldsymbol{P}_{dyn} \boldsymbol{C}^{*T}]\boldsymbol{I} - \Delta \boldsymbol{v}^* \Delta \boldsymbol{v}^{*T} - \boldsymbol{C}^* \boldsymbol{P}_{dyn} \boldsymbol{C}^{*T}\Big) \end{aligned} \tag{6.35}$$

Here $\boldsymbol{P}_{dyn}$ represents the upper 12×12 corner of the error covariance matrix corresponding to both vehicles' position and velocity vectors. The value σ_{sf}^2 represents the variance of the scaling error, σ_{pf}^2 represents the variance of the pointing error, and σ_q^2 represents the variance of the quantization error (it is assumed that $\sigma_{q_1} = \sigma_{q_2} = \sigma_{q_3} = \sigma_q$). The $Tr[\;]$ operator indicates the *trace* function for a square matrix. The result of the formula in Equation 6.35 is a 3×3 covariance matrix correction that is added to the 3×3 orbiter velocity section of the full covariance matrix. In this manner, the covariance of the orbiter velocity is adjusted to account for the velocity error injected by performing maneuvers.

6.7 Extended Kalman Filter

6.7.1 Linearization About the State Estimate

Chapter 4 presented a derivation and summary of the standard Kalman filter, which is the basis of the Precomputed Gain filter implementation. With only a few modifications, the standard Kalman filter can be "upgraded" to the Extended Kalman filter (linearized around the state estimate rather than a nominal trajectory) and the propagation and updating equations are found to be the same as before, with new definitions for a few key matrices.

This section will briefly present the differences between the standard Kalman filter and the Extended Kalman filter, but most of the details of the implementation are left for the description of the flight computer.

In the derivation of the Extended Kalman filter, Equations 4.7 and 4.8 will be disregarded. These equations state that the true and estimated state vectors can be written as the sum of a nominal state and a perturbation to the true or estimated state. While still valid, this information will no longer prove useful here.

First, a new version of the state transition matrix, $\boldsymbol{\Phi}_{i+1,i}$, will be found for propagating the error covariance matrix. Instead of linearizing around the nominal state as shown in Equations 4.18 and 4.19, do so around the state estimate using Equation 4.9:

$$\boldsymbol{x}_{i+1} = \boldsymbol{g}(\hat{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_i, t_i, \Delta t) + \boldsymbol{w}_i \qquad (6.36)$$

The first order Taylor series expansion around the *estimated state* then results in:

$$\begin{aligned}\boldsymbol{x}_{i+1} &= \boldsymbol{g}(\hat{\boldsymbol{x}}_i, t_i, \Delta t) - \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_i}\right|_{\hat{\boldsymbol{x}}_i} \tilde{\boldsymbol{x}}_i + \emptyset(\tilde{\boldsymbol{x}}_i^2) + \boldsymbol{w}_i && (6.37)\\ &= \hat{\boldsymbol{x}}_{i+1} - \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_i}\right|_{\hat{\boldsymbol{x}}_i} \tilde{\boldsymbol{x}}_i + \emptyset(\tilde{\boldsymbol{x}}_i^2) + \boldsymbol{w}_i && (6.38)\end{aligned}$$

Rearranging Equation 6.37 and again utilizing Equation 4.9 (and neglecting the higher order terms), the following form is obtained for the propagation of the state error:

$$\begin{aligned}\hat{\boldsymbol{x}}_{i+1} - \boldsymbol{x}_{i+1} &= \boldsymbol{\Phi}_{i+1,i}\,\tilde{\boldsymbol{x}}_i - \boldsymbol{w}_i\\ \tilde{\boldsymbol{x}}_{i+1} &= \boldsymbol{\Phi}_{i+1,i}\,\tilde{\boldsymbol{x}}_i - \boldsymbol{w}_i && (6.39)\end{aligned}$$

Here $\boldsymbol{\Phi}_{i+1,i}$ is the new state transition matrix as defined in Equation 6.40 for the Extended filter:

$$\boldsymbol{\Phi}_{i+1,i} = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_i}\right|_{\hat{\boldsymbol{x}}_i} = \left.\frac{\partial \boldsymbol{x}_{i+1}}{\partial \boldsymbol{x}_i}\right|_{\hat{\boldsymbol{x}}_i} \qquad (6.40)$$

The error covariance matrix propagation equation can now be derived using the same method as before, arriving at the same result.

A similar technique as above can be applied to the measurement equation to derive the new measurement sensitivity matrix for the Extended Kalman filter. Instead of the linearization shown in Equations 4.26 and 4.27, perform the calculation where:

$$\boldsymbol{z}_i = \boldsymbol{h}(\hat{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_i, t_i) + \boldsymbol{v}_i \qquad (6.41)$$

If the state error is very small then this can be expanded in a first order Taylor series, resulting in the following:

$$\begin{aligned}\boldsymbol{z}_i &= \boldsymbol{h}(\hat{\boldsymbol{x}}_i, t_i) - \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_i}\right|_{\hat{\boldsymbol{x}}_i} \tilde{\boldsymbol{x}}_i + \emptyset(\tilde{\boldsymbol{x}}_i^2) + \boldsymbol{v}_i\\ \boldsymbol{z}_i - \hat{\boldsymbol{z}}_i &= -\left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_i}\right|_{\hat{\boldsymbol{x}}_i} \tilde{\boldsymbol{x}}_i + \emptyset(\tilde{\boldsymbol{x}}_i^2) + \boldsymbol{v}_i && (6.42)\end{aligned}$$

The partial derivative in Equation 6.42 is the new measurement sensitivity matrix, denoted $\boldsymbol{H}_i$. Recall from Section 6.2.1 that the formulation of the sensitivity matrix is identical between the Precomputed Gain and Extended Kalman filters, with $\boldsymbol{x}^*$ replaced by $\hat{\boldsymbol{x}}$ in Equation 4.31.

$$\left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_i}\right|_{\hat{\boldsymbol{x}}_i} = \boldsymbol{H}_i \tag{6.43}$$

Again the remainder of the derivation can be completed as before, with the same result for the optimal gain and measurement update to the covariance.

6.7.2 Covariance Update for Maneuvers

As with the Precomputed Gain strategy, it is necessary to update the covariance matrix if the current time is a maneuver time, $t_i = t_{man}$. A maneuver corrective covariance matrix, $\boldsymbol{P}_{\Delta v}$, was derived in Section 6.6.1 for the Precomputed Gain implementation as follows:

$$\begin{aligned}\boldsymbol{P}_{\Delta v} = \sigma_{sf}^2(\Delta\boldsymbol{v}^*\Delta\boldsymbol{v}^{*T} + \boldsymbol{C}^*\boldsymbol{P}_{dyn}\boldsymbol{C}^{*T}) + \sigma_q^2\boldsymbol{I} + \sigma_{pf}^2\big(Tr[\Delta\boldsymbol{v}^*\Delta\boldsymbol{v}^{*T}]\boldsymbol{I}\\ +Tr[\boldsymbol{C}^*\boldsymbol{P}_{dyn}\boldsymbol{C}^{*T}]\boldsymbol{I} - \Delta\boldsymbol{v}^*\Delta\boldsymbol{v}^{*T} - \boldsymbol{C}^*\boldsymbol{P}_{dyn}\boldsymbol{C}^{*T}\big)\end{aligned} \tag{6.35}$$

Noting that the derivation can be repeated for the Extended Kalman filter case excluding the corrective maneuvers, this result simplifies for the current implementation:

$$\boldsymbol{P}_{\Delta v} = \sigma_{sf}^2(\Delta\boldsymbol{v}^*\Delta\boldsymbol{v}^{*T}) + \sigma_q^2\boldsymbol{I} + \sigma_{pf}^2\left(Tr[\Delta\boldsymbol{v}^*\Delta\boldsymbol{v}^{*T}]\boldsymbol{I} - \Delta\boldsymbol{v}^*\Delta\boldsymbol{v}^{*T}\right) \tag{6.44}$$

Here the change in velocity $\Delta\boldsymbol{v}^*$, considered the "nominal" maneuver in the Precomputed Gain implementation, is simply the required maneuver calculated by the onboard targeting algorithms.

Chapter 7

Orbital Rendezvous Targeting Algorithms

In Chapter 6 a description was given of the rendezvous filters used in this study. In *this* chapter, the onboard targeting algorithms for both implementations will be presented, so that the full simulated flight software can be discussed in Chapter 8. The targeting algorithms presented in this chapter are not intended to be an exhaustive list, but simply the algorithms found appropriate for the study at hand.

Chapter 3 contains a description of the baseline trajectory and the basic building blocks used to create it, including: football orbit, coelliptic approach, and glide slope. In both the Extended and Precomputed Gain implementations, the baseline trajectory is assembled from these basic pieces, however it is done in vastly different ways.

The Precomputed Gain Kalman filter, utilizes a very simple onboard targeting strategy based on precomputed nominal maneuvers and maneuver gains. The objective of the onboard system is to stay close to the precomputed nominal trajectory, which can be determined on the ground by arbitrarily sophisticated or time-consuming processes, resulting only in the short list of nominal maneuvers to be performed at nominal times. All that is required of the onboard system, then, is a method to stay close to the desired nominal trajectory.

The Extended Kalman filter system, shown to be linearized in real time about the state estimate, has the flexibility to refine the rendezvous trajectory based on current conditions. The Extended system uses simple onboard targeting algorithms to shape the trajectory as the rendezvous progresses. Unlike the Precomputed Gain system, however, where maneuvers are all triggered at predetermined times, the Extended system can operate on any desired trigger based on the time or estimated state variables. For example: a coelliptic could be initiated when the estimated relative altitude matches a desired value; a glide slope could be initiated when the orbiter thinks it crosses the v-bar; a Lambert targeted maneuver could be initiated when the orbiter thinks it crosses an imaginary plane in the relative frame; or, a corrective maneuver could be performed on a football orbit at a desired time. In this chapter the maneuver triggers and targeting algorithms will be described together, for those algorithms used in the Extended system.[1]

Five different types of targeting algorithms are used to shape the trajectory in the Extended implementation (football control, coelliptic insertion, fast transfer insertion, glide slope, and Lambert targeting). These are each described in the following sections. It might be useful to refer to Figure 3-7 while considering these targeting strategies, as nominal execution will result in the baseline trajectory (that is, executing these algorithms on the true state in the absence of execution error, unmodeled accelerations, etc., will produce the reference trajectory).

A final remark about the targeting algorithms. It is entirely possible that the final required maneuver calculated by any of these algorithms is very small, in fact near or below the spherical execution error applied to every burn. In this case, it does not make sense to perform the maneuver, as it will be "lost in the noise." For this reason, all targeting algorithms are governed by a *small maneuver constraint*, a operator-tunable minimum maneuver magnitude, below which no maneuver is executed.

7.1 Lambert Targeting

The problem of determining the orbital trajectory between two position vectors, given the time of flight and the gravitational parameter μ, is well-known as Lambert's Prob-

[1]In the Precomputed Gain system, all maneuvers are triggered at predetermined times.

lem. Much effort has been focused on solving Lambert's Problem and avoiding issues of numerical precision and other challenges; indeed, Dr. Richard Battin dedicated a whole chapter of his astrodynamics text to the subject [2]. *Lambert targeting*, the technique of solving Lambert's Problem for a given scenario, finds extensive use as the catch-all of rendezvous trajectory design. Given the current chaser position and a desired relative position some specified time later, it is possible to use a Lambert targeting algorithm to connect the two points in the desired time.

The usefulness of Lambert targeting lies in its versatility. If the proper timing and relative positions are known, all of the trajectories from Section 3.2, such as the hop or coelliptic approach, can be generated using a Lambert routine. In fact, the glide slope technique described in Section 3.2.4 is actually a repeated application of Lambert targeting at appropriate intervals. The full Lambert targeting algorithm is used both as an onboard targeting algorithm for the Extended system as well as a tool for generating the nominal trajectory in the Precomputed Gain system.

The Lambert routine used in this analysis is based on unpublished notes by Stanley W. Shepperd of the Draper Laboratory, which are in turn based loosely on algorithms presented in Richard Battin's Astrodynamics text. The implementation in this study is actually quite complex, and a detailed description has been judged out of the scope of this work. The interested reader is referred to the many varied algorithms that exist in the literature, using Battin as a good starting point [2].

7.2 Linearized Targeting

A very simple and elegant solution to Lambert's problem can be found for the case of circular or near-circular orbits. This algorithm is based on the Clohessy-Wiltshire equations of relative motion for low eccentricity orbits. When written out in their entirety (see Reference [17]), it is clear that the CW/Hill equations can be written in the following form:

$$\begin{bmatrix} \boldsymbol{r}_{rel} \\ \boldsymbol{v}_{rel} \end{bmatrix}_{i+1}^{LVLH} = \boldsymbol{\Phi}_{i+1,i}^{CW} \begin{bmatrix} \boldsymbol{r}_{rel} \\ \boldsymbol{v}_{rel} \end{bmatrix}_{i}^{LVLH} \tag{7.1}$$

Here $\boldsymbol{\Phi}^{CW}_{i+1,i}$ is a 6×6 transition matrix that contains the terms of the CW/Hill equations evaluated for Δt, where $t_{i+1} = t_i + \Delta t$. The inertial relative position and velocity vectors, $\boldsymbol{r}^I_{rel}$ and $\boldsymbol{v}^I_{rel}$, respectively, are defined by:[2]

$$\boldsymbol{r}^I_{rel} = \boldsymbol{r}^I_{Orb} - \boldsymbol{r}^I_{OS} \tag{7.2}$$

$$\boldsymbol{v}^I_{rel} = \boldsymbol{v}^I_{Orb} - \boldsymbol{v}^I_{OS} + \boldsymbol{\omega}^I \times \boldsymbol{r}^I_{rel} \tag{7.3}$$

And the position and velocity vectors can be found in the LVLH frame using the coordinate transformation as follows:

$$\boldsymbol{r}^{LVLH}_{rel} = \boldsymbol{T}_{I \to LVLH} \boldsymbol{r}^I_{rel} \tag{7.4}$$

$$\boldsymbol{v}^{LVLH}_{rel} = \boldsymbol{T}_{I \to LVLH} \boldsymbol{v}^I_{rel} \tag{7.5}$$

Here $\boldsymbol{T}_{I \to LVLH} = \boldsymbol{T}^T_{LVLH \to I}$ is a rotation matrix from the inertial coordinate system to the LVLH coordinate system (see Section 3.1). The transition matrix can be written in block form as shown in Equation 7.6 and expanded into the position equation shown below (the expansion also gives a velocity equation, not shown):

$$\begin{bmatrix} \boldsymbol{r}_{rel} \\ \boldsymbol{v}_{rel} \end{bmatrix}^{LVLH}_{i+1} = \begin{bmatrix} \boldsymbol{\Phi}^{CW}_{rr} & \boldsymbol{\Phi}^{CW}_{rv} \\ \boldsymbol{\Phi}^{CW}_{vr} & \boldsymbol{\Phi}^{CW}_{vv} \end{bmatrix}_{i+1,i} \begin{bmatrix} \boldsymbol{r}_{rel} \\ \boldsymbol{v}_{rel} \end{bmatrix}^{LVLH}_{i} \tag{7.6}$$

$$\boldsymbol{r}^{LVLH}_{rel,i+1} = \boldsymbol{\Phi}^{CW}_{rr} \boldsymbol{r}^{LVLH}_{rel,i} + \boldsymbol{\Phi}^{CW}_{rv} \boldsymbol{v}^{LVLH}_{rel,i} \tag{7.7}$$

Rearranging yields an equation for the desired initial relative velocity, given the initial and terminal position vectors. All terms in Equation 7.8 are expressed in the rotating LVLH coordinate system.

$$\boldsymbol{v}^{LVLH}_{des} = \boldsymbol{\Phi}^{CW^{-1}}_{rv} (\boldsymbol{r}^{LVLH}_{rel,i+1} - \boldsymbol{\Phi}^{CW}_{rr} \boldsymbol{r}^{LVLH}_{rel,i}) \tag{7.8}$$

[2]Note that this definition of $\boldsymbol{r}_{rel}$ is different from the one shown in Equation 5.12. That is, the LOS vector in Chapter 5 points from the orbiter to the target, while the relative position vector for this consideration of relative motion is defined with its tail at the target.

The result from Equation 7.8 is then the desired targeting algorithm describing the required velocity to travel from one position to another in a specified time (in the relative coordinate frame). The velocity $\boldsymbol{v}_{des}^{LVLH}$ is the desired initial velocity in the relative coordinate frame to arrive at the desired relative position in the specified time. Typically, spacecraft maneuvers are referenced in some inertial coordinate system, so the additional step of transforming this velocity into the desired frame must be considered before using this algorithm.

7.2.1 Position Corrections

It is mentioned in the introduction above that, for the Precomputed Gain system, nominal maneuvers are performed at predetermined times to shape the nominal rendezvous trajectory. One can imagine that, in a 'true' rendezvous scenario, simply performing the nominal maneuvers at nominal times would be inadequate to guide the chaser to the target. Factors such as unmodeled accelerations and execution error cause the actual trajectory of the orbiter to be (sometimes substantially) different than the nominal, and the precomputed maneuvers would simply not be appropriate. For this reason, a strategy was devised to allow the orbiter to perform corrective velocity maneuvers in addition to the nominal maneuvers. This targeting algorithm results in a simple system of matrices that can be precomputed and stored, a required result for the Precomputed Gain implementation.

Given the position and velocity states of the orbiter and target vehicles, it is possible to create a targeting algorithm $\boldsymbol{g}_T(\boldsymbol{x}, \Delta t)$ such that:

$$\Delta \boldsymbol{v}_{des} = \boldsymbol{g}_T(\boldsymbol{x}, \Delta t) \tag{7.9}$$

Here $\Delta \boldsymbol{v}_{des}$ is the desired change in velocity, and the state $\boldsymbol{x}$ under current consideration is defined as:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r}_{OS}^T & \boldsymbol{v}_{OS}^T & \boldsymbol{r}_{Orb}^T & \boldsymbol{v}_{Orb}^T \end{bmatrix}^T \tag{7.10}$$

Of course, the chaser vehicle does not actually know the true state $\boldsymbol{x}$, but it does have the onboard estimate of the state, $\hat{\boldsymbol{x}}$. A true onboard targeting algorithm would take this current best estimate of the state and evaluate $\boldsymbol{g}_T(\hat{\boldsymbol{x}}, \Delta t)$, thereby determining the appropriate correction. Due to ITAR restrictions and the desire to keep the onboard system as simple as possible, the Precomputed Gain system cannot simply use the onboard algorithm. Instead, consider the definition of the state estimate given in Equation 4.8:

$$\hat{\boldsymbol{x}} = \boldsymbol{x}^* + \hat{\boldsymbol{\delta x}} \tag{4.8}$$

Here $\boldsymbol{x}^*$ is the nominal state vector and $\hat{\boldsymbol{\delta x}}$ is the estimated perturbation from the nominal. Using this definition for the estimate and expanding in a first order Taylor series around the nominal trajectory yields a matrix solution to the onboard targeting algorithm problem:

$$\begin{aligned} \boldsymbol{g}_T(\hat{\boldsymbol{x}}, \Delta t) &= \boldsymbol{g}_T(\boldsymbol{x}^* + \hat{\boldsymbol{\delta x}}, \Delta t) \\ &\simeq \boldsymbol{g}_T(\boldsymbol{x}^*, \Delta t) + \left.\frac{\partial \boldsymbol{g}_T}{\partial \hat{\boldsymbol{x}}}\right|_{\boldsymbol{x}^*} \hat{\boldsymbol{\delta x}} + \emptyset(\hat{\boldsymbol{\delta x}}_i^2) \\ \Delta \boldsymbol{v}_{des} &= \Delta \boldsymbol{v}^* + \boldsymbol{C}^* \hat{\boldsymbol{\delta x}} \end{aligned} \tag{7.11}$$

Here the matrix $\boldsymbol{C}^*$ is the *maneuver gain sensitivity matrix*, also symbolized $\boldsymbol{K}_{man}$, and formally defined in Equation 7.12.

$$\boldsymbol{C}^* = \left.\frac{\partial \boldsymbol{g}_T}{\partial \hat{\boldsymbol{x}}}\right|_{\boldsymbol{x}^*} \tag{7.12}$$

Note that in Equation 7.11, the term $\boldsymbol{g}_T(\boldsymbol{x}^*, \Delta t)$ is the hypothetical targeting algorithm evaluated on the nominal state, which is equivalent to the precomputed nominal change in velocity. Thus, the targeting strategy for the Precomputed Gain system is to perform the nominal maneuver plus a corrective maneuver calculated with the $\boldsymbol{C}^*$ matrix, which adjusts the nominal maneuver for the estimated deviation from the nominal state. Of course, the nominal maneuver could be zero, resulting in a strictly corrective maneuver at that predetermined time. In this case, $\Delta \boldsymbol{v}_{des} = \boldsymbol{C}^* \hat{\boldsymbol{\delta x}}$.

It should be re-emphasized that the expression derived in Equation 7.11 is a linearization of a targeting algorithm based on the nonlinear two-body dynamics of orbital motion. This approximation is only valid under the assumption that the estimated perturbation, $\hat{\boldsymbol{\delta x}}$, is *small* relative to the nominal state.

The question remains, of course, what is $\boldsymbol{C}^*$? An appropriate solution comes from a slight reformulation of the CW/Lambert algorithm above, such that the targeting algorithm $\boldsymbol{g}_T(\boldsymbol{x}, \Delta t)$ is written in terms of the required velocity in the relative frame. This can be done as follows.

Remembering from Equation 7.9 that the desired targeting algorithm $\boldsymbol{g}_T(\boldsymbol{x}, \Delta t)$ computes a *change* in velocity, an equation for $\Delta \boldsymbol{v}$ can be formulated as:

$$\Delta \boldsymbol{v}_{des}^{I} = \boldsymbol{T}_{LVLH \to I} \left(\boldsymbol{v}_{des}^{LVLH} - \boldsymbol{v}_{rel}^{LVLH} \right) \tag{7.13}$$

Again $\boldsymbol{T}_{LVLH \to I}$ is a rotation matrix from the LVLH coordinate system to the inertial coordinate system (see Section 3.1). The desired velocity in the LVLH coordinate system was shown in Equation 7.8, and the relative velocity in the LVLH frame was formulated in Equation 7.5. The final suitable form for the desired change in velocity can then be found in inertial coordinates, noting that a rotation matrix times its transpose is identity:

$$\begin{aligned} \Delta \boldsymbol{v}_{des}^{I} = \boldsymbol{T}_{LVLH \to I} & \left[\boldsymbol{\Phi}_{rv}^{-1} (\boldsymbol{r}_{i+1}^{LVLH} - \boldsymbol{\Phi}_{rr} \boldsymbol{T}_{I \to LVLH} (\boldsymbol{r}_{Orb}^{I} - \boldsymbol{r}_{OS}^{I})) \right] \\ & - \left[\boldsymbol{v}_{Orb}^{I} - \boldsymbol{v}_{OS}^{I} + \boldsymbol{\omega}^{I} \times (\boldsymbol{r}_{Orb}^{I} - \boldsymbol{r}_{OS}^{I}) \right] \end{aligned} \tag{7.14}$$

Finally, the $\boldsymbol{C}^*$ matrix can be computed by calculating the partial derivative of $\Delta \boldsymbol{v}_{des}^{I}$ with respect to the estimated state variables, according to Equation 7.12:

$$\begin{aligned} \boldsymbol{C}_{pos}^{*} &= \left. \frac{\partial \boldsymbol{g}_T}{\partial \hat{\boldsymbol{x}}} \right|_{\boldsymbol{x}^*} = \left. \frac{\partial \Delta \boldsymbol{v}_{des}^{I}}{\partial \hat{\boldsymbol{x}}} \right|_{\boldsymbol{x}^*} \\ &= \left. \begin{bmatrix} \boldsymbol{A} & \boldsymbol{I} & -\boldsymbol{A} & -\boldsymbol{I} \end{bmatrix} \right|_{\boldsymbol{x}^*} \end{aligned} \tag{7.15}$$

Here $\boldsymbol{A}$ is defined in Equation 7.16 and $\boldsymbol{I}$ is the identity matrix.

$$\boldsymbol{A} = \boldsymbol{T}_{LVLH \rightarrow I} \boldsymbol{\Phi}_{rv}^{-1} \boldsymbol{\Phi}_{rr} \boldsymbol{T}_{I \rightarrow LVLH} - \boldsymbol{\Omega}_{\otimes} \tag{7.16}$$

In taking the partial derivative it is worth noting that the term $\boldsymbol{r}_{i+1}^{LVLH}$, the desired future relative position in the LVLH frame, is not a function of any of the state variables and so disappears in the derivative. Also, the cross product $\boldsymbol{\omega} \times \boldsymbol{r}_{rel}$ can be expressed in terms of matrix multiplication using the skew-symmetric matrix (also known as the cross-product matrix) $\boldsymbol{\Omega}_{\otimes}$, as shown in Equation 7.17.

$$\boldsymbol{\omega} \times \boldsymbol{r}_{rel} = \boldsymbol{\Omega}_{\otimes} \boldsymbol{r}_{rel} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \boldsymbol{r}_{rel} \tag{7.17}$$

Equations 7.15 and 7.16 constitute the main targeting algorithm used in the Precomputed Gain implementation of the Kalman filter, the *linearized position correction*, where the $\boldsymbol{C}^*$ matrix is computed on the ground. Several important points should be reiterated about this algorithm. First, recall that the state transition matrix used in Equation 7.16 is computed for t_i to t_{i+1}, meaning that the transfer time for the maneuver must be specified. Additionally, recall that this is a *linearized* Lambert routine, derived under the assumption of circular or near-circular orbits.

It should be noted that after a corrective maneuver is performed (that is, a burn has been executed to correct a perturbation in the state, and the targeted point has been approximately reached), the chaser spacecraft will *not* have the velocity at the target point that it would have on the nominal trajectory. A second maneuver is required at the target point to correct the velocity, or the chaser will again deviate from the nominal path. This idea is discussed further in the section to follow.

7.2.2 Velocity Corrections

It is mentioned above that after a linearized position correction maneuver is performed, the chaser spacecraft will not have the velocity at the target point that it would have on the nominal trajectory. The primary method used in the Precomputed Gain implementation to handle this condition is to perform another corrective maneuver at the target point by aiming for a desired position further in the trajectory. This strategy is employed knowing that in real rendezvous scenarios with execution error and imperfect knowledge, the target point was probably not exactly reached, and the velocity and position errors can both be re-corrected simultaneously by aiming for a point further in the trajectory.

There are occasions, however, where it is not desirable to target to a future position, and the designer just wants to correct the velocity error. An example might be the last maneuver of the trajectory. In this case, the required inertial change in velocity from Equation 7.13 is simply:

$$\begin{aligned}\Delta \boldsymbol{v}^I_{des,v} = \boldsymbol{T}_{LVLH\rightarrow I}\big[&\boldsymbol{T}_{I\rightarrow LVLH}(\boldsymbol{v}^I_{Orb,des} - \boldsymbol{v}^I_{OS,des} - \boldsymbol{\omega}^I \times (\boldsymbol{r}^I_{Orb,des} - \boldsymbol{r}^I_{OS,des})) \\ &-\boldsymbol{T}_{I\rightarrow LVLH}(\boldsymbol{v}^I_{Orb} - \boldsymbol{v}^I_{OS} - \boldsymbol{\omega}^I \times (\boldsymbol{r}^I_{Orb} - \boldsymbol{r}^I_{OS}))\big] \quad (7.18)\end{aligned}$$

Taking the partial derivative yields the following form for $\boldsymbol{C}^*_{vel}$ (noting that the product of a rotation matrix and its transpose is the identity matrix):

$$\begin{aligned}\boldsymbol{C}^*_{vel} &= \left.\frac{\partial \boldsymbol{g}_T}{\partial \hat{\boldsymbol{x}}}\right|_{\boldsymbol{x}^*} = \left.\frac{\partial \Delta \boldsymbol{v}^I_{des,v}}{\partial \hat{\boldsymbol{x}}}\right|_{\boldsymbol{x}^*} \\ &= \left.\begin{bmatrix} -\boldsymbol{\Omega}_\otimes & \boldsymbol{I} & \boldsymbol{\Omega}_\otimes & -\boldsymbol{I}\end{bmatrix}\right|_{\boldsymbol{x}^*} \quad (7.19)\end{aligned}$$

This formulation for the maneuver gain matrix is the final element of the set of targeting algorithms used in this implementation. It is used only occasionally to perform a velocity-only correction to the orbiter state.

7.3 Football Orbit Control

The reference rendezvous trajectory begins as described in Section 3.3.3, where the orbiter has the initial inertial position and velocity to enter a 2×1 km football orbit at

5 km downrange. Of course, in a real scenario the initial conditions are not ideal, and perturbations such as unmodeled accelerations could cause the football orbit to drift, so corrective maneuvers are required to maintain the holding football orbit. It was noted in Chapter 3 that holding football orbits are often used to initialize rendezvous trajectories while the orbiter 'gets its bearings.' Thus, the key aspect of the football orbit is not necessarily its size or center point, but that it does not drift towards the target in those first few orbits. For this reason, the semi-major axis matching strategy suggested in Equation 3.17 is used for corrective maneuvers on the football orbit (remember that $a_{Orb,des} = a_{OS}$).

$$\boldsymbol{v}_{des} = \frac{\boldsymbol{v}_{Orb}}{|\boldsymbol{v}_{Orb}|}\sqrt{\frac{2\mu}{|\boldsymbol{r}_{Orb}|} - \frac{\mu}{a_{Orb,des}}} \tag{3.17}$$

At pre-determined times,[3] the targeting algorithm computes the estimated semi-major axis of the target and the semi-major axis of the chaser, and performs a maneuver in the direction of the orbiter's velocity vector to match them. This guarantees that the two vehicles have the same period, so that regardless of the shape of the football orbit, there will be no drift.

7.4 Coelliptic Insertion

Section 3.2.1 describes many of the properties of coelliptics frequently used to calculate the required change in velocity – while all valid approaches, none of those were used here. The problem is in the Sun angle and target shading constraints, and the relative downrange velocity described in Section 3.2.1. Because the altitude of the top of the football orbit can vary significantly, the altitude of the coelliptic will also, and a coelliptic calculated according to Equation 3.10 could have a large range of downrange closing rates. It is possible, then, to cover the distance to the target so fast that future proximity operations occur during periods of measurement constraints

[3]Used to force corrective maneuvers to not occur in, or immediately prior to, a Sun angle or target shadowing constraint.

that were otherwise supposed to be navigated during the coelliptic. It is also possible to cover the distance so slowly that constraints on the glide slope become misaligned. Instead of the strategy from Chapter 3, then, an algorithm is used that results in an approximate coelliptic. First, the mean motion of the target is calculated:

$$\omega_{OS} = \sqrt{\frac{\mu}{a_{OS}^3}} \tag{7.20}$$

The desired mean motion for the chaser is then the mean motion of the target less some small amount, causing a small difference between the periods and therefore a closing rate:

$$\omega_{Orb} = \omega_{OS} - \Delta\omega \tag{7.21}$$

The small change in mean motion, $\Delta\omega$, is calculated as the difference between the current downrange position, $r_{rel,x,i}$ and the desired downrange position $r_{rel,x,des}$, divided by the magnitude of the target's position and the difference between the current time and the *absolute* desired end time:

$$\Delta\omega = \frac{r_{rel,x,i} - r_{rel,x,des}}{r_{OS}(t_f - t_i)} \tag{7.22}$$

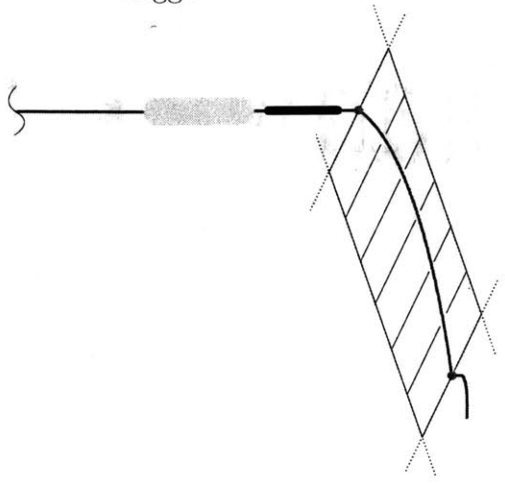
This formulation allows the operator to specify a time that is known to be after the Sun angle and target shadowing constraints, as well as a desired downrange position. It is not necessary to impose any constraints on the altitude position. From the new mean motion of the orbiter, ω_{Orb}, the required semi-major axis of the orbiter can then be calculated.

$$a_{Orb} = \sqrt[3]{\frac{\mu}{\omega_{Orb}^2}} \tag{7.23}$$

The required velocity magnitude can be easily calculated from the semi-major axis, and this velocity is then applied in the direction of the chaser's velocity vector, after removing the out-of-plane component.

Due to the timing issues with the constraints described above, a trigger was also employed that would start the coelliptic when the chaser crossed the desired coelliptic altitude on the third orbit, or reached the top of the football if 500 meters of altitude

Figure 7-1: Trigger to initialize fast transfer



was never obtained. Logic was included in the algorithm to accomodate any ‘jogs’ in the position estimate that might falsely indicate the top of the football. This trigger strategy prevents coelliptics triggered from large football orbits from having very exaggerated ‘dips’ required to slow the chaser down and reach the desired downrange distance in the desired time.

7.5 Fast Transfer Insertion

After traveling along the coelliptic through the Sun angle and target shadowing constraints (see Figure 3-7), a trigger is required to start the transfer down to the v-bar. In this case a simple geometric trigger is used. The transfer is initiated when the orbiter crosses the plane defined by: the nominal starting point of the transfer, the nominal ending point of the transfer, and the crosstrack ($\hat{\boldsymbol{y}}$) axis of the LVLH coordinate system. This configuration is illustrated in Figure 7-1. The maneuver in this case is a Lambert maneuver targeted to the desired point on the v-bar (100 m downrange) in 30 min.

7.6 Glide Slope

After the fast transfer down to the v-bar, the bounce and glide slope shown in Figure 3-8 must be initiated by another trigger. The trigger used in this case is based on the relative altitude. As the chaser is traveling down the fast transfer towards the v-bar, the algorithm triggers the maneuver at the time step where the chaser is closest to the v-bar, either above or below it. The bounce and glide slope maneuvers themselves are performed by the Lambert targeting algorithm, where the next maneuver is targeted ahead by a certain downrange offset (and to the v-bar), in the desired amount of time. Successive maneuvers are triggered by the elapsed transfer time, or the prediction of a descending v-bar crossing at the next time step. By adjusting these parameters, the desired relative rate along the v-bar can be adjusted. The bounce shown in Figure 3-8 is calculated in the same way as the rest of the glide slope (using Lambert targeting), but with different values for range and transfer time.

[This page intentionally left nearly blank.]

Chapter 8

Rendezvous Navigation and Targeting - Flight Algorithms

8.1 Precomputed Gain Algorithms

8.1.1 Rationale and Motivation

As discussed in Section 1.2, CNES participation in the MSR demonstration and the lack of a dedicated processor forced JPL to formulate a unique Precomputed Gain Kalman filter for the PREMIER-07 mission. ITAR restrictions limiting the transfer of technology out of the United States forced JPL to split the algorithm in half, keeping the guidance and navigation algorithms to themselves and sending a precomputed set of navigation filter and maneuver gains to the orbiter.

The resulting Precomputed Gain system uses a nominal rendezvous trajectory around which measurement gain, maneuver gain, and state transition matrices are calculated and stored. These matrices can then be uploaded to the orbiter and the rendezvous mission performed, simply by allowing the orbiter to calculate maneuver and state updates through matrix multiplication. The onboard system is simple and elegant, requiring perhaps a substantial amount of data storage but very little processing power. The Precomputed Gain system undoubtedly has merits, and thus is worthy of honest study here to quantify its value.

8.1.2 Configuration

The onboard navigation and targeting algorithms require a large amount of precomputed data from the ground, including: measurement gain matrices for each measurement time, maneuver gain matrices for each maneuver time, nominal changes in velocity at maneuver times, and state propagation matrices for both vehicles at each simulation time step. This data is generated around a reference trajectory as defined in advance by the operator. The following is a description of how each is generated for the Precomputed Gain algorithm, drawing heavily on the descriptions of the filter and targeting algorithms given in Chapters 6 and 7. Equations describing the implementation of these matrices (that is, how the flight computer uses them to perform autonomous navigation) are explained in Section 8.1.3.

Reference Trajectory

The required inertial nominal reference trajectory is generated by propagating the standard 2-body equations of motion for the target and the chaser. Both vehicles are given initial conditions based on the orbital elements described in Section 3.3.1 and the desired initial relative position and velocity. The trajectory is propagated between fixed initial and final times, t_0 to t_f, using a fourth order Runge-Kutta integrator with fixed time step. While this thesis assumes a spherical planet in the equations of motion, the ability to consider the J2 oblateness term exists in the implementation and could be considered in future work.

Maneuvers in the reference trajectory are performed at fixed times, determined by the operator based on experience and the desired properties of the relative trajectory. Relative maneuvers to achieve football orbits, coelliptic approaches, hops, and glide slope approaches, are performed in the inertial frame using the techniques described in Section 3.2. The required nominal changes in velocity are applied to the chaser vehicle at the desired maneuver times, and the maneuvers themselves stored for later use.

State Transition Matrices

In the simple Precomputed Gain implementation, where the flight computer is desired to be as simple as possible, a numerical integration routine is not provided to propagate the state estimate. Instead, state transition matrices are generated on the ground for both vehicles that transition the perturbations in position and velocity at time t_i to time t_{i+1}. These matrices are calculated for every time step, uploaded to the orbiter, and used at the appropriate time to propagate the state error estimate. The state transition matrices are calculated for this implementation according to the discussion in Section 6.1.1.

Maneuver Gain Matrices

Again trying to keep the onboard flight computer simple, and restricted by ITAR regulations, a very simple "targeting algorithm" was devised to keep the chaser near the nominal trajectory. For each corrective maneuver time, t_{man}, a gain matrix is computed that maps the estimated dispersion from the nominal state to the change in velocity required to reach the nominal state at a reference time. These gain matrices are the $\boldsymbol{C}^*$ matrices described in Sections 7.2.1 and 7.2.2 on the linearized targeting algorithms. A maneuver gain matrix is calculated for every corrective maneuver time, stored, and uploaded to the chaser for use in the flight computer.

Kalman Gain Matrices

The final piece of precomputed information required by the chaser vehicle is a set of measurement gain matrices corresponding to the measurement times. These measurement gain matrices, or Kalman gain matrices, allow the flight computer to calculate an update to the vehicle state using measurement data, as described in Section 8.1.3. This section will describe how the Kalman gain matrices are calculated for this implementation. The discussion of the "standard" Kalman filter in Chapter 4 is provided largely to reduce the amount of description required here, and so will be referenced heavily throughout this section.

The easiest way to discuss the calculation of the measurement gain matrices is as an algorithm which requires the following steps:

1. Initialize error covariance matrix
2. Propagate error covariance matrix
3. Increase error covariance matrix for nominal maneuvers
4. Calculate Kalman gain matrix
5. Update error covariance matrix with measurement
6. Return to step 2 at the next time step

Covariance Initialization It is necessary to initialize the error covariance matrix at time t_0. This procedure is common between both filter implementations in this study, and was described in Section 6.3 in detail.

Propagate Covariance As shown in Equation 4.23 from Section 4.2.2, the error covariance matrix is propagated at every time step using the state transition matrix (based on the nominal trajectory) from t_i to t_{i+1} and the process noise covariance matrix, $\boldsymbol{Q}_i$:

$$\boldsymbol{P}_{i+1} = \boldsymbol{\Phi}_{i+1,i}\,\boldsymbol{P}_i\,\boldsymbol{\Phi}^T_{i+1,i} + \boldsymbol{Q}_i \tag{4.23}$$

This propagation step requires the process noise covariance matrix, $\boldsymbol{Q}_i$, and the transition matrix $\boldsymbol{\Phi}_{i+1,i}$ for the whole state vector. The calculation for these two parameters is common between the Precomputed Gain and Extended Kalman filter implementations, and was described in Section 6.4 in detail (recall that the Extended system is linearized around the state estimate).

Covariance Update for Maneuvers When a maneuver is performed on the reference trajectory, a velocity error may be introduced into the orbiter state that should be reflected in the covariance matrix. The technique to update the covariance matrix for the Precomputed Gain implementation was discussed in Section 6.6.1, and the final result is repeated here:

$$\boldsymbol{P}_{\Delta v} = \sigma_{sf}^2(\Delta \boldsymbol{v}^* \Delta \boldsymbol{v}^{*T} + \boldsymbol{C}^* \boldsymbol{P}_{dyn} \boldsymbol{C}^{*T}) + \sigma_q^2 \boldsymbol{I} + \sigma_{pf}^2 \left(Tr[\Delta \boldsymbol{v}^* \Delta \boldsymbol{v}^{*T}]\boldsymbol{I} + Tr[\boldsymbol{C}^* \boldsymbol{P}_{dyn} \boldsymbol{C}^{*T}]\boldsymbol{I} - \Delta \boldsymbol{v}^* \Delta \boldsymbol{v}^{*T} - \boldsymbol{C}^* \boldsymbol{P}_{dyn} \boldsymbol{C}^{*T}\right) \quad (6.35)$$

Calculate Kalman Gain As shown in Equation 4.41 in Section 4.3, the optimal Kalman gain is calculated according to:

$$\boldsymbol{K}_i = \boldsymbol{P}_i \boldsymbol{H}_i^T (\boldsymbol{H}_i \boldsymbol{P}_i \boldsymbol{H}_i^T + \boldsymbol{R}_i)^{-1} \quad (4.41)$$

That is, at time t_i, the following values are required to compute the optimal gain: the covariance matrix $\boldsymbol{P}_i$; the measurement sensitivity matrix $\boldsymbol{H}_i$; and, the measurement noise covariance matrix $\boldsymbol{R}_i$. The method to compute the fully propagated and corrected covariance matrix comes from the algorithm above. The required calculation for the measurement noise covariance matrix is calculated as described in Section 6.5 and is repeated here:

$$\boldsymbol{R}_i = \begin{bmatrix} \sigma_e^2 & 0 & 0 \\ 0 & \sigma_\alpha^2 & 0 \\ 0 & 0 & \sigma_r^2 \end{bmatrix} \quad (6.28)$$

The measurement sensitivity matrix is defined in Equation 6.20 of Section 6.2.1.[1] Note that when the measurement sensitivity matrix of Equation 6.20 is evaluated around the nominal state in the Precomputed Gain system, several interesting simplifications can be made. First, the partial derivatives of the LOS unit vectors simplify as follows, noting that the nominal azimuth and elevation

[1]Note again that, for the Precomputed Gain implementation, the measurement sensitivity matrix is evaluated around the nominal state, $\boldsymbol{x}^*$.

angles are zero and their biases are nominally zero:

$$\boldsymbol{p}_e^c = \frac{\partial \hat{\boldsymbol{i}}_{LOS}^c}{\partial e} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \hat{\boldsymbol{i}}_z^c \tag{8.1}$$

$$\boldsymbol{p}_\alpha^c = \frac{\partial \hat{\boldsymbol{i}}_{LOS}^c}{\partial \alpha} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \hat{\boldsymbol{i}}_y^c \tag{8.2}$$

Similarly, the $\cos^2(e)$ terms in the sensitivity matrix go to unity. The nominal LOS vector is in the x-direction of the camera frame, or $\hat{\boldsymbol{i}}_{LOS}^c = \hat{\boldsymbol{i}}_x^c$. The resulting simplified measurement sensitivity matrix follows:

$$\boldsymbol{H}|_{\boldsymbol{x}^*} = \begin{bmatrix} \frac{\boldsymbol{p}_e^I}{|\boldsymbol{r}_{rel}|} & \frac{\boldsymbol{p}_\alpha^I}{|\boldsymbol{r}_{rel}|} & \hat{\boldsymbol{i}}_{LOS}^I \\ 0 & 0 & 0 \\ -\frac{\boldsymbol{p}_e^I}{|\boldsymbol{r}_{rel}|} & -\frac{\boldsymbol{p}_\alpha^I}{|\boldsymbol{r}_{rel}|} & -\hat{\boldsymbol{i}}_{LOS}^I \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ \hat{\boldsymbol{i}}_y^c & -\hat{\boldsymbol{i}}_z^c & 0 \\ \hat{\boldsymbol{i}}_y^c & -\hat{\boldsymbol{i}}_z^c & 0 \end{bmatrix}_{\boldsymbol{x}^*}^T \tag{8.3}$$

Having now described methods for calculating all of the required terms in Equation 4.41, the *Kalman gain can be calculated at time* t_i, and stored for later use.

Update Covariance The goal of this entire section was the Kalman gain matrix, which was calculated in the step above. Before concluding, however, one final calculation should be included to fully represent the Precomputed Gain Kalman filter loop, and keep the algorithm going for the next time step. Having taken a measurement at time t_i and calculated the optimal gain, the error covariance matrix should be updated to represent the improved information about the state that the measurement brings. Knowing the optimal gain, the covariance update is a simple calculation using Equation 4.42:

$$\boldsymbol{P}_i^+ = (\boldsymbol{I} - \boldsymbol{K}_i\boldsymbol{H}_i)\boldsymbol{P}_i^- \tag{4.42}$$

Every term in Equation 4.42 is known, and the updated covariance matrix may be computed for use in the next time step.

8.1.3 Flight Software

The design of the JPL/CNES Precomputed Gain navigation and targeting algorithms is such that the bulk of the computation occurs on the ground. The effect of this division is that the onboard algorithm is actually quite simple, as shown in Figure 8-1.

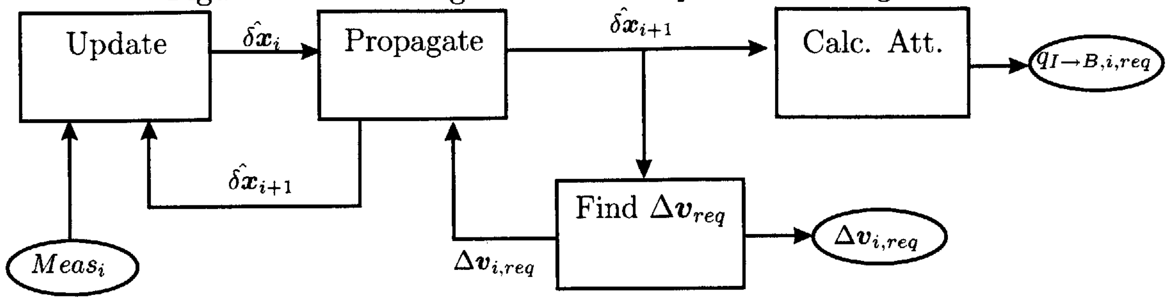
The four main computational components of the onboard algorithm as listed in the figure are:

1. Update the vehicle states with measurements
2. Find the required $\Delta\boldsymbol{v}$
3. Propagate the vehicle states
4. Calculate required attitude

The following sections describe these components as implemented in the simulation of the CNES/JPL Precomputed Gain algorithms.

Update Vehicle States With Measurement At a given measurement time, $t_i = t_{meas}$, the spacecraft must be able to take the measurement from its sensors and

Figure 8-1: Flow diagram for Precomputed Gain Algorithms


Update
$\hat{\delta x}_i$
Propagate
$\hat{\delta x}_{i+1}$
Calc. Att.
$q_{I\rightarrow B,i,req}$
$\hat{\delta x}_{i+1}$
Find $\Delta \boldsymbol{v}_{req}$
$\Delta \boldsymbol{v}_{i,req}$
$Meas_i$
$\Delta \boldsymbol{v}_{i,req}$

use it to update the onboard state estimate. This is done using the precomputed Kalman gain (measurement gain) matrix. As described above, the measurement gain matrix maps the deviation of the actual measurement from the expected measurement to a correction to the estimated state. This relation is shown in Equation 8.4.

$$\hat{\delta \boldsymbol{x}}_i^+ = \hat{\delta \boldsymbol{x}}_i^- + \boldsymbol{K}_{meas,i}\, \boldsymbol{\rho}_i \tag{8.4}$$

Here, $\hat{\delta \boldsymbol{x}}_i^+$ is the flight computer's updated estimate of the dispersion from the nominal state at time t_{meas}. The variable $\hat{\delta \boldsymbol{x}}_i^-$ is the flight computer's estimate of the dispersion from the nominal state, also at time t_{meas}, but before the measurement has been incorporated at that time. As above, $\boldsymbol{K}_{meas,i}$ is the measurement gain matrix precomputed for time t_{meas}, and $\boldsymbol{\rho}_i$ is the difference between the actual measurement and the estimated measurement at t_{meas}, called the residual. The residual is calculated as in Equation 8.5.

$$\boldsymbol{\rho}_i = \boldsymbol{z}_i - \hat{\boldsymbol{z}}_i \tag{8.5}$$

Here $\boldsymbol{z}_i$ is the "actual" measurement computed by the environment model for t_{meas}, and $\hat{\boldsymbol{z}}_i$ is the flight computer's estimated value of what the measurement should be. In this implementation, three measurements are used based on an optical observation of the target from the chaser vehicle: elevation, azimuth, and range. The estimated values of the elevation and azimuth are always zero, since the pseudo 6-DOF simulator assumes it can always rotate the chaser vehicle to train the camera boresight on the target. The actual measurement and the estimated measurement are written as:

$$\boldsymbol{z}_i = \begin{bmatrix} e_i \\ \alpha_i \\ r_i \end{bmatrix}, \hat{\boldsymbol{z}}_i = \begin{bmatrix} \hat{e}_i \\ \hat{\alpha}_i \\ \hat{r}_i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \hat{r}_i \end{bmatrix} \tag{8.6}$$

Here e_i, α_i, and r_i are the actual elevation, azimuth, and range measurements at measurement time t_{meas}, respectively. The estimated range value at t_{meas} is calculated as shown in Equations 8.7 through 8.9.

$$\hat{r}_i = \left|(\boldsymbol{r}^*_{OS,i} + \hat{\delta\boldsymbol{r}}^-_{OS,i}) - (\boldsymbol{r}^*_{Orb,i} + \hat{\delta\boldsymbol{r}}^-_{Orb,i})\right| \tag{8.7}$$

$$\boldsymbol{x}^*_{OS} = \begin{bmatrix} \boldsymbol{r}^*_{OS} \\ \boldsymbol{v}^*_{OS} \end{bmatrix}, \boldsymbol{x}^*_{Orb} = \begin{bmatrix} \boldsymbol{r}^*_{Orb} \\ \boldsymbol{v}^*_{Orb} \end{bmatrix} \tag{8.8}$$

$$\hat{\delta\boldsymbol{x}}_{OS} = \begin{bmatrix} \hat{\delta\boldsymbol{r}}_{OS} \\ \hat{\delta\boldsymbol{v}}_{OS} \end{bmatrix}, \hat{\delta\boldsymbol{x}}_{Orb} = \begin{bmatrix} \hat{\delta\boldsymbol{r}}_{Orb} \\ \hat{\delta\boldsymbol{v}}_{Orb} \end{bmatrix} \tag{8.9}$$

Here $\boldsymbol{x}^*$ is the full nominal state as described in the section above, $\boldsymbol{r}^*$ is the nominal inertial position vector, and $\boldsymbol{v}^*$ is the nominal inertial velocity vector.

Note also that, as described above, it is possible for the measurements generated by the environment model to be out of range due to any of several constraints (FOV angle, sun angle, target eclipse). When this happens, the environment flags the measurement with the constraint violation and passes that to the flight computer, rather than the actual invalid measurement. Thus, when the flight computer is updating vehicle states with measurements, it checks for the invalid measurement flags, and skips the update if the measurement is bad. That is, for an invalid measurement:

$$\hat{\delta\boldsymbol{x}}^+_i = \hat{\delta\boldsymbol{x}}^-_i \tag{8.10}$$

Likewise, if the current time step t_i is not a measurement time, no update is made and Equation 8.10 applies. In summary, the result of the state update with a camera measurement is a new estimate of the dispersion from the nominal state, $\hat{\delta x}_i^+$, still at the measurement time t_{meas}.

After the discussion above, the attentive reader might have noticed that not all of the states in the state vector have been updated or used in the measurement estimate. In fact, the only states updated or used for the measurement estimate (or propagation of vehicle dynamics) were the first 12 of 20, the position and velocity of both vehicles. The measurement angle biases, static alignment error, and inertial attitude states are not used at all. How, then, can this be considered a 20 state filter? The answer is that all 20 states are used in the calculation of the Kalman gain matrices on the ground, "puffing up" the error covariance matrix and reducing the weight of certain states at certain times in the optimal gain. Even though these last 8 states are not *estimated*, their inclusion as error sources in the gain calculation is essential to proper filter performance.

The goal of the Precomputed Gain implementation was to create as simple a system as possible onboard the CNES orbiter due to ITAR restrictions and a simple onboard computer, so estimation of the last 8 states was not initially suggested. It was decided that a 12-state measurement estimate would be attempted first, and that the remaining states would be implemented if the system did not perform well. As will be seen in Chapter 9, however, the Precomputed Gain system did perform adequately with the current implementation, and so it remains with only a 12-state estimate. Implementation of the full state vector estimate is thus left to future work.

Find Required $\Delta \boldsymbol{v}$ At a given corrective maneuver time, t_{man}, the spacecraft must be able to take the flight computer's estimate of the dispersion from the nominal state, and calculate the maneuver required to drive back to the nominal trajectory. The flight computer performs this task in a very simple fashion, as shown in Equation 8.11.

$$\Delta \boldsymbol{v}_{req,i} = \Delta \boldsymbol{v}_i^* + \boldsymbol{K}_{man,i} \hat{\delta \boldsymbol{x}}_i \tag{8.11}$$

Here $\Delta \boldsymbol{v}_i^*$ is the nominal maneuver required at t_{man} (if any), $\boldsymbol{K}_{man,i}$ is the maneuver gain matrix at t_{man}, and $\hat{\delta \boldsymbol{x}}_i$ is the flight computer's best estimate of the dispersion from the nominal state. Thus, the output $\Delta \boldsymbol{v}_{req,i}$ is a combination of the nominal maneuver and the required corrective maneuver, and can be passed to the environment model and incorporated into the integration of the dynamics.

It is possible that the required maneuver that results from Equation 8.11 is quite small, and indeed well below the chaser vehicle's ability to execute it reliably (particularly with the execution error model described above). For this reason, the simulation has the ability to compare the magnitude of $\Delta \boldsymbol{v}_{req,i}$ to a user-specified lower bound, below which the maneuver is not worth attempting. For example, a typical small maneuver threshold would be 2 mm/s. If the magnitude of the result from Equation 8.11 is then less than 2 mm/s, the simulation sets $\Delta \boldsymbol{v}_{req,i}$ to the zero vector and flags an alert for the user. In this way, potential maneuvers that would be lost "in the noise" are never attempted.

Propagate Vehicle States At every simulation time step t_i, the filter must propagate the state from time t_i to time t_{i+1}. This too can be done with straightforward matrix multiplication, given the precomputed state propagation matrices for the target and chaser as described above, and knowledge of any corrective maneuver.

It should be noted that the required maneuver calculated above, $\Delta \boldsymbol{v}_{req,i}$, includes both the corrective maneuver and the nominal maneuver at time t_{man}, if a nominal maneuver is required. This calculation results in a desired/commanded value sent to the environment model. The flight propagation algorithms, however, assumes that the nominal maneuvers have been performed as required, and thus only needs to know about the required corrective part. The first step in the algorithm is then to remove the nominal maneuver from $\Delta \boldsymbol{v}_{req,i}$ as shown below.

$$\Delta \boldsymbol{v}_i = \Delta \boldsymbol{v}_{req,i} - \Delta \boldsymbol{v}_i^* \tag{8.12}$$

An inherent assumption here is that the maneuver requested by the flight computer, $\Delta \boldsymbol{v}_{req,i}$, is the maneuver that gets executed by the spacecraft. This, of course, is not true, as some unmeasured execution error is applied in the environment model, so a small amount of velocity navigation error gets injected in this fashion at each maneuver.

Since the target vehicle performs no maneuvers, propagating its state is simply a matter of multiplying the state transition matrix for the target vehicle by the dispersion from the target's nominal state. For the chaser vehicle, the state transition matrix must be applied to the vehicle state together with the required maneuver.

$$\widehat{\delta \boldsymbol{x}}_{OS,i+1} = \boldsymbol{\Phi}_{OS,i+1,i}\, \widehat{\delta \boldsymbol{x}}_{OS,i} \tag{8.13}$$

$$\widehat{\delta \boldsymbol{x}}_{Orb,i+1} = \boldsymbol{\Phi}_{Orb,i+1,i} \left[\widehat{\delta \boldsymbol{x}}_{OS,i} + \begin{bmatrix} 0 \\ \Delta \boldsymbol{v}_i \end{bmatrix} \right] \tag{8.14}$$

Here $\boldsymbol{\Phi}_{i+1,i}$ is the state transition matrix for the appropriate vehicle at time step t_i, and $\widehat{\delta \boldsymbol{x}}_{i+1}$ is the propagated state estimate, valid for the next simulation time step t_{i+1}. The actual vehicle state estimates, rather than the dispersion from the nominal states, can be readily calculated by adding the nominal state and the dispersion from the nominal.

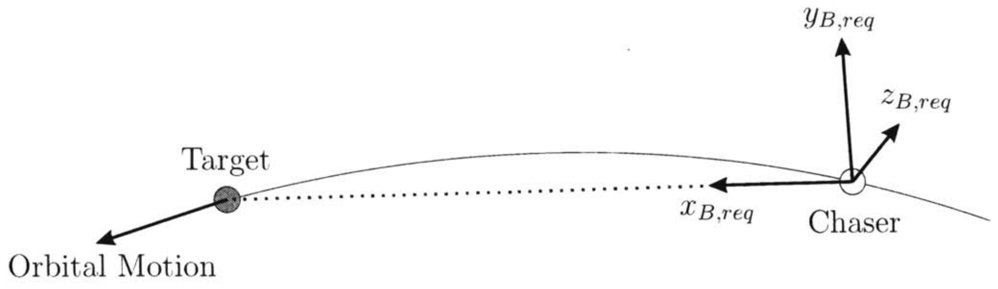
$$\hat{\boldsymbol{x}}_{OS,i+1} = \boldsymbol{x}^*_{OS,i+1} + \widehat{\delta \boldsymbol{x}}_{OS,i+1} \tag{8.15}$$

$$\hat{\boldsymbol{x}}_{Orb,i+1} = \boldsymbol{x}^*_{Orb,i+1} + \widehat{\delta \boldsymbol{x}}_{Orb,i+1} \tag{8.16}$$

The full state estimate is then used to calculate the required attitude at time t_{i+1} as described below. This attitude requirement, along with the required change in velocity, $\Delta \boldsymbol{v}_{req}$, are passed to the environment model and used to update the true vehicle position, velocity, and attitude.

Calculate Required Attitude The "required attitude" for the chaser vehicle is the attitude that provides boresight pointing towards the chaser vehicle, based on the flight computer's estimate of the both vehicle states. The attitude is

Figure 8-2: Camera frame definition



represented in the simulator as a quaternion describing the rotation from the inertial frame to the body frame of the spacecraft. In order to calculate this quaternion, a rotation matrix for the inertial to body transformation is first formulated, then converted into the corresponding quaternion. The inertial to body transformation, $\boldsymbol{T}_{I\rightarrow B}$, is calculated as shown in Section 3.1.1, using Equations 3.4 and 3.6. A graphical representation of this coordinate system is shown in Figure 8-2. The simulator converts the rotation matrix $\boldsymbol{T}_{I\rightarrow B}$ into a quaternion $q_{I\rightarrow B}$, and this is passed to the environment module as the desired attitude.

8.2 Extended Kalman Filter & Onboard Targeting

8.2.1 Rationale and Motivation

Chapter 1 talks briefly about the development of the Precomputed Gain system for the CNES PREMIER-07 orbiter, and the subsequent departure of CNES from the MSR timeline. In the absence of ITAR restrictions and a simple onboard computer requiring the Precomputed Gain implementation, there are other more conventional systems that can be considered for the MSR mission. One familiar example that would make a good candidate is a full onboard 'extended' Kalman filter, where the dynamics and measurements are linearized about the current best estimate of the state, rather than a nominal trajectory.

The Extended Kalman filter is a robust algorithm that, coupled with flexible targeting algorithms allowed by its design, is expected to prove more versatile than the Precomputed Gain system. The Extended filter is not limited to staying near the nominal trajectory, nor are the targeting algorithms limited to a system that can be implemented with matrix multiplication. It does not have the significant data uplink requirement imposed on the Precomputed Gain system. Extended Kalman filtering is not without its drawbacks, however. The Extended filter requires more computational power using more complex onboard software, leaving open a greater potential for flaws in the code. It is interesting to ask whether the added flexibility of this system is worth the associated complexity, particularly if the Precomputed Gain system performs well inside reasonable design boundaries.

The Extended Kalman filter is a well-known algorithm with potential for use in the MSR mission and several clear strengths against the Precomputed Gain system. For these reasons, the Extended filter has been selected as a comparison to the Precomputed Gain system, and this section will described the implementation used for the present work.

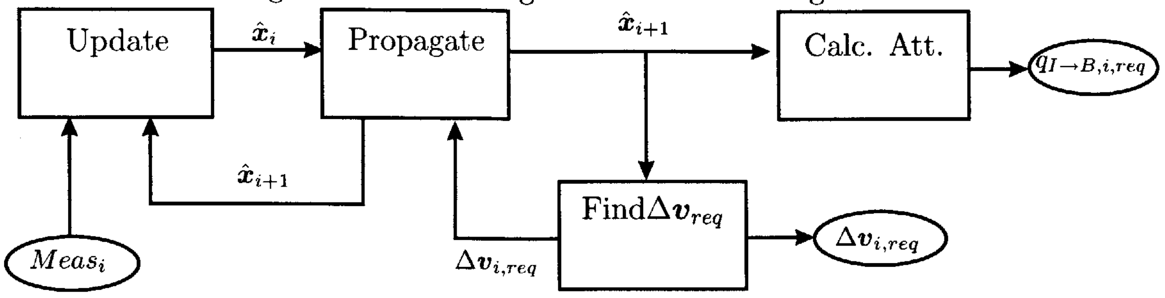
8.2.2 Configuration

At this point the Precomputed Gain Kalman filter implementation required a lengthy description of the precomputed data required for the operation of the filter. The Extended Kalman filter, however, has no such requirements. With the filter states and measurement model established in Chapter 6, and targeting algorithms described in Chaper 7, the Extended algorithm is fully configured. The next section describes its operation in the flight software.

8.2.3 Flight Software

Unlike the Precomputed Gain implementation, the Extended Kalman filter and targeting algorithm are carried onboard the spacecraft, and all computations are done in real time. This adds some complexity to the flight computer, but the flow diagram

Figure 8-3: Flow diagram for Extended algorithms



as shown in Figure 8-3 is very similar to the Precomputed Gain system. The difference is that, in each box shown in the figure, the Extended Kalman filter has to do calculations that were approximated on the ground in Precomputed Gain .

The four main computational components of the onboard algorithm as listed in the figure are:

1. Update vehicle states with measurements
2. Propagate the vehicle states
3. Find the required $\Delta \boldsymbol{v}$
4. Calculate required attitude

Not included in the list but implicit in the Kalman filter algorithm is an initialization step at time t_0. The following sections describe these components as implemented in the simulation of the Extended Kalman filter and targeting algorithms.

Covariance Initialization It is necessary to initialize the error covariance matrix at time t_0. This procedure is common between both filter implementations in this study, and was described in Section 6.3 in detail.

Update State with Measurement At every measurement time t_{meas}, the filter must update the state with the new measurement information. The measurement update equation is given in Section 4.3 as:

$$\hat{\boldsymbol{x}}_i^+ = \hat{\boldsymbol{x}}_i^- + \boldsymbol{K}_i(\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i) \tag{4.24}$$

Here $\boldsymbol{K}_i$ is the optimal Kalman gain as described in Equation 4.41, $\boldsymbol{z}_i$ is the actual measurement data, and $\hat{\boldsymbol{z}}_i$ is the estimated measurement. In Equation 4.41, $\boldsymbol{P}_i$ is the current error covariance matrix, $\boldsymbol{H}_i$ is the current measurement sensitivity matrix, and $\boldsymbol{R}_i$ is the measurement noise covariance matrix.

$$\boldsymbol{K}_i = \boldsymbol{P}_i\boldsymbol{H}_i^T(\boldsymbol{H}_i\boldsymbol{P}_i\boldsymbol{H}_i^T + \boldsymbol{R}_i)^{-1} \tag{4.41}$$

Referring heavily to previous sections, it will be quickly shown that the proper tools to calculate the terms in Equation 4.41 have already been derived and explained. The measurement estimate, $\hat{\boldsymbol{z}}$ which was essentially not performed in the Precomputed Gain system due to ITAR restrictions and processor limitations, will require more explanation below.

For the Extended Kalman filter, the measurement sensitivity matrix was shown to have the following formulation, where the measurement partials are identical to those given in Equation 6.20 for the Precomputed Gain system:

$$\left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_i}\right|_{\hat{\boldsymbol{x}}_i} = \boldsymbol{H}_i \tag{6.43}$$

Evaluating this matrix at the current state estimate yields one term required in the optimal gain calculation. The measurement noise covariance matrix, $\boldsymbol{R}_i$, was described in Section 6.5 in detail, and applies to this implementation as well as the Precomputed Gain algorithm. The Kalman gain matrix can now be calculated using Equation 4.41, leaving only the measurement estimate left to perform for a measurement update.

Recall that in the Precomputed Gain system, the need to keep the onboard navigation algorithm as simple as possible justified an exceptionally simple means of estimating the current measurement. It was assumed that the vehicle could point the camera at the target within the limit of its attitude sensors, and therefore the estimated angle measurements were always zero. The range estimate was a simple calculation using the estimated position vectors. In the Extended

Kalman filter implementation, the opportunity exists to generate a much more sophisticated (and much more accurate) measurement estimate.

The filter is estimating several states that can and should be used to improve the measurement estimate, including: camera alignment errors, measurement angle biases, and inertial attitude error.

In a generic coordinate system defined by three axes, the standard rotation matrices about a primary axis through an angle α are given in Equations 8.17 through 8.19. If α is very small, then the rotation matrices can be approximated as shown.

$$\boldsymbol{R}_1(\alpha_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_1 & \sin\alpha_1 \\ 0 & -\sin\alpha_1 & \cos\alpha_1 \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \alpha_1 \\ 0 & -\alpha_1 & 1 \end{bmatrix} \tag{8.17}$$

$$\boldsymbol{R}_2(\alpha_2) = \begin{bmatrix} \cos\alpha_2 & 0 & -\sin\alpha_2 \\ 0 & 1 & 0 \\ \sin\alpha_2 & 0 & \cos\alpha_2 \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & -\alpha_2 \\ 0 & 1 & 0 \\ \alpha_2 & 0 & 1 \end{bmatrix} \tag{8.18}$$

$$\boldsymbol{R}_3(\alpha_3) = \begin{bmatrix} \cos\alpha_3 & \sin\alpha_3 & 0 \\ -\sin\alpha_3 & \cos\alpha_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & \alpha_3 & 0 \\ -\alpha_3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.19}$$

Then for a rotation about all three axes using a vector of three small rotation angles $\boldsymbol{\alpha} = [\alpha_1 \ \alpha_2 \ \alpha_3]^T$, the full rotation matrix is found to be (remembering that the angles are small, so second order terms are negligible):

$$\boldsymbol{R}(\boldsymbol{\alpha}) = \boldsymbol{R}_1(\alpha_1)\boldsymbol{R}_2(\alpha_2)\boldsymbol{R}_3(\alpha_3) \approx \begin{bmatrix} 1 & \alpha_3 & -\alpha_2 \\ -\alpha_3 & 1 & \alpha_1 \\ \alpha_2 & -\alpha_1 & 1 \end{bmatrix} \tag{8.20}$$

Now recall that the static camera alignment error and the inertial attitude error states are vectors of three small angles representing rotations from the nominal state. In the case of the body-fixed camera alignment error, $\boldsymbol{\epsilon}$, the transformation represents small differences between the orbiter body frame and the camera frame. In the case of the inertial attitude error, $\boldsymbol{\Theta}$, the transformation represents small knowledge errors between the inertial and body frame (in addition to the standard inertial to body transformation). Given this description and the knowledge that the filter is *estimating* non-zero values for these angular misalignments, the relative position vector can be formulated as follows:

$$\boldsymbol{r}^c_{rel} = \boldsymbol{T}_{cam\_nom \rightarrow c} \boldsymbol{T}_{b \rightarrow cam\_nom} \boldsymbol{T}_{I' \rightarrow b} \boldsymbol{T}_{I \rightarrow I'} (\boldsymbol{r}^I_{OS} - \boldsymbol{r}^I_{Orb}) \tag{8.21}$$

$$\boldsymbol{r}^c_{rel} = \boldsymbol{R}(\boldsymbol{\epsilon}) \boldsymbol{T}_{b \rightarrow cam\_nom} \boldsymbol{T}_{I' \rightarrow b} \boldsymbol{R}(\boldsymbol{\Theta}) (\boldsymbol{r}^I_{OS} - \boldsymbol{r}^I_{Orb}) \tag{8.22}$$

$$\hat{\boldsymbol{r}}^c_{rel} = \boldsymbol{R}(\hat{\boldsymbol{\epsilon}}) \boldsymbol{T}_{b \rightarrow cam\_nom} \boldsymbol{T}_{I' \rightarrow b} \boldsymbol{R}(\hat{\boldsymbol{\Theta}}) (\hat{\boldsymbol{r}}^I_{OS} - \hat{\boldsymbol{r}}^I_{Orb}) \tag{8.23}$$

This is the best calculation of the relative position vector in the camera frame using all available data from the estimated state. Recall that the measurements for the optical camera depend solely on the relative position vector and the orientation of the camera, and are calculated according to Equations 5.18 through 5.21.

There is still one more piece of information from the filter mentioned above that can be used to improve the estimate still further. The filter is estimating the parameters b_e and b_α, the camera measurement angle biases. These biases can be subtracted out to yield the final best measurement estimate:

$$\hat{\boldsymbol{z}}_i = \begin{bmatrix} \hat{e}_i - \hat{b}_e \\ \hat{\alpha}_i - \hat{b}_\alpha \\ \hat{r} \end{bmatrix} \tag{8.24}$$

At last, every term in Equation 4.24 is known, and the state can be updated with the measurement data at time t_i.

Propagate States and Covariance At every time step it is necessary to propagate the state vector and covariance matrix to the next time. The first thing to calculate is an update to the covariance matrix if the current time is a maneuver time, $t_i = t_{man}$. The details of this calculation have been presented in Section 6.6.1 and are applied here.

The next task is to propagate the state vector. Since the Extended Kalman filter implementation assumes an onboard processor capable of reasonable levels of computing, the target and chaser position and velocities are simply integrated using the orbital equations of motion and a fourth order Runge-Kutta numerical integration technique. The remaining 8 states, modeled as first order Markov processes as before, are propagated using the technique developed in Section 6.1.2. Namely:

$$\boldsymbol{\Phi}_{i+1,i} = \begin{bmatrix} \boldsymbol{\Phi}_{b_c,b_\alpha}\ (2\times2) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi}_\epsilon\ (3\times3) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Phi}_\Theta\ (3\times3) \end{bmatrix}_{i+1,i} \tag{8.25}$$

Here the individual state transition matrices are calculated according to Equation 6.15:

$$\boldsymbol{\Phi}_{y,i+1,i} = e^{-\Delta t/\tau}\boldsymbol{I} \tag{6.15}$$

In order to propagate the covariance matrix to the next time step, the full state transition matrix and process noise covariance matrix are required. Recall Equation 4.23:

$$\boldsymbol{P}_{i+1} = \boldsymbol{\Phi}_{i+1,i}\,\boldsymbol{P}_i\,\boldsymbol{\Phi}^T_{i+1,i} + \boldsymbol{Q}_i \tag{4.23}$$

The state transition matrix and process noise covariance matrix are calculated as described in Section 6.4, allowing propagation of the error covariance matrix according to the equation above.

Find $\Delta\boldsymbol{v}$ Required Referring to the flow diagram in Figure 8-3, the flight computer's next step is to use the estimate of the filter state at time t_{i+1} to calculate the required change in velocity at that time step, $\Delta\boldsymbol{v}_{i+1}$. The targeting algorithms for the Extended implementation have been described in Chapter 7. This calculation, then, simply requires executing the targeting algorithm (including triggers) on the current time and state estimate.[2]

Calculate Required Attitude The "required attitude" for the chaser vehicle is the attitude that provides boresight pointing towards the chaser vehicle, based on the flight computer's estimate of the both vehicle states. This calculation is identical to the Precomputed Gain calculation of the same quantity, with no revisions. A description of the calculation can be found in Section 8.1.3.

[2]Note that the linearized position and velocity corrections described in Chapter 7 are not used by the Extended system. The required targeting algorithms are: Lambert targeting, football orbit control, coelliptic insertion, fast transfer insertion, and the glide slope.

Chapter 9

Performance Envelope

The primary quantitative result from this thesis is the performance analysis of the two sets of algorithms studied. In preceding chapters, the details of the environment models and flight algorithms have been described in detail. In this chapter, the performance analysis tools are presented, followed by the actual operating parameters for the study and the results of the analysis.

9.1 Performance Analysis Tools

The analysis tool used in this thesis is a pseudo 6-DOF Monte-Carlo simulator, described fully by the preceding chapters. The Monte-Carlo simulator uses parameters supplied by the operator to repeatedly simulate the actual rendezvous experiment. An important question when performing the Monte-Carlo analysis was: how many trials are required for a statistically significant result? One would like to run many hundreds, perhaps thousands of trials to ensure that the statistics are fully covered in the analysis.

Unfortunately, running 1000 cases per trial or more is not computationally practical for this analysis. One of the known drawbacks to Monte-Carlo analyses is the large computational burden, and this rendezvous simulator is no exception – a case requiring 1000 trials is prohibitive in many computer resources, including memory, storage, and time. It is the experience of the author, after a great deal of time spent evaluating results from this tool, that cases involving more than about 200 trials have reached a point of diminishing returns, and very little is added to the result by continuing further. For this reason, every Monte-Carlo case discussed in this section will represent 200 trials, and it shall be up to the reader to evaluate whether or not this provides a sufficient level of confidence for his or her purposes.

Given the large volume of data generated by the Monte-Carlo simulator, it is necessary to reduce it into a format that is useful and clear. Chapter 2 briefly mentioned several tools that will be used to show the results of the analysis. These include: mean and variance of the required $\Delta \boldsymbol{v}$; magnitude of position and velocity navigation error, particularly at specific epoch times such as closest approach; position dispersions at epoch times such as maneuver times and closest approach; and, target pointing error at epoch times, such as when the target leaves eclipse. In addition, passive abort outlook will be discussed, along with the basic 'trend' of the results shown in terms of hair plots. Each of these analysis tools will be discussed briefly here, such that the figures in the sections to follow will be readily understandable.

Required $\Delta \boldsymbol{v}$ One of the most observable metrics from the Monte-Carlo analysis is the amount of $\Delta \boldsymbol{v}$ required for a given scenario. Intuitively, the amount of $\Delta \boldsymbol{v}$ required relates directly to the amount of fuel required for the rendezvous mission, which is of keen interest to mission designers. It is a simple matter while performing the simulations to record every applied maneuver for every trial, and perform any number of analyses on the data in post-process.

In this thesis, where the relevant comparison is between two separate sets of flight algorithms, the presentation judged appropriate is the mean and standard deviation of the total required Δv for a given case.[1] That is, for a particular set of input parameters, the Monte-Carlo tool runs 200 simulated rendezvous experiments (trials), and stores every required maneuver for each trial. Then it is possible for every trial to calculate the total required Δv by summing the magnitudes of each required maneuver. The result is a list of total required

[1]Note that this is a scalar change in velocity, $\Delta v = |\Delta \boldsymbol{v}|$.

change in velocity for 200 trials. The mean and standard deviation of this list can be easily computed, and presentations of this data can be found in Sections 9.3.1 and 9.3.2.

Navigation Error For every Monte-Carlo trial, there exists both an estimated state vector, $\hat{\boldsymbol{x}}$, and a "true" state vector, $\boldsymbol{x}$, where the estimate is kept by the flight computer and the truth is the simulated reality of the environment model. The *navigation error* can then be calculated according to Equation 4.9 as follows:

$$\tilde{\boldsymbol{x}}_i = \hat{\boldsymbol{x}}_i - \boldsymbol{x}_i \tag{4.9}$$

That is, in the discrete time of the simulation, the navigation error is the difference between the flight computer's estimate of the state and the true state. Clearly this is computationally intensive, but a useful visualization can result from processing of this sort.

For every Monte-Carlo trial, the truth state can be subtracted from the estimated state at every time step. The result is the navigation error for that particular trial at every time. Remembering that the power of the Monte-Carlo method is in the large number of deterministic trials, this calculation can be performed for all 200 trials simulated for a given parameter set. The mean of this error is expected to be zero (and the statistical analysis can provide a good check for this), but the standard deviation can be easily calculated and plotted as a function of time. It is found most useful to plot the navigation error for relative position and velocity (in LVLH coordinates),[2] and figures showing this information can be found in Sections 9.3.1 and 9.3.2.

It is interesting to note that, for the Precomputed Gain system, most events happen at the same time by design. Maneuvers are triggered at certain times, and closest approach is, to within a small error, passed at the same time in

[2]There are two reasons for this view. First, the Precomputed Gain system is only estimating position and velocity for both vehicles, so it does not make sense to try to compare static alignment, inertial attitude, etc. Second, it is much easier to conceptualize the navigation error in a coordinate system defined by 'downrange,' 'crosstrack,' and 'altitude,' than it is in an inertial coordinate system.

every trial. In the Extended system, on the other hand, maneuvers are performed when commanded by geometric triggers, so they happen over a small range of times. Likewise, some trajectories finish sooner or later than the 'nominal' trajectory and so the time of closest approach may also vary over some small range. Thus after subtracting the truth state from the flight state in the Precomputed Gain system, a reasonable pairing is made when taking statistics among the trials. In the Extended system, however, it is entirely possible that a time-synchronized comparison would be comparing a point on one trajectory *before* a maneuver to a point on another trajectory *after* that same maneuver. This difficulty and strategies to avoid significant error will be presented in their place.

Position Dispersions For a given parameter set, each trial of a Monte-Carlo run has a slightly different trajectory based on accumulating random errors and the performance of the targeting algorithms. It is important to evaluate the overall trend for the parameter set under study, particularly at certain epoch times such as maneuvers and closest approach to the target. As mentioned in the Navigation Error section, the true state vector is stored at every time step (and thus the true relative position vector can be calculated), so it is a simple matter to accumulate the relative position of the orbiter at the epoch times of interest.[3] The author has found that one of the most useful ways to present this data is a simple scatter plot, where the relative position is plotted as a point for every trial, at every epoch time. In this way the dispersion ellipses can be visualized at desired times. Figures showing this information can be found in Sections 9.3.1 and 9.3.2.

Target Pointing Error In this pseudo 6-DOF simulator, the orbiter is continuously attempting to point the camera boresight at the target spacecraft, based on the flight computer's estimate of the state vector (see Sections 8.1.3 and 8.2.3). Ideally, the camera would be pointed directly at the target for every measurement,

[3]Note again that, as was discussed for the Navigation Error, epoch times such as maneuvers and closest approach must be used, rather than absolute times.

and every azimuth and elevation measurement would be zero. Of course, this is never true in a real rendezvous scenario, and there is always some measure of *target pointing error* for each measurement. As shown in Section 5.2.4, it is important that the angle from the camera boresight to the target not exceed the FOV of the camera, or no measurement update can be performed.

In the Monte-Carlo simulation, the target pointing error can be stored for every measurement of every trial, and this data post-processed as desired. One particularly interesting time to look at target pointing error is when the orbiter takes its first measurement after exiting a constraint (such as a target shadowing or Sun angle constraint). In these situations, no measurements have been taken for extended periods of time, and the flight computer has been forced to propagate the state vector with no update. Navigation error grows due to initial knowledge errors and accumulating unmodeled accelerations. When the orbiter finally exits the constraint, it points the camera boresight at the target according to the state estimate and takes a measurement. It is this time when the target pointing errors are highest, and the target is often not in the camera's field of view. Thus, a useful way to analyze the stored target pointing error data is to collect the measured angle for every trial at each "constraint exit." This data can be viewed in a histogram, or plotted in terms of mean and standard deviation of the target pointing angle at these epoch times. Figures showing this information can be found in Sections 9.3.1 and 9.3.2.

9.2 Configuration

The Kalman filter algorithms and Monte-Carlo environment model are very flexible and contain a large number of parameters that can be adjusted by the user. This section aims to present all of the actual parameters used in this study in a clear and orderly way. An attempt will be made to refer the reader to the appropriate section of this document to review the use of these variables as they are described. Note that the Precomputed Gain and Extended Kalman filter implementations have been unified such that all of these parameters are required by both sets of algorithms, so no distinctions are made.

Section 9.2.1 contains values (used in either the environment model or filtering algorithms) that are to be considered fixed for the present study. These values can certainly be changed to evaluate a different scenario but were used as shown to generate the results that follow.

Section 9.2.2 deals with the nominal values for parameters that have been adjusted for this study. These are the approximate values that the designers expect to see in the actual mission. The *nominal case* uses these values in both the flight computer and environment models.

Section 9.2.3 deals with the off-nominal values for parameters that have been adjusted for this study. One alternate parameter set for unmodeled acceleration is shown, along with two alternate execution error models. When these off-nominal parameter sets are used in both the environment model and the flight algorithms, the results are the *off-nominal cases* considered in this analysis.

Section 9.2.4 describes combinations of off-nominal parameter sets used in the environment model with nominal parameters used in the flight computer. The result is a set of *stress cases* designed to test the limits of the flight algorithms. This section also contains a table that summarizes all of the cases run in this analysis (nominal, off-nominal, and stress).

9.2.1 Fixed Parameters

As mentioned in Section 2.1, the problem at hand has many angles that could be studied in a variety of ways. Time and space considerations forced the scoping of this problem down to two main tunable parameters: unmodeled accelerations and execution error. All other parameters, including the spacecraft orbit and reference rendezvous trajectory have been fixed. These "fixed" parameters are presented here, with a note that the baseline trajectory and spacecraft orbital elements have already been described in Chapter 3.

The epoch time used in this analysis was presented in Table 3.2 of Section 3.3.1

as April 1, 2011, just after midnight on March 31. In addition to the epoch time, a surprisingly small number of physical values must be specified. The central body in this analysis is Mars, and the values for the gravitational parameter, μ_M, and the equatorial radius, r_{eq}, are provided in Table 9.1. Recall that a spherical planet is assumed in this analysis, so the J2 gravity term is taken to be zero. The J2 model could be considered in future work, however for relative motion and the short time scales considered in this work, the oblateness term is expected to have a negligible effect.

Parameter	Value
μ_M	$4.2828385943 \times 10^{13}$ (m^3/s^2)
r_{eq}	3397200 (m)

Table 9.1: Physical constants used in simulation

Several physical parameters are required relating to the interaction of the onboard cameras with the environment. Recall that the MTO spacecraft will be equipped with two cameras, a wide-angle camera (WAC) and a narrow-angle camera (NAC). These cameras never operate simultaneously in the simulation, and have very different characteristics. Section 5.2.4 described the Sun angle and FOV constraints, both of which depend on properties of the camera. Table 9.2 lists the minimum allowable Sun angle for both cameras (no measurements are allowed if the angle between the target and the Sun is less than this value), as well as the full FOV angle for both cameras. The minimum Sun angle has been calculated as approximately 30° plus half

Constant	Value
NAC Min Sun Angle	30°
WAC Min Sun Angle	52.5°
NAC full FOV	1.4°
WAC full FOV	45°
Illuminator Range	0 m
WAC Range	20 m
Target Diameter	0.20 m

Table 9.2: Sensor related parameters used in simulation

Range (m)	Δ Measurement Time (s)
≥ 250	60
< 250	30

Table 9.3: Measurement rates used in simulation

of the camera FOV. The table also indicates the assumed diameter of the spherical target, the range at which the orbiter switches from the NAC to the WAC, and the range at which the target illuminator comes on. Note that, with a value of zero, the illuminator is never used in this analysis. The capability was included for future study, but in this work a worst-case scenario was assumed where capture cannot occur while the target is shadowed.

Another important property governing the simulation operation is the measurement rate. Table 9.3 indicates the time between measurements given the range from the chaser to the target.

In the calculation of the measurement noise covariance matrix shown in Section 6.5, the variance of the angle error is required for elevation and azimuth. For this study these two angles are calculated in the same way from the LOS vector, so their standard deviations are the same as shown in Table 9.4. The standard deviations

	NAC (rad)	WAC (rad)
σ_e	0.024×10^{-3}	0.77×10^{-3}
σ_α	0.024×10^{-3}	0.77×10^{-3}

Table 9.4: Measurement angle error standard deviation

are calculated by assuming a one pixel uncertainty in the image processing and a 1024 × 1024 pixel detector that is baselined for the mission. Thus, for example, the angular uncertainty for the NAC is given by Equation 9.1:

$$\sigma_e = \sigma_\alpha = \frac{1.4^\circ}{1024} = 0.0014^\circ/\text{pixel} = 0.024 \text{ mrad/pixel} \tag{9.1}$$

The same calculation is repeated for the WAC, remembering the larger FOV from Table 9.2.

Finally, the last set of "fixed" parameters is the expected standard deviations for the elements of the state vector (those states modeled by Markov processes also require a time constant to be specified, as shown in Section 6.1.2). The standard deviations of the Markov states are shown in Table 9.5. Note that the measurement biases, b_e and b_α, have different values between the two cameras, while the values for the components of the static alignment error, $\boldsymbol{\epsilon}$, and the inertial attitude error, $\boldsymbol{\Theta}$, do not. The value of the angle bias standard deviation is the same as that for the angle measurement error, and for the same reason. The inertial attitude error has a time constant of 100 sec, which is similar to the expected time between star tracker updates to the inertial attitude. The other states are considered biases, and so have time constants that are very large relative to the simulation integration time (10^{16} sec in simulation).

State	Std Deviation (rad)
$b_{e,NAC}$	0.024×10^{-3}
$b_{e,WAC}$	0.77×10^{-3}
$b_{\alpha,NAC}$	0.024×10^{-3}
$b_{\alpha,WAC}$	0.77×10^{-3}
$\epsilon_1, \epsilon_2, \epsilon_3$	$0.01/3 \times \frac{\pi}{180}$
$\Theta_1, \Theta_2, \Theta_3$	$0.1/3 \times \frac{\pi}{180}$

Table 9.5: Markov states: Initial standard deviations and time constants

The initial standard deviations for the position and velocity states are given in LVLH coordinates, as described in Section 6.3. The standard deviations for the target are an order of magnitude larger than those for the chaser, indicating NASA's ability to track the orbiter using the Deep Space Network (DSN), but not the OS. Section 6.3 also describes the known correlations between the vehicle position and velocity in LVLH coordinates, used in the initialization of the covariance matrix.

	OS (m)	Orb (m)
$3 \cdot \sigma_x$	100	10
$3 \cdot \sigma_y$	30	3
$3 \cdot \sigma_z$	30	3

	OS (m/s)	Orb (m/s)
$3 \cdot \sigma_{\dot{x}}$	0.03	0.003
$3 \cdot \sigma_{\dot{y}}$	0.03	0.003
$3 \cdot \sigma_{\dot{z}}$	0.03	0.003

Table 9.6: Position and velocity states: Initial LVLH standard deviation

9.2.2 Nominal Case

As was mentioned above, the two primary 'design knobs' for this analysis are the unmodeled accelerations and the execution error model. This section presents the nominal values for these parameters, representing values actually expected for the MTO mission. When the nominal parameters are used in both the environment model and the flight algorithms, the result is the 'nominal case', indicated as case 3 in the summary table below.

The nominal values of unmodeled acceleration, shown in Table 9.7, represent a case where the effects of these small accelerations would result in approximately 100 m of downrange error in 24 hours, in the absence of measurement updates.

	Nominal (m^2/s^3)
$k_{u,OS}$	3×10^{-13}
$k_{u,Orb}$	3×10^{-13}

Table 9.7: Nominal unmodeled acceleration

The nominal execution error model is shown in Table 9.8, fully described in both the environment and filter algorithms by four parameters. The Kalman filter execution error model given in Sections 6.6.1 and 6.7.2 specifically describes the use of the scale factor, pointing factor, and quantization standard deviations (σ_{sf}, σ_{pf}, and σ_q, respectively). The environment execution error model is described by Table 5.1 in Section 5.2.2. Note that the generic execution error model that the environment is capable of using allows for different error models for large and small maneuvers. For this study, this capability has not been used. In Table 5.1, $\sigma_R = \sigma_{sf}$, $\sigma_T = \sigma_{pf}$, and $\sigma_S = \sigma_q$, applied to maneuvers of any size. The fourth parameter in Table 5.1, labeled *SMC*, represents the value used by the targeting algorithms for the small maneuver

	Nominal
$3 \cdot \sigma_{sf}$	0.01 (m/s)
$3 \cdot \sigma_{pf}$	0.1 deg
$3 \cdot \sigma_{q}$	0.003 (m/s)
SMC	0.006 (m/s)

Table 9.8: Nominal execution error model

constraint, as described in the introduction to Chapter 7.

9.2.3 Off-Nominal Cases

When the off-nominal parameter sets are used in both the environment and the flight computer algorithms, the result is a group of 'off-nominal cases.' Table 9.9 shows two alternate sets of values for unmodeled accelerations, and Table 9.10 shows two alternate parameter sets for the execution error model. The off-nominal values of unmodeled acceleration shown in the table would result in approximately 1000 m of downrange error in 24 hours for the "high" parameter set, and approximately 10 m of downrange error in 24 hours for the "low" parameter set, in the absence of measurement updates.

	Low (m^2/s^3)	**Nom** (m^2/s^3)	**High** (m^2/s^3)
$k_{u,OS}$	3×10^{-15}	3×10^{-13}	3×10^{-11}
$k_{u,Orb}$	3×10^{-15}	3×10^{-13}	3×10^{-11}

Table 9.9: Nominal and off-nominal unmodeled accelerations

	Low	**Nom**	**High**
$3 \cdot \sigma_{sf}$	0.01 (m/s)	0.01 (m/s)	0.05 (m/s)
$3 \cdot \sigma_{pf}$	0.1 deg	0.1 deg	0.1 deg
$3 \cdot \sigma_{q}$	0.001 (m/s)	0.003 (m/s)	0.003 (m/s)
SMC	0.002 (m/s)	0.006 (m/s)	0.006 (m/s)

Table 9.10: Nominal and off-nominal execution error models

	Stress 1 (m^2/s^3)	**Stress 2** (m^2/s^3)
$k_{u,OS}$	1.2×10^{-10}	4.8×10^{-10}
$k_{u,Orb}$	1.2×10^{-10}	4.8×10^{-10}

Table 9.11: Stress case unmodeled accelerations

	EE	UM	**Filter**						
EE			Low		Nom			High	
UM			Nom	High	Low	Nom	High	Nom	High
Env.	Low	Nom	1						
		High		2					
	Nom	Low							
		Nom			9	3			
		High				12	4		
	High	Nom	7		10	13		5	
		High	8		11	14			6
		Stress 1				15			
		Stress 2				16			

Table 9.12: Summary of parameters for Monte-Carlo cases

9.2.4 Stress Cases

When the sources of error in the environment model are larger than the values expected by the Kalman filter, the flight algorithms become stressed. One of the objectives of this study is to evaluate how the Precomputed Gain and Extended Kalman filters perform under this kind of stress, so several cases have been designed on this approach. Table 9.11 presents two more levels of unmodeled accelerations that were used to stress the Kalman filters. The levels of unmodeled accelerations in parameter sets "Stress 1" and "Stress 2" correspond to 2000 m/day and 4000 m/day of downrange navigation error, respectively.

Table 9.12 presents a summary of every case evaluated in this work, including the nominal and off-nominal cases mentioned above. The abbreviations "EE" and "UM" stand for execution error and unmodeled accelerations, respectively. The stress cases are indicated as case 7 through case 16.

9.3 Results

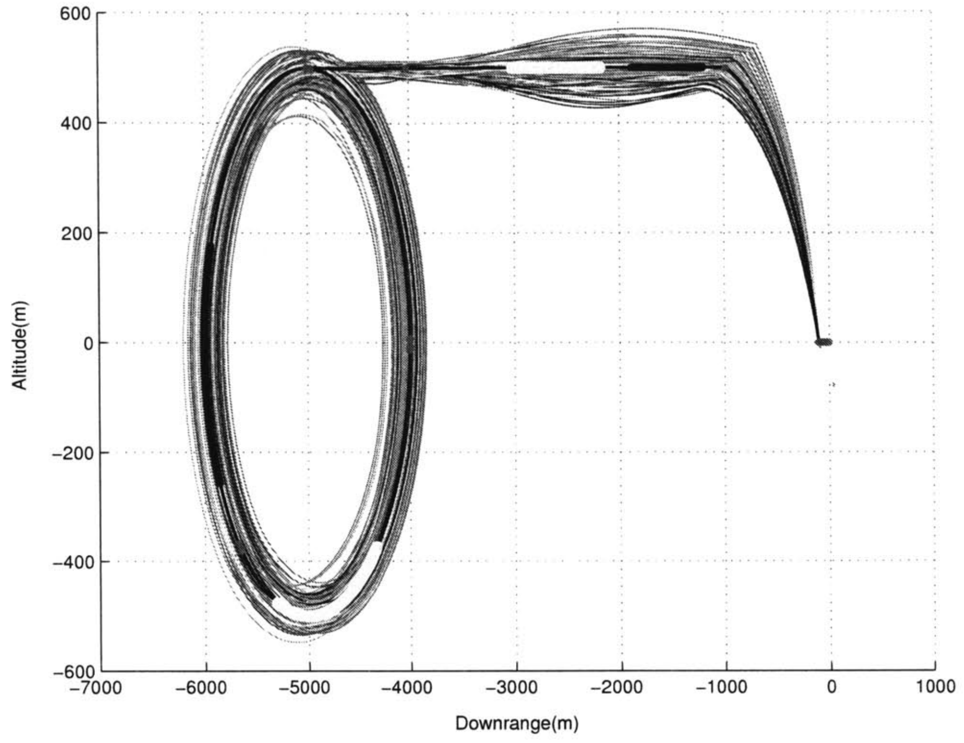
The Monte-Carlo cases listed in Table 9.12 were run using both the Precomputed Gain and Extended algorithms. A large array of computational resources were pooled to complete this massive analysis in a timely fashion. From the data that resulted, the **performance envelope** was established for both algorithms, for the mission and trajectory under study. Additionally, a great deal was learned about the performance of both algorithms *inside* their respective envelopes. Section 9.3.1 addresses the results from the analysis of the Precomputed Gain system, while Section 9.3.2 addresses the results from the analysis of the Extended system. Finally, Section 9.3.3 provides a comparison between the two algorithms.

9.3.1 Precomputed Gain

There are three levels of analysis that are of interest in this study. First, for every case that is run, it is interesting to analyze the performance of the navigation and targeting algorithms through detailed looks into the target pointing error, $\Delta \boldsymbol{v}$ required *for each maneuver*, position dispersions, and navigation error. Second, it is interesting to investigate, for every case run using a particular algorithm, how the different cases compare to each other in terms of mean pointing error, total Δv required, navigation error, and position dispersions. Finally, it is interesting to compare the performance of the two different implementations under study when configured in the same way (same 'cases').

Unfortunately, their are too many cases to present a detailed performance analysis of each, even in a lengthy work such as this. As a compromise, an in-depth analysis will be shown for the nominal case, so the details of the algorithm performance will be recorded for the actual expected mission configuration, and the reader can get a sense of what types of analysis can be performed with this tool. What then follows is the second point of interest above: a less detailed but equally valuable comparison between the different cases run on the Precomputed Gain implementation. The third point of interest, a comparison between the Precomputed Gain and Extended

Figure 9-1: PG Nominal case: downrange vs. altitude hair plot

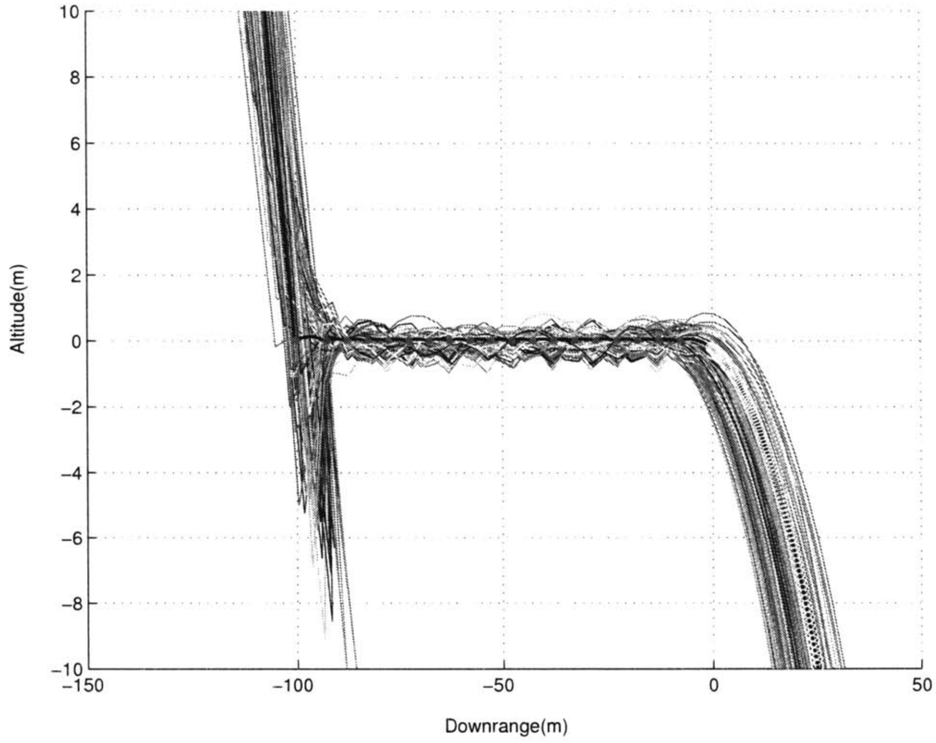


systems, is presented in Section 9.3.3.

Nominal Case

The **nominal case** is defined by the parameter sets shown in Section 9.2.2. Both the flight computer and the environment model are configured to those parameters, resulting in a "properly tuned" filter using expected mission values. Figures 9-1 through 9-3 show several views of a *hair plot* for a 200 trial Monte-Carlo run on the nominal case. The hair plot is generated by plotting all 200 trials on the same figure, and has proved to be an excellent way to view the data and make qualitative assessments of algorithm performance. Figure 9-1 shows a full view of the trajectory in the most natural 2-dimensional coordinate system. The LVLH frame is used, as described in Section 3.1, and the chosen axes are downrange distance and altitude. Note that, in this frame, the target is at the origin of the plot, and the chaser vehicle

Figure 9-2: PG Nominal case: downrange vs. altitude glide slope hair plot



is tracing out the trajectories shown.

A great deal of additional information is shown on the plot. The reference trajectory is indicated by a heavy line in the midst of the Monte-Carlo trials, as a reminder that the Precomputed Gain system always endeavors to remain close to the nominal. If the reader finds the reference trajectory difficult to see, recall that the baseline rendezvous trajectory was presented by itself in Figure 3-7. The target shadowing and Sun angle constraints are indicated on the reference trajectory by series of heavy dots and light circles, respectively. Recall that no measurements are taken during these constraints. Maneuvers are also indicated on the reference trajectory as single dots, but are more readily identified on Figure 3-7.

Figure 9-2 is an enlargement of the glide slope portion of the trajectory, showing the point of closest approach and the series of rapid maneuvers that allows the chaser to approach the target along the v-bar (see Section 3.2.4). The observant reader might notice that the reference trajectory is much "smoother" than the individual

Monte-Carlo trajectories. This is simply an artifact of data sampling, as the large number of trials makes presentation of the full data set prohibitive.

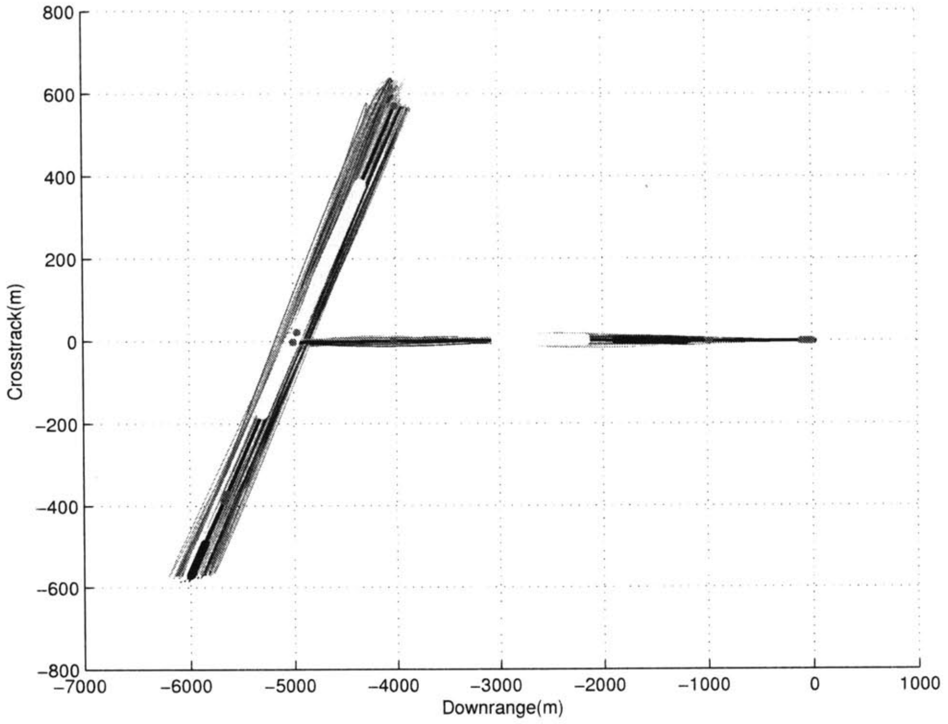
It seems useful at this point to remind the reader that the mission under consideration is a rendezvous *demonstration*, not an actual rendezvous mission. The chaser vehicle will not be equipped with capture hardware. It is for this reason that the target point of the final maneuver is not the origin, where the orbiting sample is, but a point 3 meters downrange. The purpose of the rendezvous demonstration is to validate the hardware and software that will be used in the MSR mission, and this validation will be achieved by performing proximity operations, rather than an actual rendezvous. Here, then, is a good example of the type of information that gain be gained from the hair plot presentation of the Monte-Carlo data. It is clear from Figure 9-2 that the 3 meter target point, for this characterization of the environment and configuration of the flight computer, would not be suitable for the rendezvous demonstration. Many of the 200 trajectories, rather than passing a safe 3 meters from the target, would put the chaser on a collision course with the target. In the actual mission, the closest downrange approach will probably be significantly further out than 3 meters.[4] It is also worth commenting that the scale of the plot in Figure 9-2 is not square, so the glide slope is actually a good deal "flatter" than it looks in the figure.

Figure 9-3 can be thought of as a "top-down" view of the Monte-Carlo rendezvous trajectories, showing downrange distance versus crosstrack distance. In this view, the out-of-plane component of the initial holding football orbit can be seen clearly, as well as the fact that the rest of the trajectory has very little crosstrack motion at all.

Of interest in all of the hair plots shown is the fact that all of the Monte-Carlo trajectories remain relatively close to the nominal trajectory. This is true by design, and largely due to the nature of the linearized Lambert targeting algorithm used in

[4]Note also that the chaser altitude is quite small when the downrange component goes to zero after the last maneuver, even on the nominal trajectory. This would prove to be too close for comfort in the actual mission, but the 3 meter value was selected to verify the proximity operations in simulation. The reader can easily imagine an earlier termination of the demonstration simply by cutting off the glide slope after any maneuver, equivalent to shifting the approach down the v-bar away from the target.

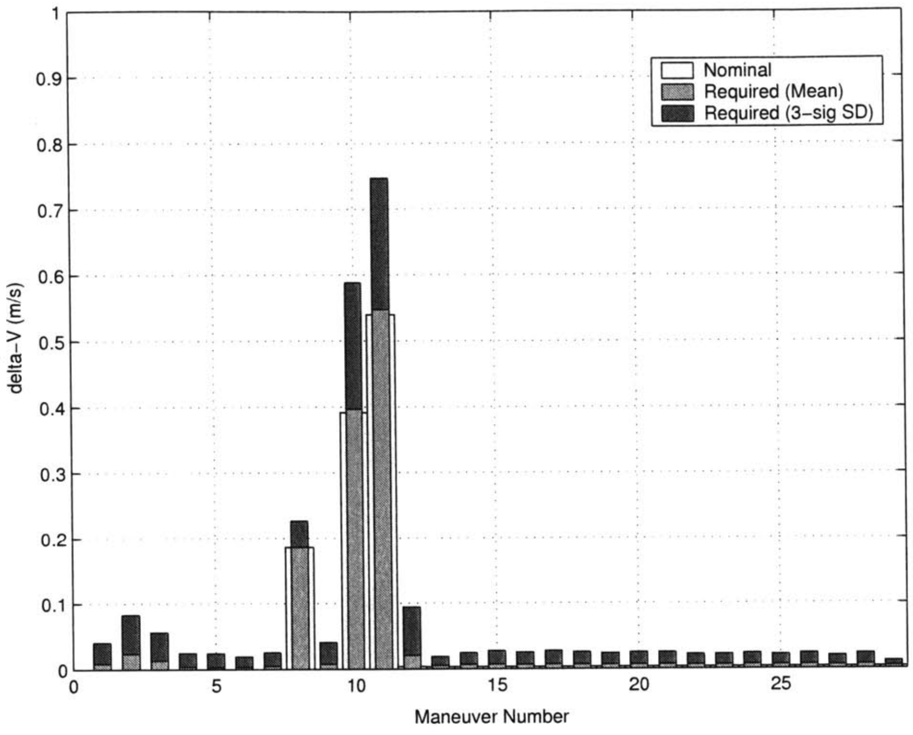
Figure 9-3: PG Nominal case: downrange vs. crosstrack hair plot



the Precomputed Gain system (see Section 7.2.1), which is configured to target back to the nominal trajectory in a specified period of time. It is necessary to prevent large position dispersions from the nominal trajectory, which can cause the linearizations used in the calculation of the precomputed data to become invalid, and the Precomputed Gain system to break down.

When talking about a single Monte-Carlo case of 200 trials, it is possible (and useful) to plot the statistics of the "applied" maneuvers at each maneuver time. That is, for every Monte-Carlo trajectory, every maneuver that the environment model actually applies to the orbiter state can be stored (this, then, would include the maneuver execution error and would be proportional to the actual amount of fuel used in the mission). Then, the magnitude of each maneuver can be calculated, and from this the mean and standard deviation for each maneuver (compare this to the *total* Δv magnitude strategy described above for comparing Δv between cases). Figure 9-4 shows the required Δv statistics for the nominal case. The widest bars on

Figure 9-4: PG Nominal case: applied Δv statistics



the plot represent the nominal maneuver, or the maneuver that is commanded at that time, to create the reference trajectory. The narrow bars represent the 3-σ standard deviation from the mean, stacked on top of the mean required change in velocity. The reader is again referred to the baseline rendezvous trajectory shown in Figure 3-7 to associate the maneuvers with their locations on the trajectory. Maneuvers 1 through 7 are purely corrective maneuvers (no nominal part) on the football orbit. Maneuver 8 initiates the coelliptic transfer, and maneuver 9 is a purely corrective maneuver on the coelliptic. Maneuver 10 initiates the fast transfer, maneuver 11 closes it and creates the small 'bounce,' and the remainder of the maneuvers establish the glide slope. The bars in the figure correlate very well to this description. The first 7 maneuvers have no nominal piece, but a statistical part indicating that a certain amount of $\boldsymbol{\Delta v}$ *is* required to maintain the football orbit near the nominal. Notice that the required $\boldsymbol{\Delta v}$ is higher in the first few corrective burns than in the last few, as the flight computer removes the dispersions in the initial state. The maneuvers to start and end the fast transfer are the largest by far, indicative of the desire to cover a relative large distance in a relatively short period. The glide slope maneuvers are all nominally very small, but require corrective parts that are larger, to keep the orbiter on course while fighting the maneuver execution errors and unmodeled accelerations. Every maneuver shown, even if it does not have a nominal part, *does* have a corrective part.

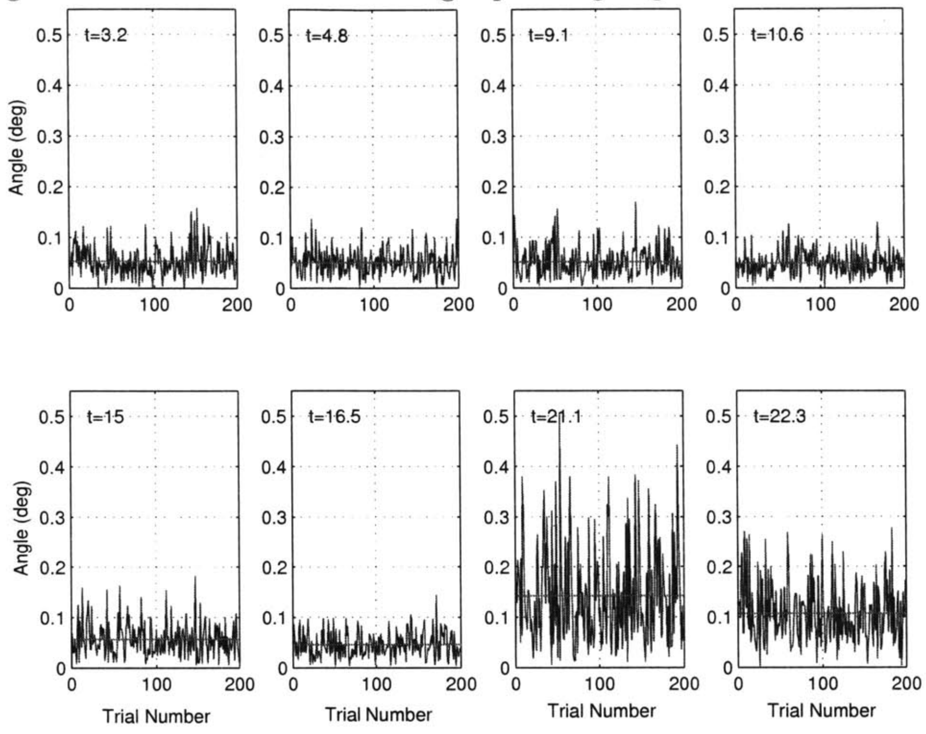
Note that, within a small error, the mean maneuver magnitude is the same as the nominal maneuver magnitude. This indicates that the corrective parts of the maneuvers, generated using the $\boldsymbol{C}^*$ matrices and the perturbation from the state, have a mean of zero. That is, the nominal maneuver is always performed as precomputed, resulting in a mean requirement equivalent to the nominal, and the corrective parts are zero mean and so only add to the standard deviation.

The 3-σ standard deviation added to the mean creates a maximum maneuver size, below which one could expect 99.7% of all maneuvers to fall, at that maneuver time. By summing the mean and 3-σ standard deviation over all of the maneuvers, one can find the maximum total required Δv for the nominal case, again with a 99.7% certainty. For the nominal case, that maximum magnitude is 2.36 m/s.[5]

Section 9.1 describes the *target pointing error* as one tool that can be used to evaluate the performance of these algorithms for a given set of input parameters. When evaluating multiple cases, the data must be drastically reduced to be presented in a useable form; for the single case of the nominal, however, the target pointing error at each constraint exit can be presented at every constraint exit in a useful way. Figure 9-5 shows a line plot indicating the measured target pointing error for every Monte-Carlo trajectory, at every constraint exit. There are a total of 8 periods where the orbiter is not able to take measurements: 3 target shadowing constraints and 3 Sun angle constraints on the holding football orbits, and one each on the coelliptic. The trajectory was specifically designed to 'finish' the rendezvous before

[5]Note that this is a sum of the 3-σ worst-case magnitude for every maneuver. In any given trajectory, it is highly unlikely that the 3-σ value would be reached each time. In fact, it is likely that some maneuvers would be smaller than the nominal, while others are larger. Thus, if one were to take the mean and standard deviation of total Δv required for each rendezvous, it is expected that this value would be smaller than the one quoted above (see Figure 9-11).

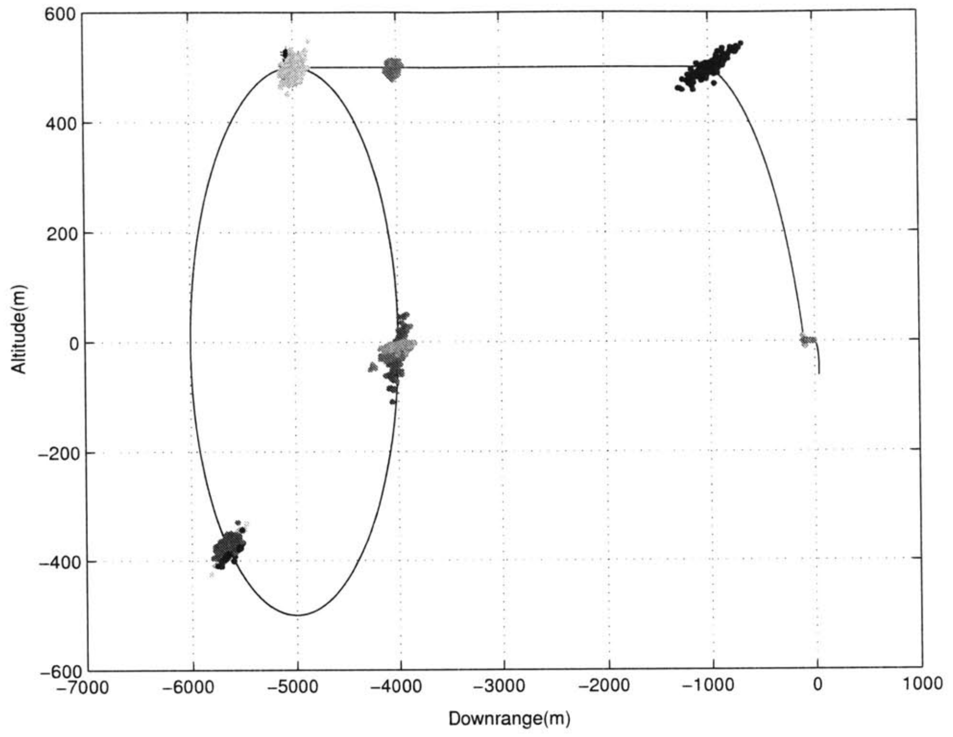
Figure 9-5: PG Nominal case: target pointing angle at constraint exits



a new constraint is entered on the glide slope. When exiting all 8 constraints on the rendezvous trajectory, the orbiter is using the narrow angle camera (NAC). Thus, the maximum allowable measured angle from the boresight is taken as half the FOV angle shown in Table 9.2. It can be seen from the figure that, even for 200 trials, the maximum angle of 0.7 degrees is never approached for the nominal case. A single value approaches 0.5 degrees, while most of the rest are below 0.4 degrees. The mean target pointing error is indicated on the figure as a horizontal line through the data.

As was mentioned above, the main reasons for the target pointing error are imperfect knowledge when entering the constraints, and unmodeled accelerations 'pushing' the vehicles off of their 2-body trajectories while no measurement updates can be made. If the times that the constraints are effective remains the same, and they largely do (particularly the shadowing constraint, as the target is in eclipse for the same amount of time each orbit), then one would expect the target pointing errors to worsen as the range to the target decreases. This is due to the fact that, as range decreases, the same position error results in a larger angle error. This is clearly shown

Figure 9-6: PG Nominal case: position dispersions at maneuver times

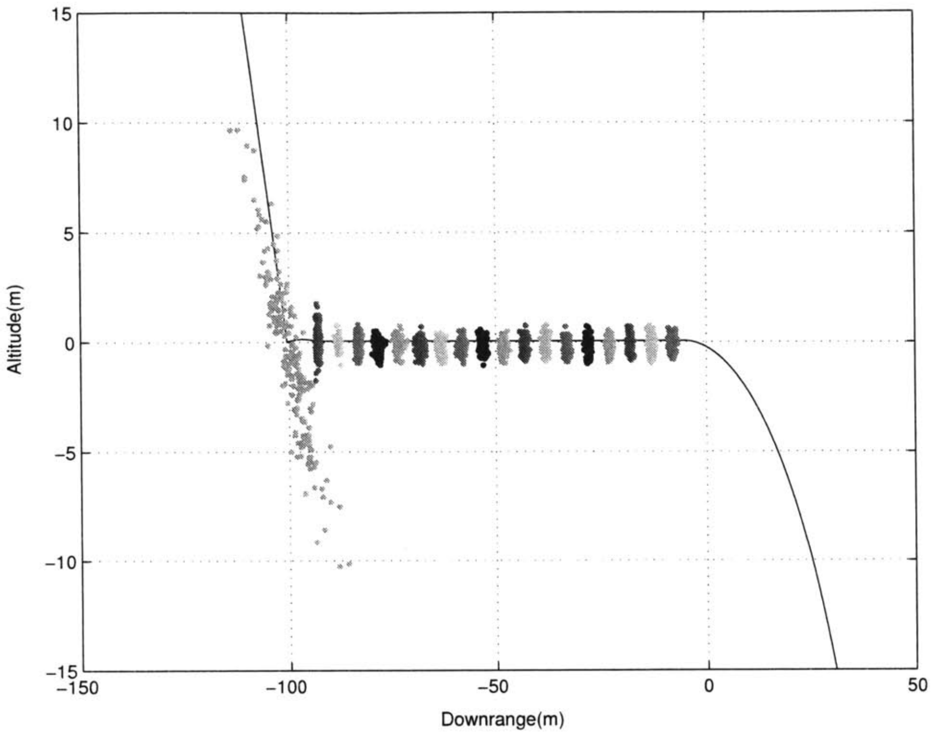


in the figure, as the last two constraint exits are at a significantly closer range, and the recorded target pointing errors are significantly larger than the first six which occur on the football orbit.

Section 9.1 includes a description of position dispersions, and their significance for the present study. Figure 9-6 presents a plot of position dispersions for the nominal case, where the dispersions are plotted at maneuver times. Recall that in the Pre-computed Gain implementation, all maneuvers happen at the same absolute times in every Monte-Carlo trajectory, so this is a meaningful way to collect the data.

Of the corrective maneuvers that happen near zero altitude on the football orbit, the collection of points that has a large positive slope (near vertical) represents the first corrective maneuver to happen at that point. The position dispersions there are largely due to the initial conditions of the simulation, and subsequent corrective maneuvers on the football orbit all have similarly oriented ellipses resulting from corrective maneuvers and orbital dynamics. The dispersion ellipse at the maneuver to

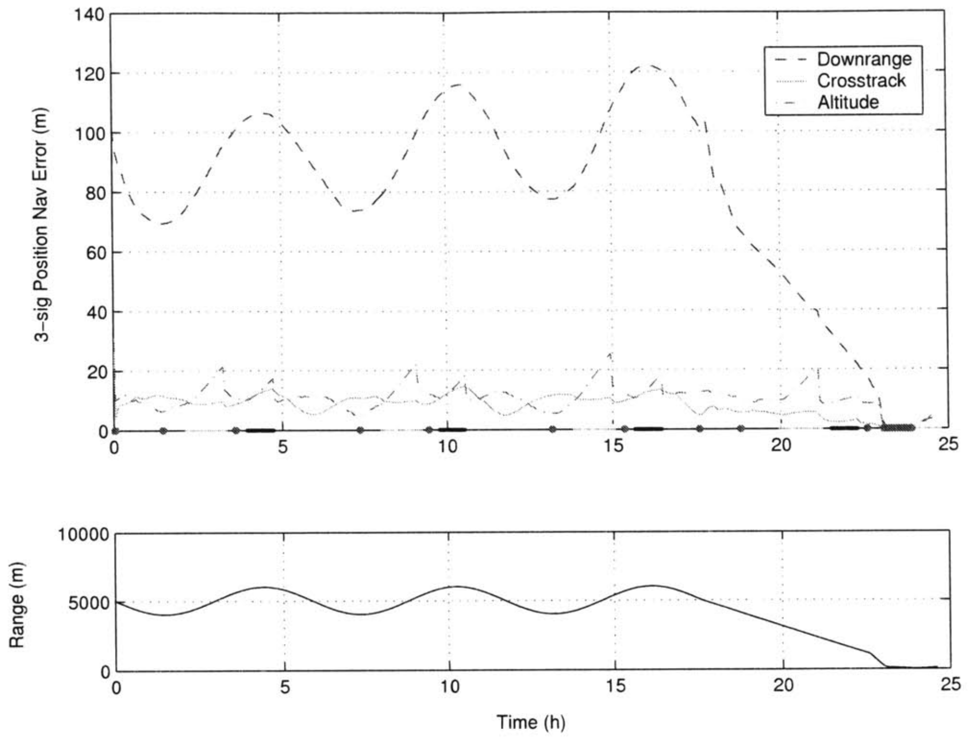
Figure 9-7: PG Nominal case: glide slope position dispersions at maneuver times



enter the fast transfer has a distinctive shape, indicative of the tendency for vehicles on higher coelliptics to have a larger downrange velocity. Thus, those trajectories where the coelliptic ended up a little 'higher' than nominal covered slightly more distance, while those with a lower coelliptic covered slightly less distance than the nominal. This trend is clearly evidenced by the ellipse.

An enlargement of the glide slope portion of the trajectory is shown in Figure 9-7. The initial dispersion ellipse at the end of the fast transfer indicates the end of the "funneling" that can be seen in the hair plot in Figure 9-1. The dispersion ellipse is largely parallel to the trajectory, with an elongation that indicates a small amount of improper timing of the maneuver to end the transfer, since the maneuver time is predetermined and cannot be altered with conditions. Subsequent dispersion ellipses are significantly smaller, generally contained in about 3 m of downrange distance and 1.5 m of altitude. Note again that the axes on this figure are not square, so the dispersion ellipses are significantly 'flatter' than they appear.

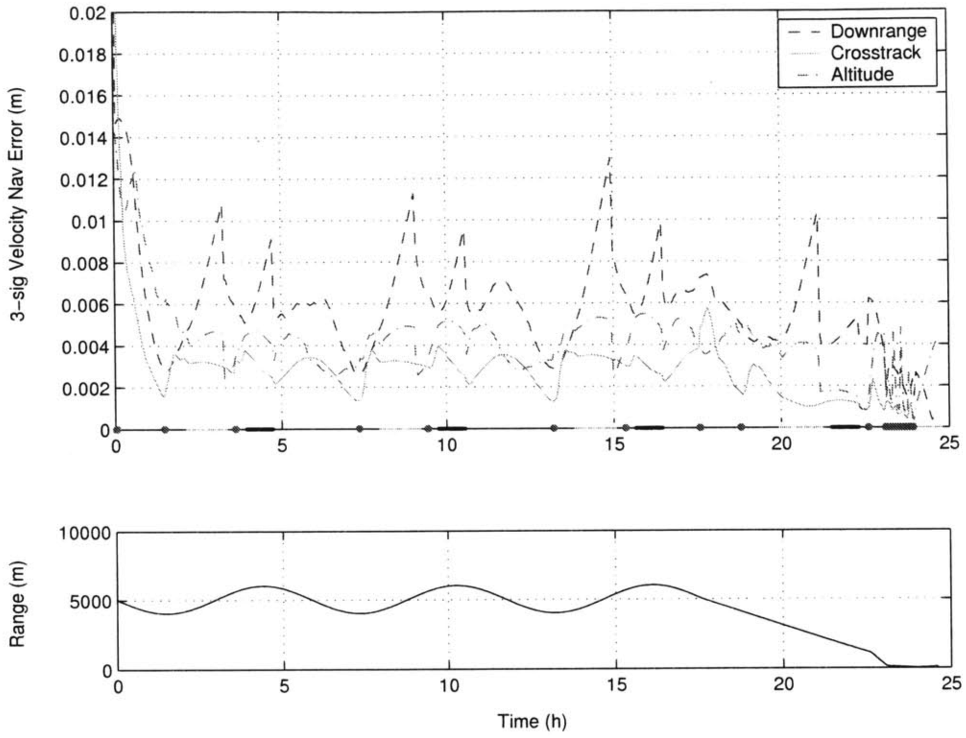
Figure 9-8: PG Nominal case: 3-σ position navigation error



Section 9.1 above introduced navigation error as the difference between the estimated state, maintained by the flight computer, and the 'true' state, maintained by the environment model. The position and velocity navigation errors can be shown for the Precomputed Gain system in a very intuitive manner if placed in the LVLH relative coordinate system. Figure 9-8 shows the position navigation error in the LVLH coordinate system, and Figure 9-9 shows the velocity navigation in the LVLH coordinate system. Both plots include a correlated range plot, as it is clear that the range plays an important role in the navigation error.

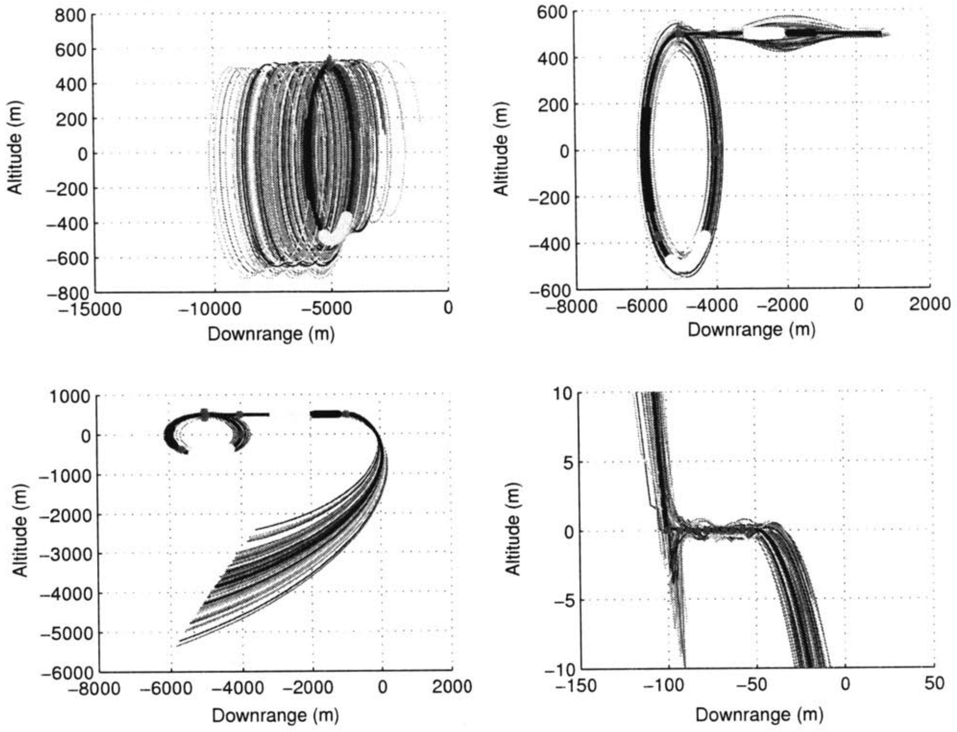
It can be seen in both plots that the altitude navigation error increases while no measurements are being taken, in the Sun angle or target shadowing constraints. Additionally, it is clear that performing a maneuver has a direct effect on the velocity navigation error, and in several places an effect on the position error as well. As the range decreases dramatically towards the end of the trajectory, the quality of the range measurement increases, and the position error improves significantly.

Figure 9-9: PG Nominal case: 3-σ velocity navigation error



It has been mentioned that the passive abort outlook for a given trajectory is of interest to designers, and can be investigated using hair plots such as the one shown in Figure 9-1. The baseline rendezvous trajectory has been designed with exceptional passive abort in mind. There are several places along the reference trajectory where an orbiter failure (power failure, propulsion failure, computer failure, etc) would create different kinds of abort trajectories. In particular, the orbiter could fail: on the football orbit; on the coelliptic approach; after performing the fast transfer insertion burn; and, traveling down the glide slope. Figure 9-10 is a summary of the passive abort outlook for the Precomputed Gain implementation using the nominal parameters, as shown by 200 trials of simulation. The baseline trajectory was modified to represent the four failed cases listed above, where all trajectories are still propagated for the same total length of time as the nominal. In every scenario, it can be seen that the now inert chaser vehicle has virtually no chance of ever striking the target.

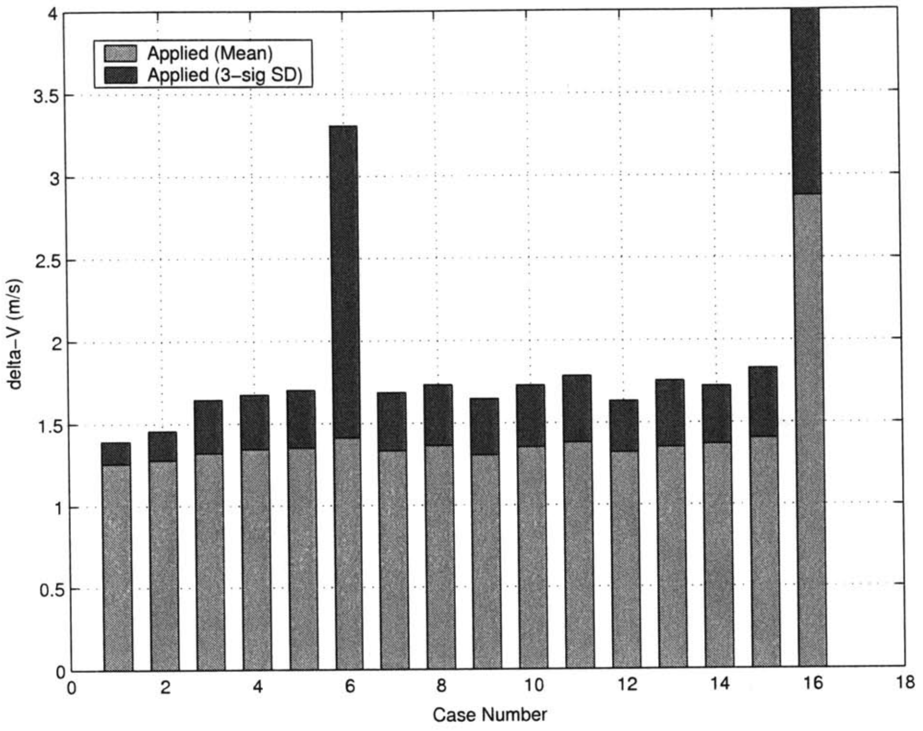
Figure 9-10: PG Nominal case: passive abort outlook



Recall that the football orbit shown in the upper-left sub-plot has a properly phased out-of-plane component that greatly reduces the chances of a collision, even if the football orbit drifts all the way to the target. The large hops initiated in the bottom two plots, if propagated for a longer period, carry the chaser vehicle further away from the target, increasing downrange separation. The coelliptic trajectory, shown in the upper right subplot, will clearly pass the target with a wide margin and then continue to increase downrange separation until command and control is restored.

At some point along the glide slope, shown in the lower right plot, the chaser vehicle must be "committed" the the rendezvous operation. For this particular scenario, that critical point occurs at approximately 20 meters downrange. Before that point (further downrange), the chaser will fall away in the large negative hop as shown. Inside of 20 meters, however, there is a non-zero (and increasing) chance that the chaser will hit the target if control is lost.

Figure 9-11: PG Comparison of total applied Δv



Performance Envelope

All of the cases described by Table 9.12 were simulated using the Precomputed Gain implementation. The section above presents the results for the nominal case, and introduces the types of analysis that can be performed for a single case. In this section, data of the type shown above will be compactly represented and compared for all of the cases run on the Precomputed Gain system. Because they are a relatively qualitative metric and cannot be condensed at all, hair plots will not be shown for the rest of the cases.

The first metric presented above for the nominal case was the applied Δv, which will be a good starting point for this comparison as well. Figure 9-11 is a bar chart showing the magnitude of the total change in velocity required for each numbered case. The 3-σ standard deviation is stacked on the mean total Δv. The first thing to notice is that the total required Δv for case 6 is much larger than the previous cases, and that the required Δv for case 16 runs off of the plot (this figure has been scaled to show detail in the more reasonable runs). For the reader's information, the total 3-σ applied Δv in case 16 is 26.1 m/s. These two cases that statistically could require a large amount of Δv actually define the *performance envelope* of the Precomputed Gain system.

In case 6, only one Monte-Carlo trajectory out of 200 diverges (meaning that the navigation filter and targeting algorithms were not able to perform a successful rendezvous). Large amounts of fuel are used as the flight computer struggles to reign in the errant case, but since the maneuver gain matrices are precomputed around a reference trajectory, they become increasingly inadequate when the chaser has significantly departed from the nominal. Several 200 trial cases were run to verify that the small number of divergent trajectories seen in case 6 were not a statistical anomaly, and in each case the results were the same or a little worse: about 1% of the trajectories were found to diverge. Interestingly, this case represents a "properly tuned" flight computer, where the filter is properly configured to expect the characteristics of the environment. Additional cases were run using the same filter configuration but a slightly more benign environment, and these cases also contained a small number of divergent trials. This suggests that the filter configuration in case 6 is just too conservative to function reliably, regardless of the characteristics of the environment.

The primary reason for the divergence seen here in case 6 is likely to be the nonlinearity introduced by the camera's FOV constraint. Recall from Section 5.2.4 that the camera is limited to taking measurements when the target is in the field of view. While this seems natural and obvious, it is not obvious to the filter. That is, the Kalman gain calculated for a given measurement time might be able to operate on angles that would put the target outside of the field of view, but the camera would return no data and an update could not be made. The filter does not expect this ceiling to the measurements. In the Precomputed Gain system, this limitation can be a serious problem, because each Kalman gain matrix is calculated on the assumption that previous measurements were appropriately incorporated into the state estimate. When a measurement is missed, the Kalman gain matrices that get used when measurements are restored are then inappropriate, possibly enough to cause the filter to diverge as seen in case 6.

Case 16 represents an "improperly tuned" filter that is expecting the nominal levels of execution error and unmodeled accelerations, but the environment is configured for the highest level of execution error studied, and a very large level of unmodeled acceleration. This case represents one limit to the severity of the environment that can be accommodated by the nominal flight computer configuration. Of course, there will be another limit using a maneuver execution error model larger than any studied here. It would be interesting to fully populate Table 9.12 (perhaps even an expanded version of this table) with experimental trials and more completely define the performance envelope, however the intensity of the deterministic method used in this study prevented such an analysis.

Another interesting thing to note about Figure 9-11 is that the required Δv to perform the rendezvous increases slightly for the properly tuned filters (cases 1 through 6), as the levels of maneuver execution error and unmodeled accelerations increase. The increases are slight, however (with the obvious exception of case 6), and are probably not of concern to designers as long as the required Δv is bounded.

When speaking about the nominal case above, the relative position navigation error was shown in Figure 9-8 as a tool to help describe the performance of the algorithm, given the nominal configuration. The same analysis can be performed on every case under study, and by plotting the results on one figure a very useful tool emerges to compare the different configurations. Figure 9-12 is a plot of 3-σ downrange relative position navigation error generated from 200 Monte-Carlo trials each, for cases 1 through 16. Immediately upon viewing this plot, trends are detected that were shown in the Δv plot above. Two lines, marked with a circle and a triangle and labeled with their case numbers, are seen to diverge towards the end of the trajectory. Figure 9-13 is an enlargement of the final portion of the trajectory.

As mentioned before, for the data shown in the figure only one trajectory out of 200 diverged for case 6. Since the plotted values are the standard deviation of the navigation error, however, one divergent case has a dramatic effect on a metric

Figure 9-12: PG Comparison of downrange position navigation error

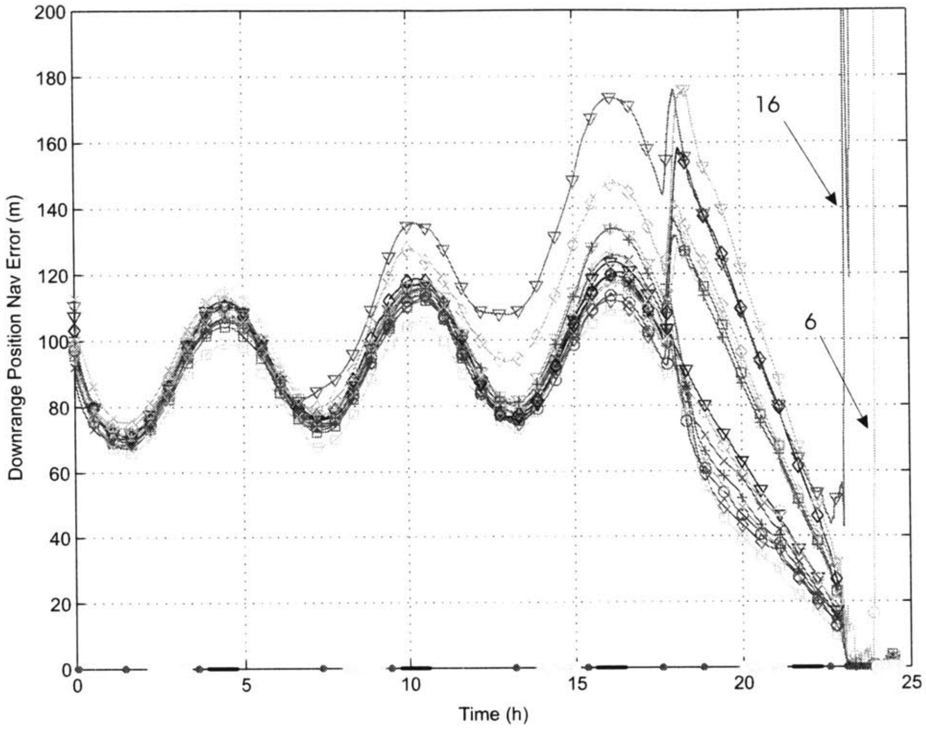


Figure 9-13: PG Comparison of downrange position navigation error, glide slope

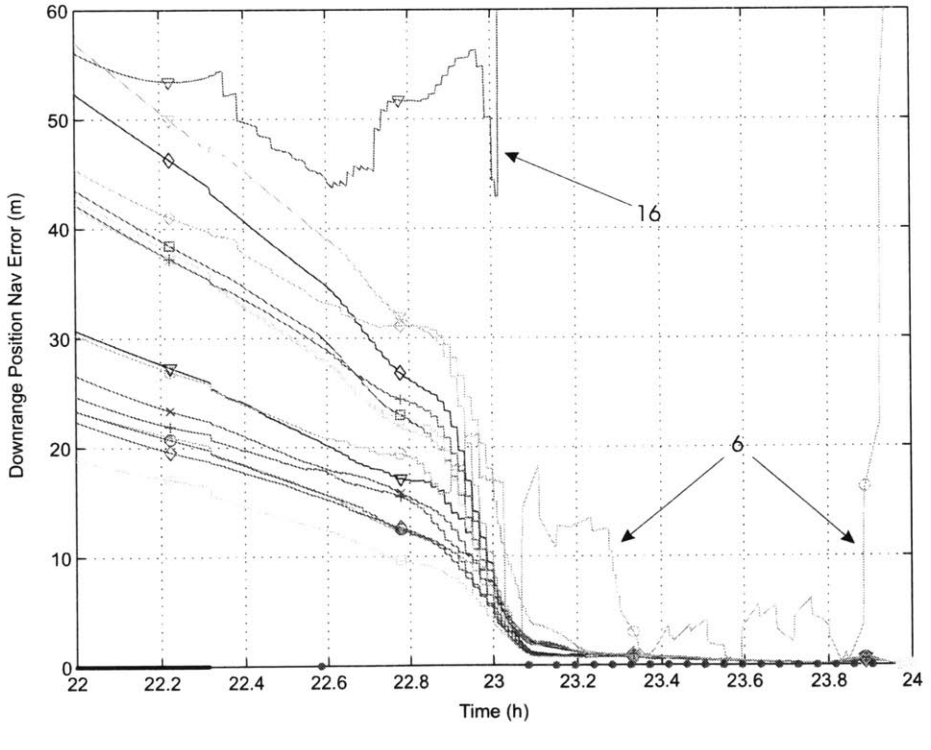
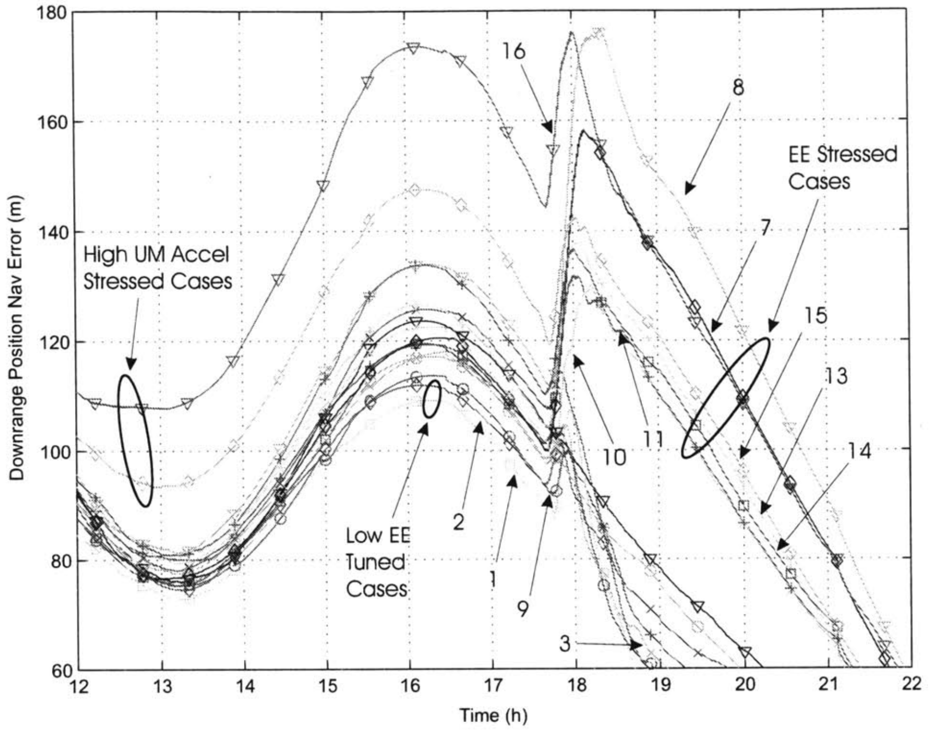


Figure 9-14: PG Comparison of downrange position navigation error, transfer burn



of this type. It is clear that case 6 diverges at the end of the trajectory, beginning on the fast transfer but growing increasingly out of control on the glide slope. This is consistent with trajectory's actual performance if plotted as a single 'hair.' Thus, using a different metric case 6 is again established as one boundary to the performance envelope of the Precomputed Gain implementation.

From Figure 9-12 it is clear that the divergence occurring in case 16 is significantly worse than what is happening in case 6. The navigation error is already anomalous upon exiting the target shadowing constraint, and becomes completely divergent just before beginning the glide slope. Unlike case 6, in this case there are dozens of trajectories that do not complete the rendezvous due to divergence. Thus, case 16 is again shown to be another limit to the performance envelope of the Precomputed Gain system.

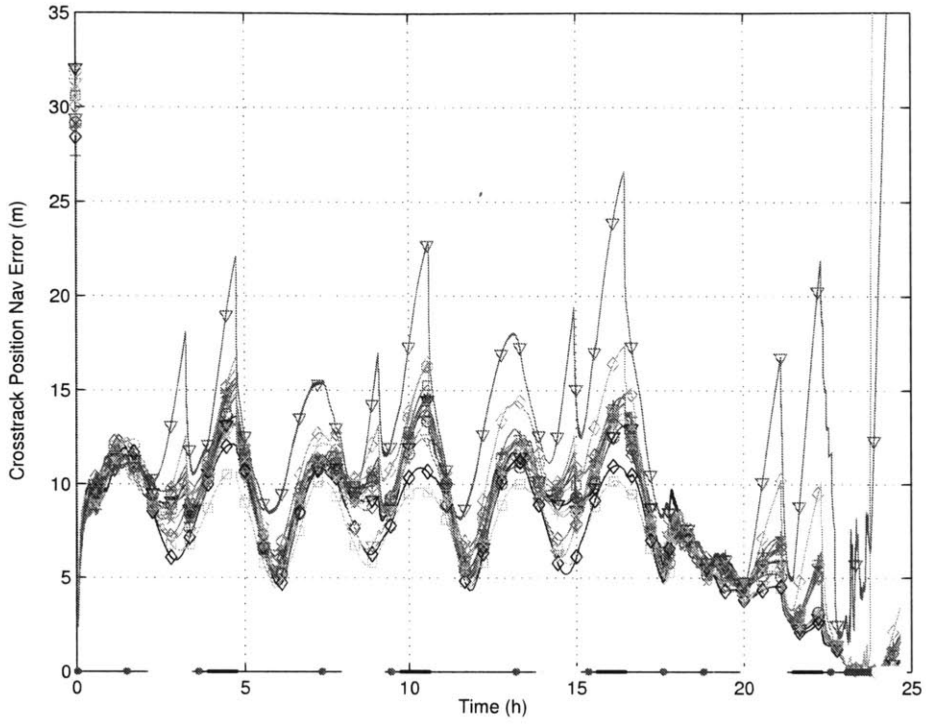
Figure 9-14 is an enlarged view of the downrange position navigation error, with emphasis on the last football orbit and the burn to enter the coelliptic transfer. Most of the distinguishable cases are labeled on the figure, as well as some interesting trends. While the chaser is stationkeeping in the holding football orbit, the maneuvers required are small and infrequent. During this time, the unmodeled accelerations dominate the position navigation error. It can be seen that cases 1 and 2, the properly tuned filters with the lowest maneuver execution errors, have the best navigation at the end of the football orbit. Cases 15 and 16 are improperly tuned and have very high levels of unmodeled accelerations, and can be seen to have the worst navigation at this same time.

The maneuver to begin the coelliptic transfer occurs at 17.6 hours, and is marked in Figure 9-14 by a pronounced rise in the downrange position navigation error. The maneuver to begin the coelliptic transfer is largely in the downrange direction, resulting in a relatively large amount of maneuver execution error in that direction, and thus a velocity navigation error.[6] Over time, the velocity error translates into position error, which is seen in the figure. Note that the cases with a marked rise in navigation error after the maneuver, namely cases 7, 8, 10, 11, 13, 14, 15, 16, are the improperly tuned filters receiving more maneuver execution error from the environment than the flight computer expects. No case that is properly tuned in execution error shows up in this group. It takes the flight computers in these stressed cases more time to settle on the proper estimate after receiving an error source they did not expect.

As a specific example, compare cases 3 and 13 at 19 hours. Case 3 is a properly tuned filter expecting the nominal execution error model and nominal unmodeled accelerations. Case 13 expects the same level of errors but receives the nominal unmodeled accelerations and the *highest* level of execution error. At 19 hours, case 3 sees approximately 61 m of 3-σ downrange position error, while case 13 sees approximately 118 m of downrange position error. This is almost twice as much position error in the downrange direction due to the improper tuning of the filter, 1.4 hours after the maneuver to enter the coelliptic transfer (note that there is a purely corrective maneuver at 18.8 hours that will contribute to the position error due to execution error

[6]Recall that the maneuver execution error model has a piece proportional to the magnitude and in the direction of the applied burn, in addition to the pointing error and quantization error.

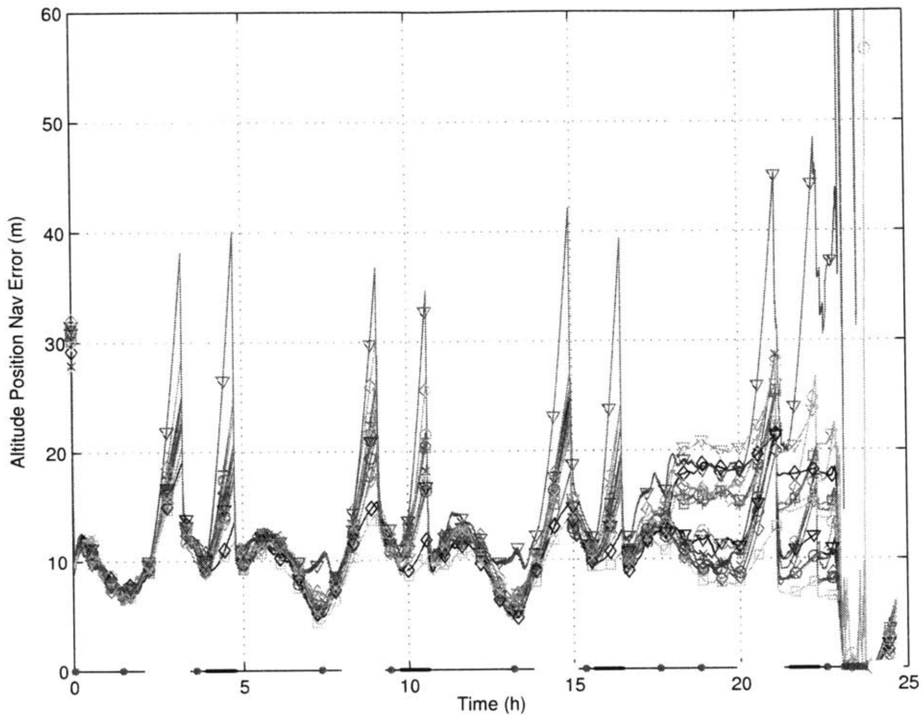
Figure 9-15: PG Comparison of crosstrack position navigation error



and improper navigation when calculating the burn). Figures 9-15 and 9-16 present the crosstrack and altitude components of the position navigation error, exhibiting the same trends as discussed above. Notice that the massively divergent case 16, seen on both figures as the highest line denoted by a triangle, shows substantially more error than the rest of the cases. This again culminates in the divergence of many of the trials towards the end of the run.

One of the tools described in Section 9.1 and used to analyze the nominal case above is the target pointing error. For a single case, the target pointing error can be viewed as shown in Figure 9-5, which shows the target pointing error for every trial, every time a Sun angle or target eclipse constraint is exited. An overlay plot for multiple cases would be too cluttered to be useful, so another visualization method must be used. For a particular constraint exit, the mean and 3-σ standard deviation can be plotted as was done before for the required Δv, and in this way multiple cases can be compared in a concise way. It was shown in Section 9.3.1 that the primary

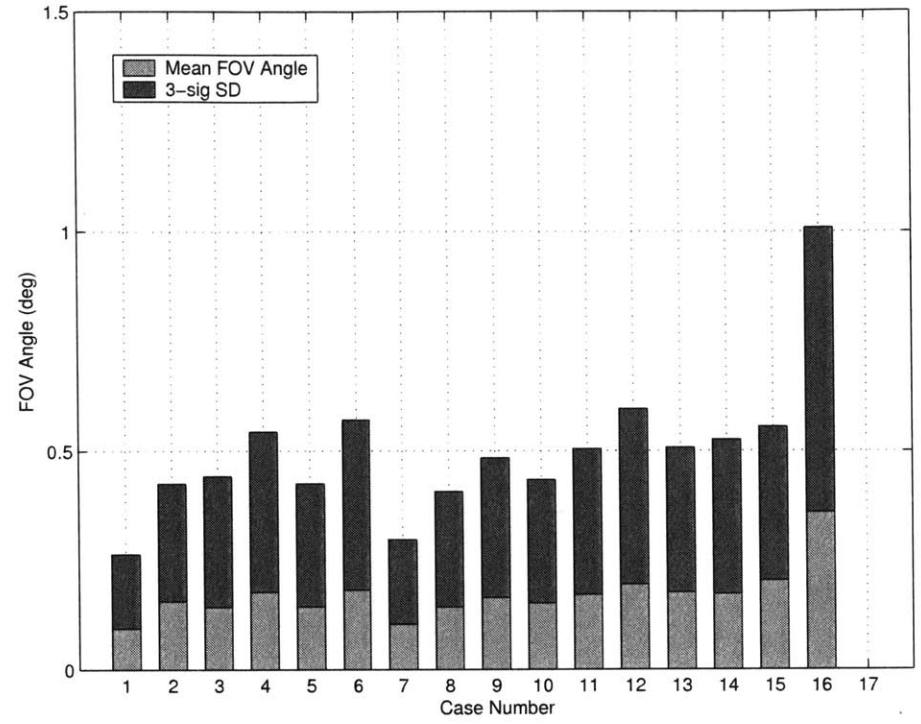
Figure 9-16: PG Comparison of altitude position navigation error



"constraint exits" of interest are numbers 7 and 8, which are on the coelliptic. The other 6 constraints fall on the holding football orbit at 5 km, and no trajectories in this study have had problems dealing with those gaps in the measurement data. Figures 9-17 and 9-18 show the mean and 3-σ standard deviation of the target pointing error for the Precomputed Gain system after constraints 7 and 8, respectively.

At this point it would serve the reader to again refer to the baseline trajectory, shown in Figure 3-7. While traveling down the coelliptic, a Sun angle and target shadowing constraint are navigated in close succession. There is a gap between the constraints near 2 km downrange, during which time 21 measurements are expected by the flight computer on the nominal trajectory, at a rate of one measurement every minute (just over 21 minutes between the constraints). That is, in the precomputed data, there are Kalman gain matrices calculated for 21 residuals, and the flight computer *assumes* that these measurements are made. It is possible, for a given Monte-Carlo trajectory, that small factors such as perturbed target/chaser relative positions can cause one or more of these measurements to be missed, and this can

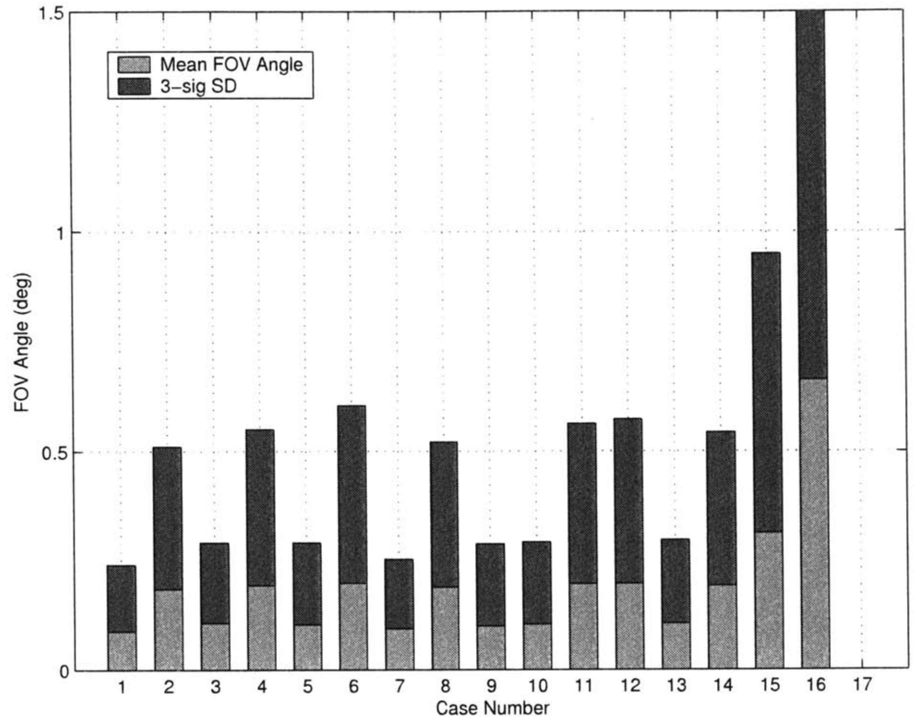
Figure 9-17: PG Comparison of target pointing error after constraint 7



have consequences for filter operation (for example, the second measurement out of the constraint uses a Kalman gain matrix that was calculated assuming that there was a first measurement, which likely made a significant improvement to the state estimate and reduced the covariance matrix).

Looking at Figure 9-17, several trends can be seen that are worth mentioning. First, recall that the NAC has a maximum FOV angle of 1.4 degrees, meaning that the maximum target pointing error must be less than 0.7 degrees (the distance from the boresight to the edge of the frame) to obtain a measurement. Right away it is observed that there is a reasonable statistical chance that Case 16 will encounter pointing errors larger than the camera FOV. This is a precursor to the trend discovered in the position navigation plots above, where the navigation error is seen to begin diverging during the final constraint. It will be noted that, other than this exception, the means of the target pointing errors are small compared to the 0.7 degree limit, and even the 3-σ values are within a reasonably comfortable level. As might be expected, the properly tuned filter with the lowest error levels (case 1) demonstrates the lowest

Figure 9-18: PG Comparison of target pointing error after constraint 8



target pointing errors overall. Case 6, known to have difficulty later in the trajectory, has the worst behavior of the tuned filters, but still in a safe region.

Moving to Figure 9-18, several different but interesting trends will also be noticed. Case 16, a badly tuned flight computer experiencing large levels of unmodeled acceleration, is not improved for having missed several of only 21 possible measurements in the time between the constraints. The 3-σ value for that case is 2.2 degrees, allowed to run off of the plot so that the other cases might be seen more clearly.

It can be seen from the figure that, after a few measurements between constraints to re-establish the state estimate, the dominant factor through the final constraint is unmodeled accelerations. Those cases experiencing large unmodeled accelerations from the environment, tuned or otherwise, have significantly higher target pointing errors than those will small values. Specifically, cases 2, 4, 6, 8, 11, 12, 14, and 15 have approximately twice the mean and 3-σ standard deviation values than their counterparts. Case 15, with the largest unmodeled acceleration from the environment, actually has a statistical chance of a pointing error outside the camera FOV. While

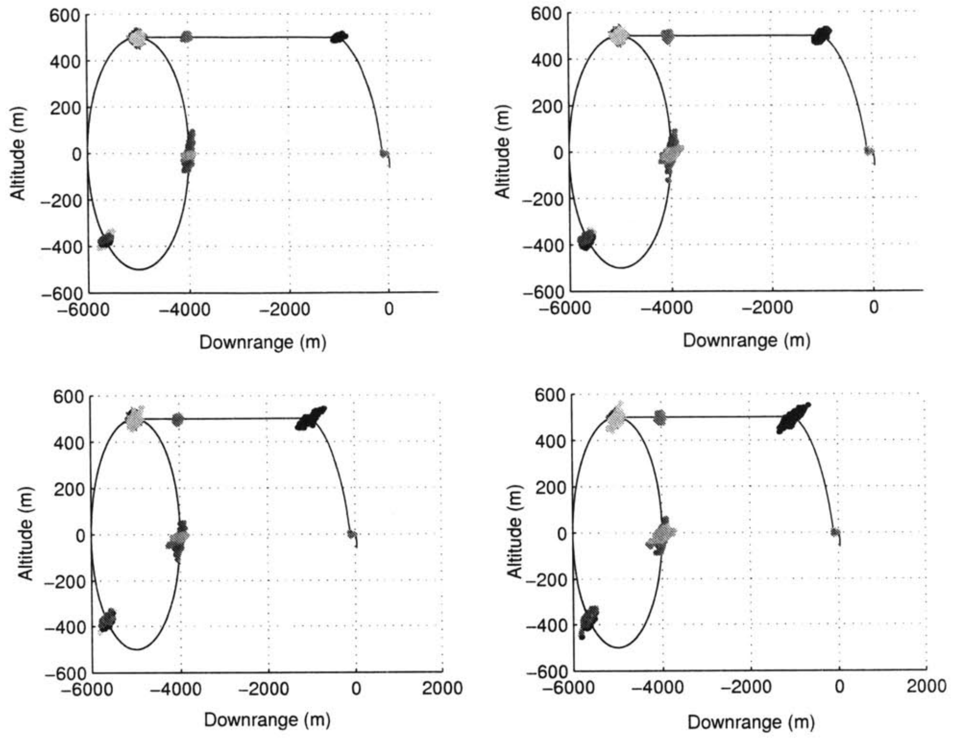


Figure 9-19: PG Comparison of position dispersions: Cases 1, 2, 3, 4

this was seen to happen in several (16 of 200) Monte-Carlo trajectories, the flight computer was able to maintain a reasonable estimate and use the mosaic mode to recover the target; no trajectories diverged.

Figure 9-6 above presents a relative position dispersion plot created for the nominal case. Similar plots can of course be generated for every case, but no simple method has been found to make a useful comparison between all 16 cases in this study. The approach will be to present several of the plots in a grouped fashion to illustrate certain ideas, and it will be noted when many cases are nearly represented by the same visualization.

Figure 9-19 shows the full reference trajectory with dispersions at maneuver times for cases 1, 2, 3, and 4 (moving across the rows). These are the properly tuned filters, in order of increasing sources of error. It can be seen from the figure that, as the unmodeled accelerations and execution error increase, so do the position dispersions. This comes as no surprise, but is a good validation of the expected trends.

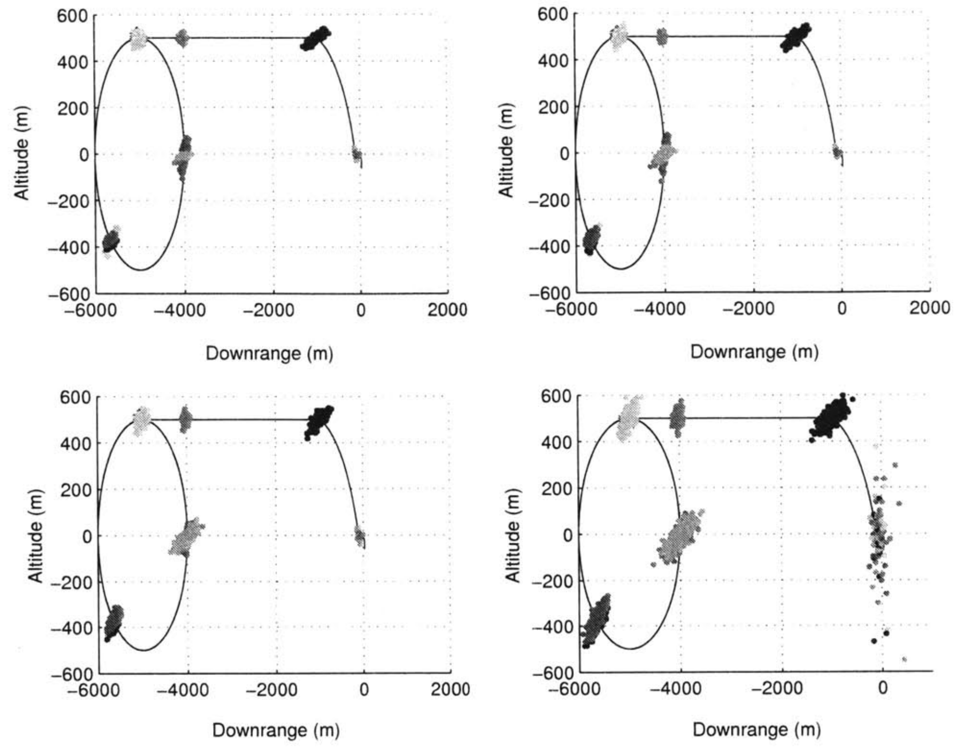


Figure 9-20: PG Comparison of position dispersions: Cases 13, 14, 15, 16

Likewise, Figure 9-20 shows the full reference trajectory with dispersions at maneuver times for cases 13, 14, 15, and 16. These are the stress cases associated with the worst maneuver execution error in the environment, and the nominal configuration of the flight computer (see Table 9.12. As before, it comes as no surprise that the dispersions grow with the error sources. Recall that case 16 has multiple Monte-Carlo trajectories that diverge on the fast transfer and glide slope. This is shown clearly in the lower-right subplot of Figure 9-20, as the distinct ellipses indicating nominal maneuvers are obscured by the errant cases.

Figure 9-21 shows an enlarged view of the glide slope portion of the reference trajectory, with dispersions at maneuver times for cases 1, 3, 6, and 16. The goal of this plot is to show the extremes of the possible dispersions. Case 1 uses a tuned filter and the best possible environment, resulting in very small dispersion ellipses. The final dispersion ellipse in this case has a downrange dimension of approximately 0.75 m, and an altitude dimension of approximately 0.5 m. The case 2 ellipses are

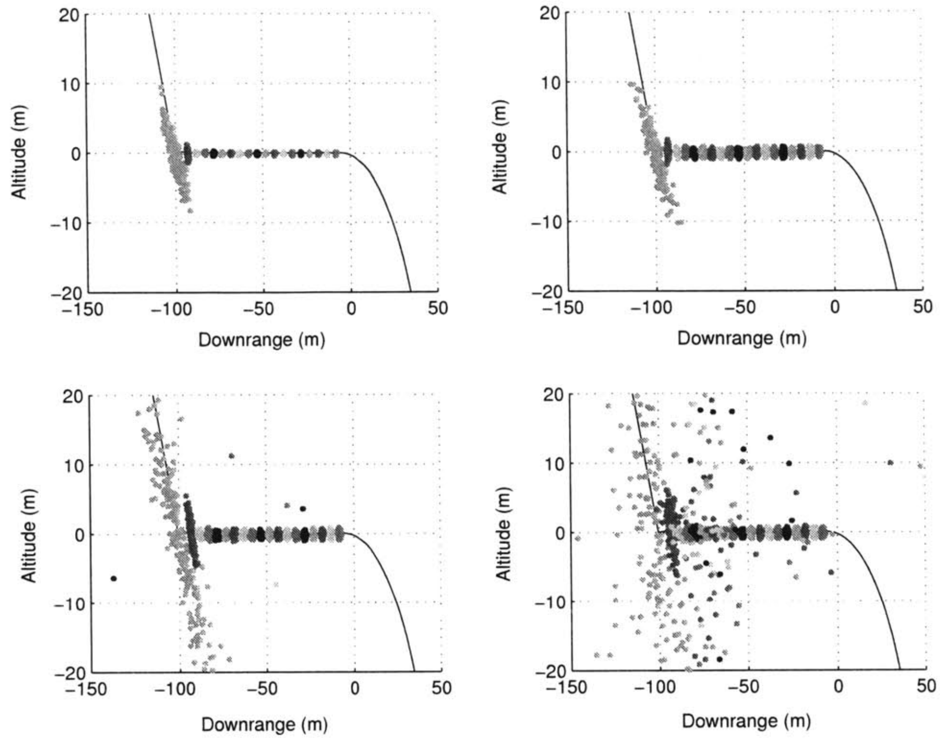


Figure 9-21: PG Comparison of position dispersions, glide slope: Cases 1, 3, 6, 16

very similar to case 1. Case 3 is the nominal case, shown because it *is* the nominal and because it represents the appearance of most of the other trajectories not shown (the only exceptions are cases 6 and 16, also shown in the figure). Case 6 has been discussed previously as one boundary of the performance envelope, containing one trial that is divergent. This divergent trial is evident on the figure, which otherwise looks very similar to case 3 except in the first two ellipses. Finally, case 16 is the divergent case, another boundary of the performance envelope for the Precomputed Gain system. Case 16 contains many divergent cases, clearly evident on the figure as wildly misplaced position marks.

9.3.2 Extended Kalman

This section will follow the same organization as Section 9.3.1 above. The nominal case will be presented first, orienting the reader to the type of results that are generated by the Monte-Carlo analysis tool, applied to the Extended Kalman filter and onboard targeting algorithms. This will also provide a detailed presentation of results for the parameters that seem most reasonable for the MTO demonstration. Specific differences between the Extended and Precomputed Gain systems will be noted in their place. Following the nominal case is a comparison of all 16 cases listed in Table 9.12, including analyses of mean pointing error, total Δv required, navigation error, and position dispersions. A general comparison of the Extended and Precomputed Gain systems is included in Section 9.3.3.

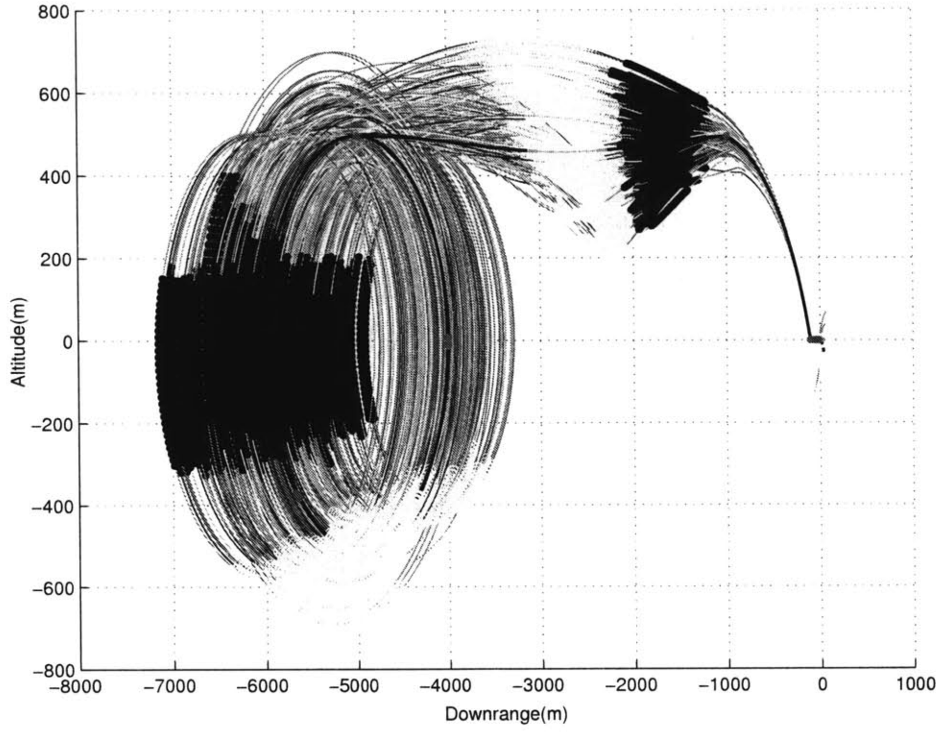
Nominal Case

The **nominal case** is defined by the parameter sets shown in Section 9.2.2. Both the flight computer and the environment model are configured to those parameters, resulting in a "properly tuned" filter using expected mission values. Figures 9-22 through 9-24 show several views of a *hair plot* for a 200 trial Monte-Carlo run on the nominal case. The hair plot is generated by plotting all 200 trials on the same figure, and has proved to be an excellent way to view the data and make qualitative assessments of algorithm performance. Figure 9-22 shows a full view of the trajectory in the downrange and altitude directions of the LVLH coordinate system. The reference trajectory is indicated on the plot by a heavy black line in the midst of the Monte-Carlo trials, and the target shadowing and Sun angle constraints are indicated on each trajectory by series of heavy dots and light circles, respectively.[7] If the reader finds the reference trajectory difficult to see, recall that the baseline rendezvous trajectory was presented by itself in Figure 3-7.

At first glance, a comparison between Figures 9-22 and 9-1 would suggest that

[7]Unlike in the Precomputed Gain system, it is not at all obvious that each trajectory enters the Sun angle and target shadowing constraints at nearly the same time as the reference trajectory. Thus, the constraints are plotted on the individual trajectories.

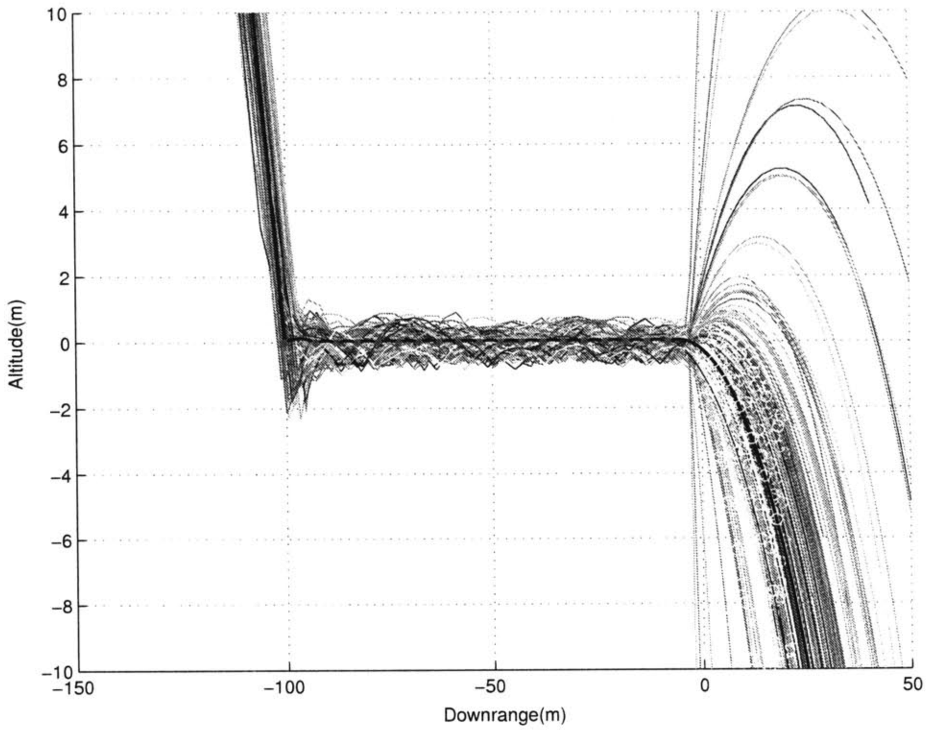
Figure 9-22: EX Nominal case: downrange vs. altitude hair plot



the Precomputed Gain system is far more robust than the Extended system. In fact, quite the opposite is true. The wide range of possible trajectories indicated by Figure 9-22 is a tribute to the flexibility of the onboard targeting algorithms and the linearization of the Kalman filter around the state estimate. It is not necessary to remain close to the "nominal" trajectory as it was in the Precomputed Gain system, so the targeting algorithms can be designed with other priorities, such as limiting the number of required maneuvers and perhaps the total fuel expended.

The reader will notice several things about the hair plot shown. First, football orbits of many sizes exist, some apparently drifting towards or away from the target. This is due to the 'football orbit control' targeting algorithm, which is designed to prevent *large* amounts of drift towards the target, but with no priority to return to the nominal path. Some amount of drift is allowed by the small maneuver constraint, which will veto very small maneuvers calculated to perfectly match the energies of the target and chaser. Another thing to note is that the reference trajectory shown is modestly different from the baseline trajectory of Chapter 3. There is a slight dip in

Figure 9-23: EX Nominal case: downrange vs. altitude glide slope hair plot



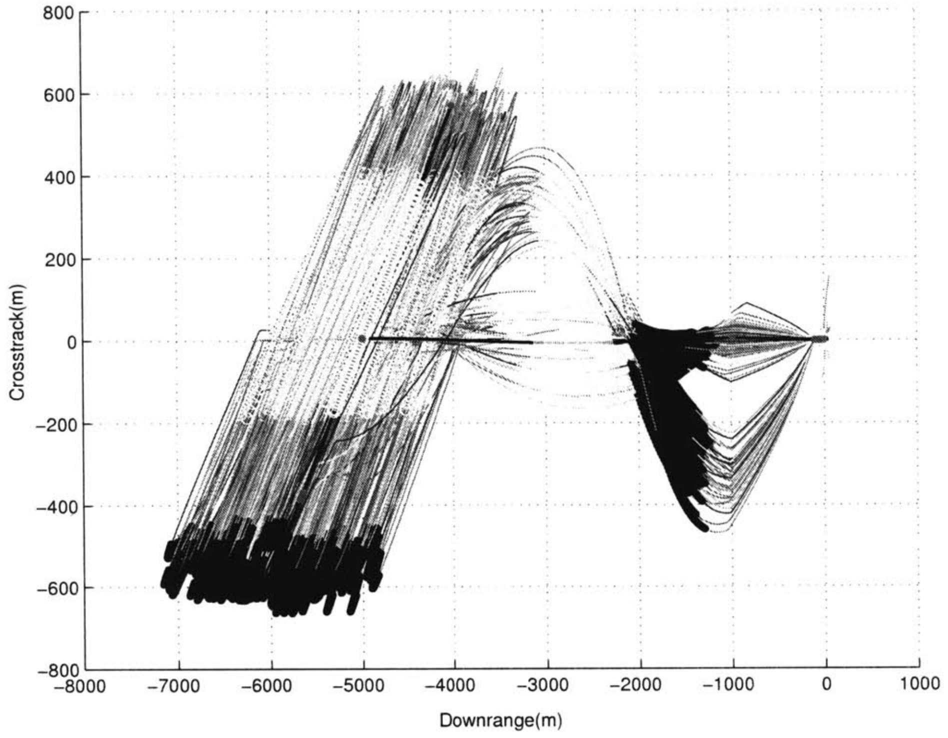
the coelliptic approach, caused by the nature of the coelliptic targeting algorithm from Section 7.4. The coelliptic targeting was designed to be as unconstrained as possible, however some difficulty arose with the target shadowing constraint encroaching on the maneuver to end the coelliptic. The simple solution was to introduce a small bias in the desired transfer time, forcing all of the trajectories to take a little longer on the coelliptic than in the baseline trajectory. This bias manifests itself as a dip in the coelliptic transfer. The reader will also notice the remarkable funneling that occurs on the fast transfer, due to the geometry of the trajectory and the highly constrained Lambert targeting used for this maneuver.

Figure 9-23 is an enlargement of the glide slope portion of the trajectory. The immediate observation here is that the dispersions after the last maneuver seem rather wild, clearly worse than those in the Precomputed Gain system and not at all satisfactory for a rendezvous demonstration. The cause for this large fan of trajectories is again the targeting algorithm on the glide slope. Assuming a rendezvous mission, rather than a demonstration, the glide slope targeting algorithm was designed to target the final maneuver to a point on the v-bar 3 meters from the target (when it is evident from downrange distance that this will be the last maneuver). This is true even if the chaser is, for example, 3.1 meters downrange and 1 meter below the target. In this case, a burn would be calculated to quickly send the chaser up through the 3 meter point, on a trajectory that would afterwards carry it much further downrange than desired for the demonstration. This could certainly be fixed with more sophisticated logic in the targeting algorithm, a task for future work on this problem. Now recall that in the Precomputed Gain system, two things are different. First, the targeting algorithms aim to keep the trajectory close to the nominal. Thus, trajectories where the chaser is very close to the target for the last maneuver are very rare. Second, maneuver corrections calculated using the linearized position correction algorithm are targeted to a point further along the reference trajectory. Thus, trajectories that do end up close to the target for the final maneuver are targeted down the receding path of the nominal trajectory, rather than back to the desired target point.

On the other hand, notice that the dispersions at the beginning of the glide slope are much smaller than those in Figure 9-2. This is because the targeting algorithm to initiate the glide slope is capable of triggering when it is closest to the v-bar, rather than at some pre-determined time as in the Precomputed Gain system.

Figure 9-24 is the same data as shown in Figure 9-22, now plotted for the downrange and crosstrack directions. Here the out-of-plane component of the football orbit is readily seen, as well as the reasonably small amount of drift the chaser sees in the football orbits. It can also be seen that there are apparently two classes of trajectory that have distinctly different amounts of crosstrack. In fact, this is true. Recall that the coelliptic transfer is triggered on the third football orbit when either the chaser reaches the desired coelliptic altitude of 500 m, or it reaches the top of the football without having reached that target altitude. Those trajectories that trigger the maneuver below 500 meters, trigger at the top of the football, where the crosstrack component is the smallest, and thus the crosstrack is largely removed for

Figure 9-24: EX Nominal case: downrange vs. crosstrack hair plot

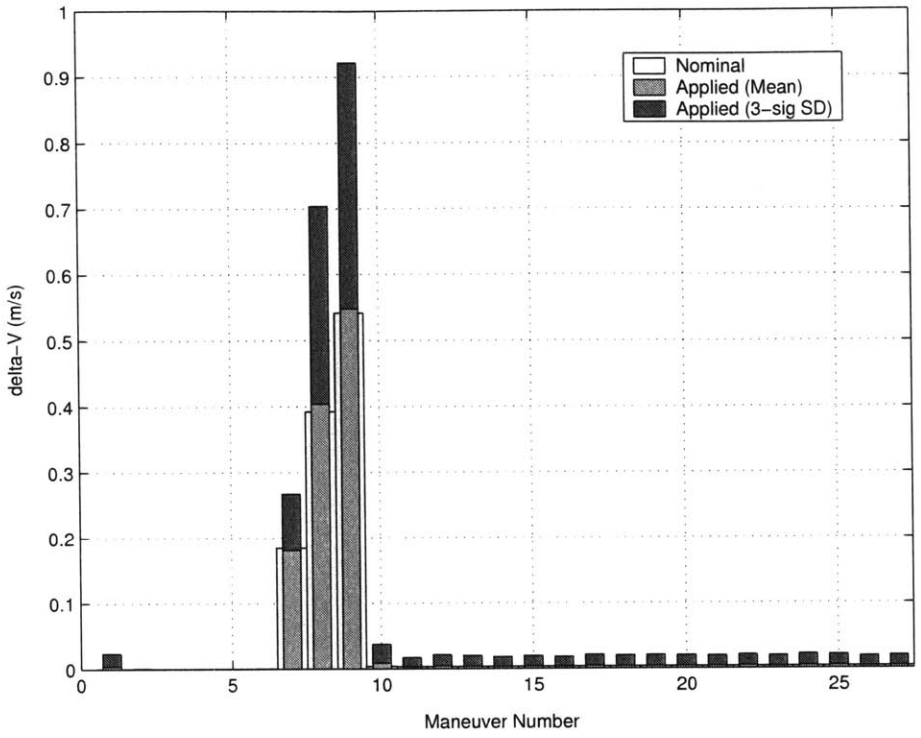


the coelliptic. The maneuvers triggered before the top of the football, however, are performed when a large amount of crosstrack still exists, which must then be removed by further maneuvering.

The next metric for evaluating the nominal case is the applied Δv at each maneuver time in the trajectory. Figure 9-25 presents this data in a bar chart format. The widest bars on the chart represent the "nominal" maneuvers, or the maneuvers that are required to create the nominal baseline trajectory for the Extended system. The other bars show the 3-σ standard deviation of the maneuver magnitude stacked on the mean maneuver magnitude.

There are several key differences between this plot and the equivalent plot shown for the Precomputed Gain system in Figure 9-4. First notice that there are fewer total maneuvers. Since it is no longer important to stay close to the nominal trajectories, several of the corrective maneuvers required in the Precomputed Gain implementation have been removed. This includes one maneuver on the first football orbit and

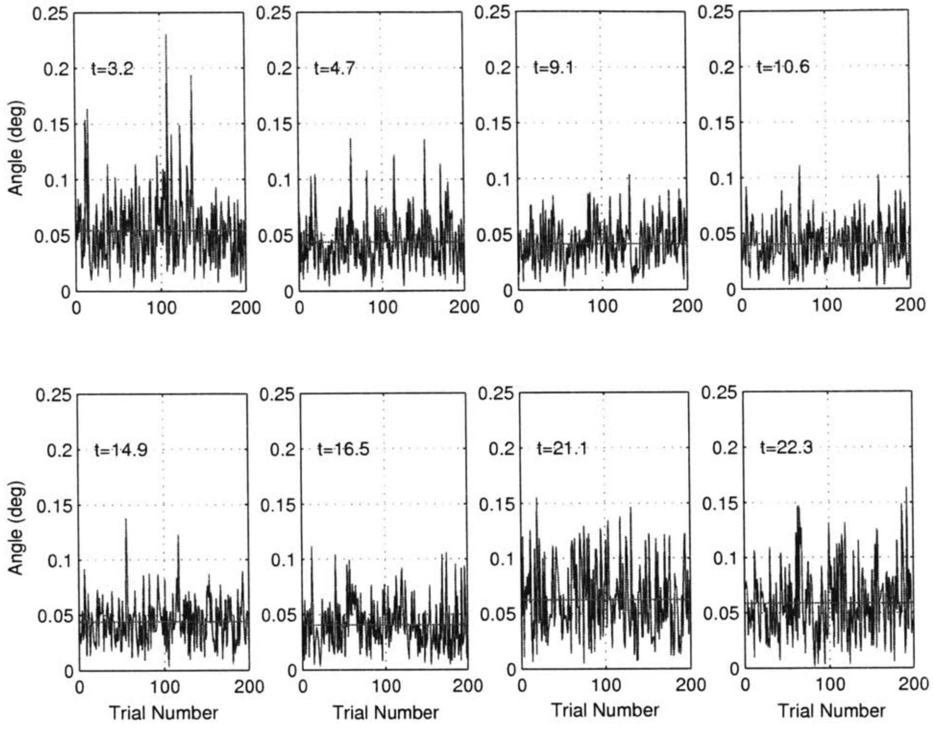
Figure 9-25: EX Nominal case: applied Δv statistics



one maneuver on the coelliptic approach. Note also that, because of the design of the targeting algorithms, virtually no maneuvering is required on the football orbits after the very first maneuver. Once the holding football orbit has been established to a reasonable level of drift, there is no further need to maneuver using the Extended system. It can also be seen that, while the nominal (and hence the mean) required Δv are essentially the same between the Extended and Precomputed Gain systems, the Extended system has a larger standard deviation associated with the nominal maneuvers. This is because, with the flexibility allowed by the targeting algorithms, larger dispersions exist at maneuver times, and it is possible for a wide variety of maneuvers to be required to enter certain parts of the trajectory (including potentially large maneuvers). It is expected that, with additional work on the targeting algorithms, the size of these bars could be reduced.

Figure 9-26 presents the target pointing error as the chaser comes out of each of the 8 constraints on the rendezvous trajectory. Recall that the maximum allowable

Figure 9-26: EX Nominal case: target pointing angle at constraint exits



angle for the target pointing error to still receive a measurement is half of the FOV angle, or 0.7 degrees for the NAC. In each subplot in Figure 9-26, the vertical axis only goes to 0.25 degrees, showing that the target pointing error is easily in bounds for the nominal case using the Extended system. As in the Precomputed Gain case, the highest values are generally found in the last two constraints exits (with a few large errors arising after the first constraint, due to initial conditions), which are on the coelliptic approach and the closest to the target. The actual values for the pointing error after these last two constraints are significantly lower than the values for the Precomputed Gain system, indicating improved pointing in the Extended system due to the estimation of all 20 states and their use in predicting the measurements (thus improving the state estimate and reducing navigation error).

Figure 9-27 shows the position dispersions at maneuver times for the nominal case using the Extended system. It is important that in this plot, unlike the position dispersion plot for the Precomputed Gain system, the marks are not placed on the

Figure 9-27: EX Nominal case: position dispersions at maneuver times

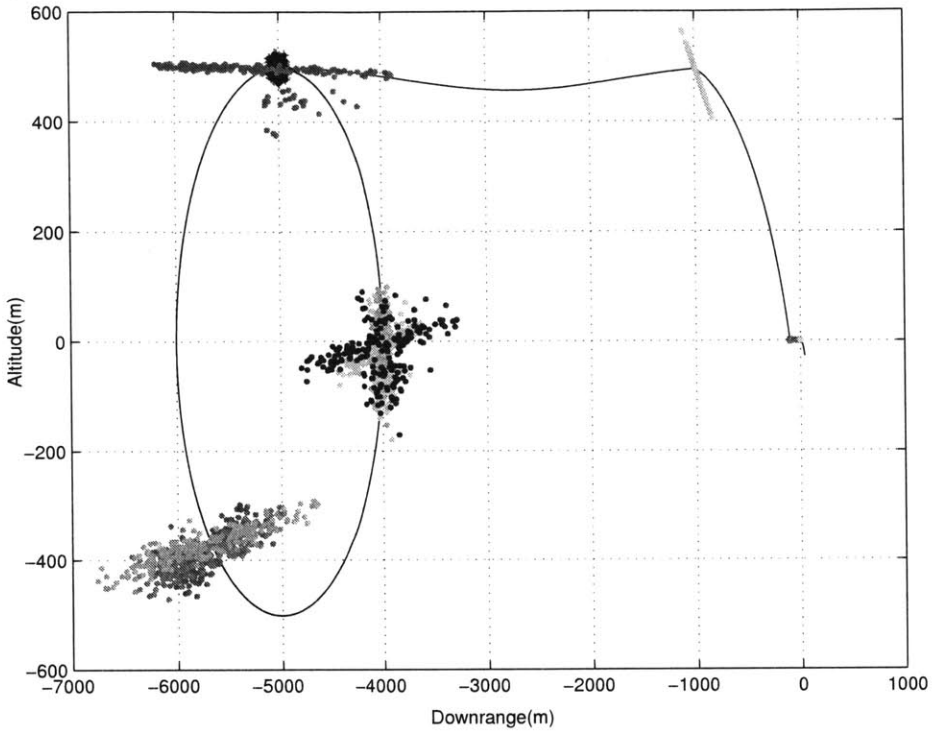
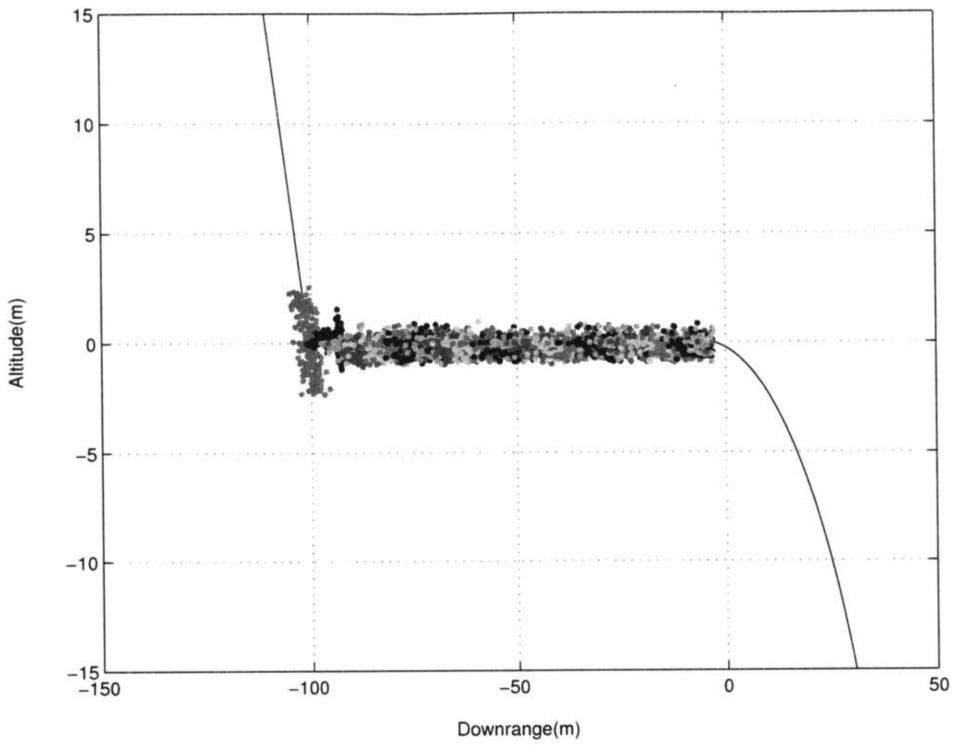


figure at the same *absolute* time for each trajectory. Since the maneuvers are triggered based on many possible properties of the trajectory including geometry, the maneuvers do not necessarily happen at the same absolute time from trial to trial.

On the football orbit the ellipses shown are actually conglomerates of three maneuvers, one on each revolution around the football. The cross-shaped dispersion is evidence of football orbit maintenance maneuvering. The first dispersion ellipse at that point, not easily visible in the grayscale plot, is a vertical ellipse. At this time, some trajectories require a maneuver to match the energy of the target and some do not. Those that do, largely retain the vertical orientation for the next two revolutions around the football orbit. Those that do not disperse into the mostly horizontal dispersion ellipse.

At the top of the football orbit, the maneuver to enter the coelliptic is clearly visible. Most the the trajectories were triggered at the height of the coelliptic, after drifting slightly towards or away from the target. A small scattering of trajectories were triggered at an altitude lower than the coelliptic, at the top of their particu-

Figure 9-28: EX Nominal case: glide slope position dispersions at maneuver times



lar football orbit. The heavy black ellipse at the top of the football represents the dispersions on the initial conditions of the trajectories.

At the end of the coelliptic, a very distinctive ellipse is defined by the position dispersions. This is due to the nature of the trigger for the fast transfer down to the v-bar, as described in Section 7.5. In the two dimensions of Figure 9-27, the maneuver is triggered when the trajectory crosses the line that intersects the starting and ending points of the fast transfer. This line is clearly identified by the dispersion ellipses.

Figure 9-28 is an enlargement of the glide slope portion of the rendezvous trajectory. As was mentioned in the discussion of the hair plot, the dispersions at the end of the fast transfer are significantly smaller than those seen for the Precomputed Gain system. This is because the flexible targeting algorithm to begin the glide slope triggers on the integration time step that is closest to v-bar. The glide slope itself does not contain distinct ellipses like those seen for the Precomputed Gain system.

This is again due to the nature of the glide slope targeting algorithm, which seeks to maintain the relative approach speed down the v-bar rather than seeking particular locations at particular times. Thus, the dispersion ellipses on the glide slope for the Extended system can become muddy as shown in the figure.

Figures 9-29 through 9-32 show the navigation error for the nominal case, using the Extended implementation. Figures 9-29 and 9-30 show the position and velocity navigation error in the LVLH frame, where the trajectories are fully synchronized at their initial times. Figures 9-31 and 9-32 show the position and velocity navigation error in the LVLH frame as well, but here the trajectories are synchronized at the target point of the rendezvous demonstration. The time at which the trajectories cross this point has been dubbed t_{zero}, and this time can be different for each Monte-Carlo trial.

The reason for this split approach is similar to the reason the position dispersion were plotted at *actual* maneuver times, rather than *nominal* maneuver times. Since each trajectory has its own maneuver schedule, based on the conditions of that particular trial, maneuvers can happen at different times and the point of closest approach will be different between the various trajectories. Since, in this implementation, the football orbit control maneuvers actually are triggered at particular times, it is most useful to look at Figures 9-29 and 9-30 to evaluate the navigation error. Towards the end of the trajectory, however, particularly on the glide slope, it makes more sense to look at the figures synchronized to the target point crossing time, t_{zero}. Note that on all four figures the maneuver locations, Sun angle constraint, and target shadowing constraints have all been indicated. This information is an approximation and should only be used for reference, as it is based on the nominal rendezvous trajectory, and each trial performs maneuvers and enters the constraints at slightly different times.

During the three circuits of the holding football orbit, when Figures 9-29 and 9-30 are most valid, it can be seen that the 3-σ navigation error is very smooth in all axes for 200 trials. Recall from the discussion above that there are fewer total maneuvers performed by the Extended system than the Precomputed Gain system, so the veloc-

Figure 9-29: EX Nominal case: 3-σ position navigation error

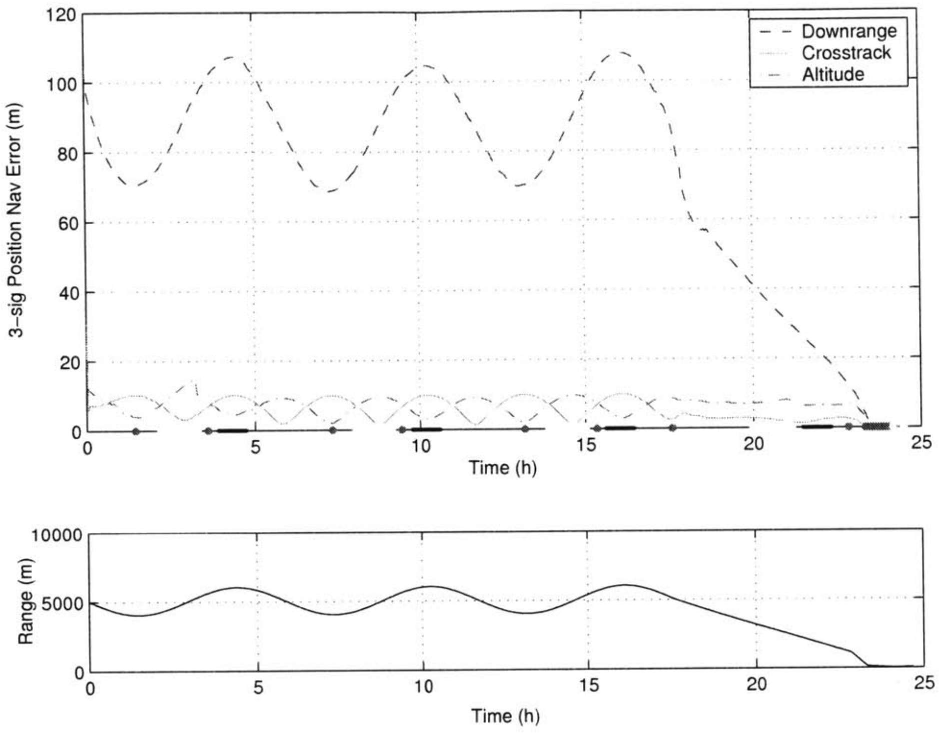
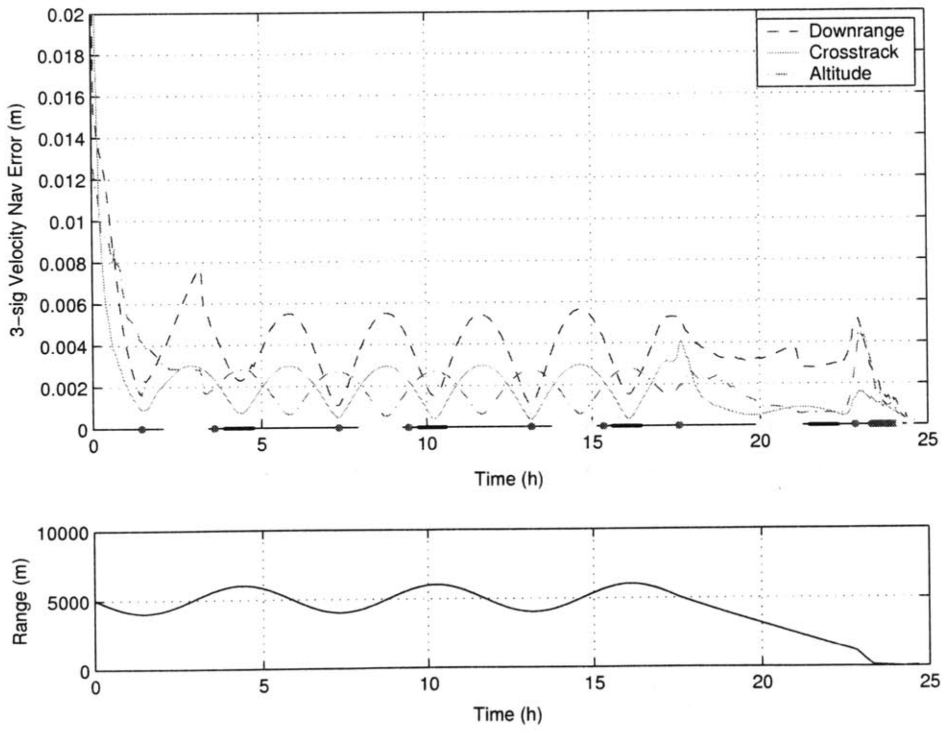


Figure 9-30: EX Nominal case: 3-σ velocity navigation error



ity navigation error is expected to be smoother for the Extended system. Improved velocity navigation upon entering the Sun angle and target shadowing constraints translates into improved navigation on exiting, resulting in a reduction of the navigation error growth during constraints that was seen in both position and velocity navigation error for the Precomputed Gain system.

To consider navigation error for the last part of the trajectory, move to Figures 9-31 and 9-32, where the trajectories have been synchronized at time t_{zero}. The strong correlation between range and downrange position error is clearly evident, as the navigation is significantly improved as range decreases rapidly on the coelliptic and fast transfer. Position navigation error on the glide slope is very small, and velocity error is similar to that seen in the Precomputed Gain system.

Figure 9-33 is a summary of the passive abort outlook for the Extended implementation using the nominal parameters, as shown by 200 trials of simulation. The baseline trajectory was modified to represent the four possible failed cases, where all trajectories are still propagated for the same total length of time as the nominal.[8] The reader will notice that this figure looks very much like the passive abort outlook for the Precomputed Gain system, shown in Figure 9-10, and indeed it should. The baseline trajectory was designed with passive abort in mind, and the basic trajectory pieces, when left to orbital mechanics (i.e. no maneuvering), will exhibit the same trends.

In every scenario, like in the Precomputed Gain system, it can be seen that the now inert chaser vehicle has virtually no chance of ever striking the target. Recall that the football orbit shown in the upper-left sub-plot has a properly phased out-of-plane component that greatly reduces the chances of a collision, even if the football orbit drifts all the way to the target. The large hops initiated in the bottom two plots, if propagated for a longer period, carry the chaser vehicle further away from the target, increasing downrange separation. The coelliptic trajectory, shown in the upper right

[8]Recall from Section 9.3.1 that orbiter failures in the following places on the trajectory would result in unique abort scenarios: on the football orbit; on the coelliptic approach; after performing the fast transfer insertion burn; and, traveling down the glide slope.

subplot, will clearly pass the target with a wide margin and then continue to increase downrange separation until command and control is restored.

At some point along the glide slope, shown in the lower right plot, the chaser vehicle must be "committed" the the rendezvous operation. For this Extended system, like the Precomputed Gain system, that critical point occurs at approximately 20 meters downrange. Before that point (further downrange), the chaser will fall away in the large negative hop as shown. Inside of 20 meters, however, there is a non-zero (and increasing) chance that the chaser will hit the target if control is lost.

Figure 9-31: EX Nominal case: 3-σ capture correlated position navigation error

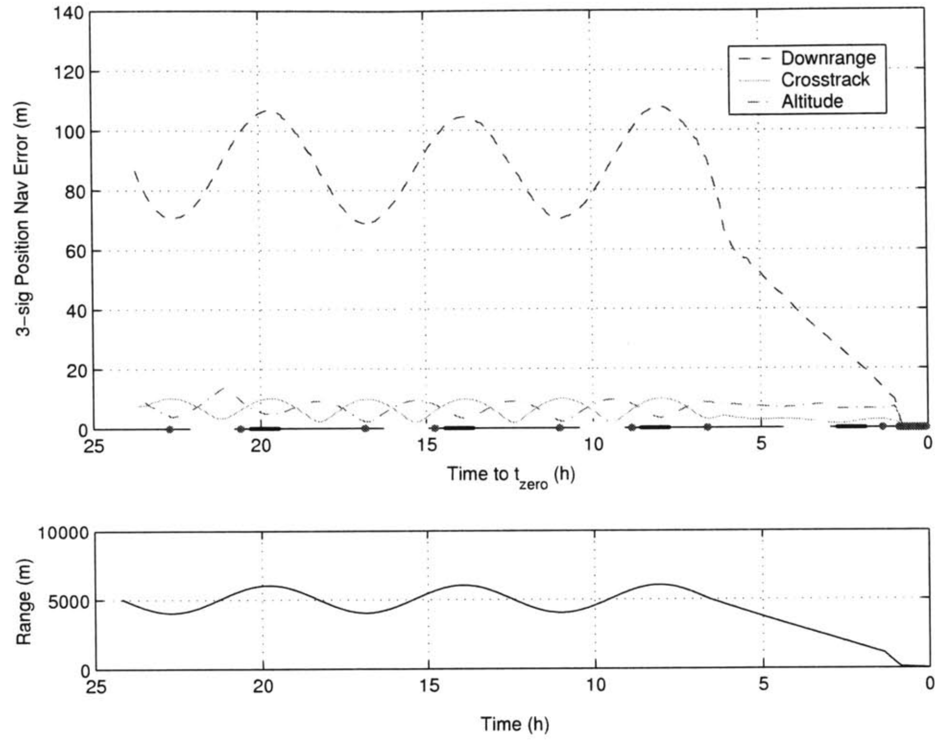


Figure 9-32: EX Nominal case: 3-σ capture correlated velocity navigation error

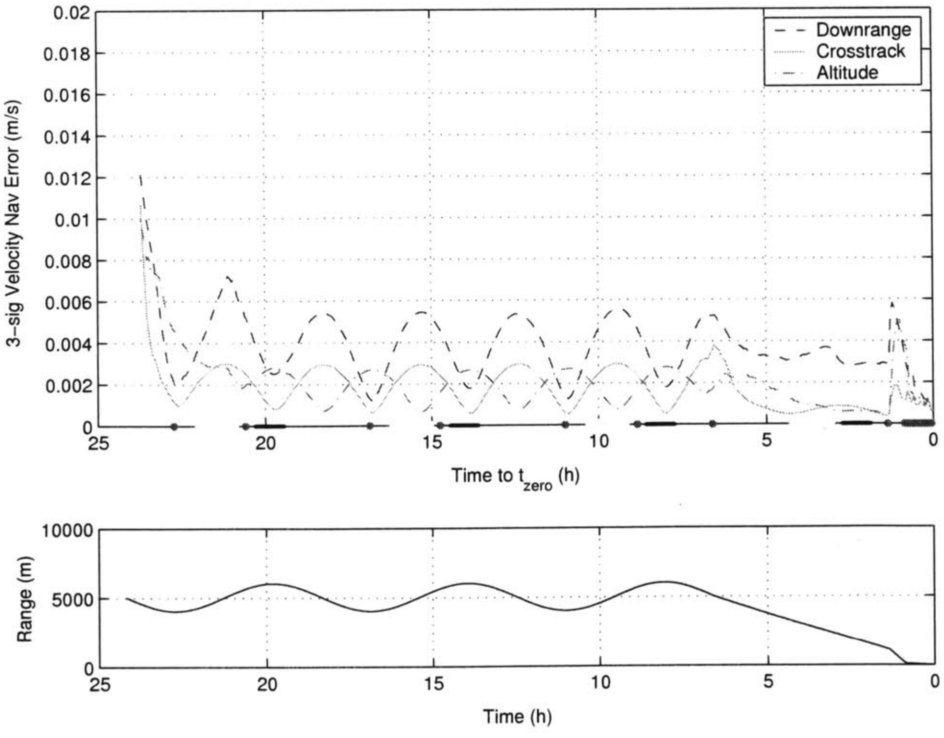


Figure 9-33: EX Nominal case: passive abort outlook

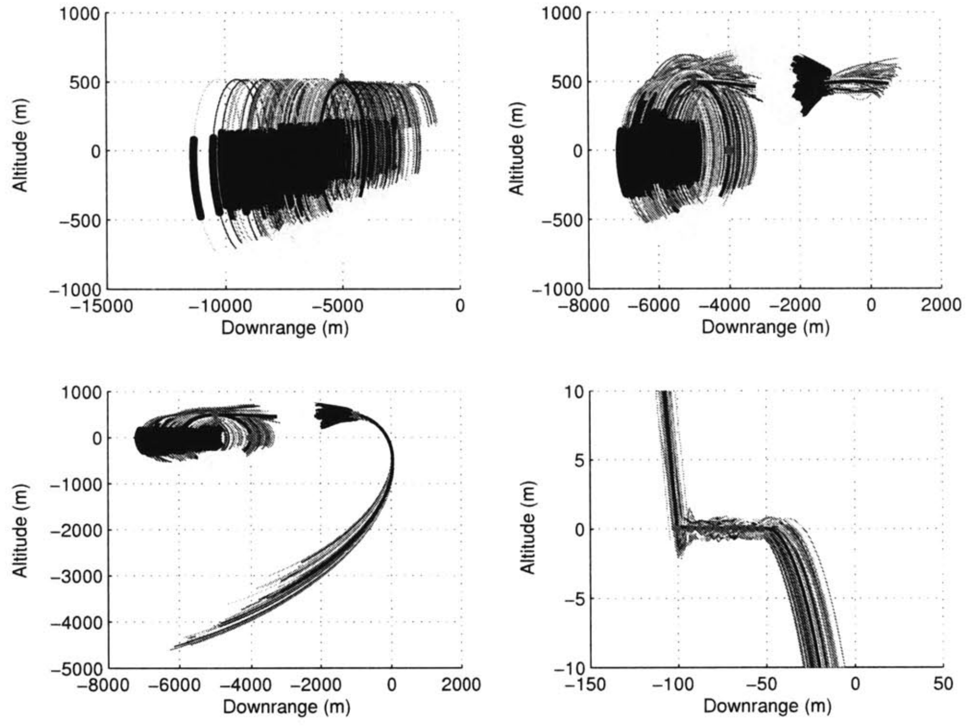
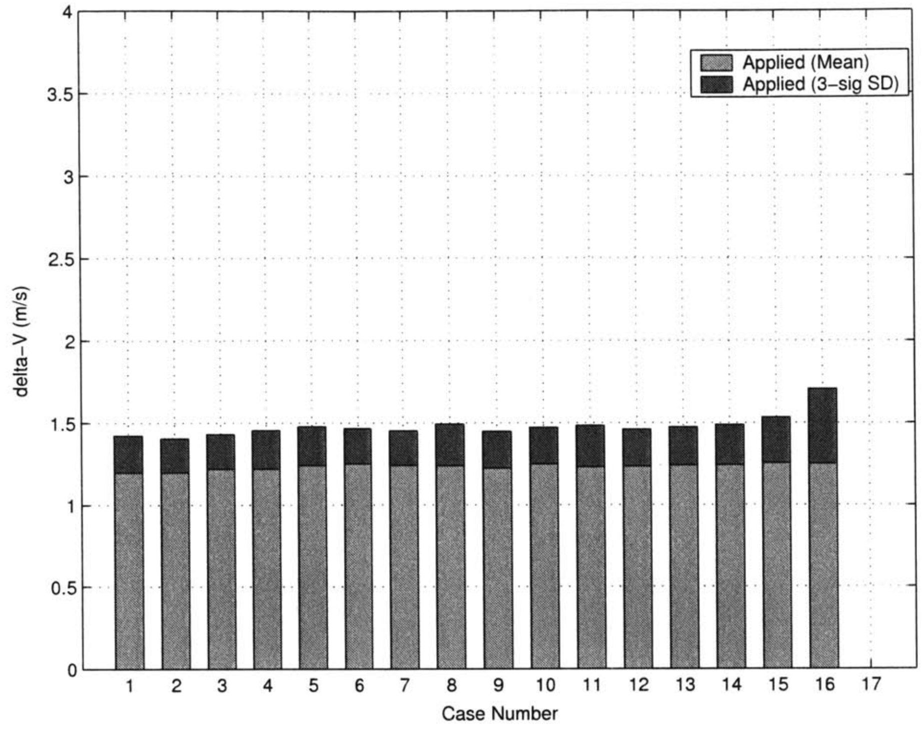


Figure 9-34: EX Comparison of total applied Δv



Performance Envelope

In Section 9.3.2 strong emphasis was placed on the *performance envelope* that was established for the Precomputed Gain system. In this section, data is presented for the cases in Table 9.12 when run on the Extended implementation. Instead of expanding the table to find the points where the Extended system diverged, emphasis is placed instead on showing how this system performs in (and outside of) the envelope established for the Precomputed Gain algorithm. The organization of this section will mirror the Precomputed Gain section, including similar plot axes, so that the reader may quickly compare if desired.

Figure 9-34 is a bar chart showing the magnitude of the total change in velocity required for each numbered case. The 3-σ standard deviation is stacked on the mean total Δv. This plot is particularly uninteresting, as it is immediately obvious that there is very little difference in the mean total required change in velocity between all

16 cases. The small variations, when magnified, do exhibit several expected trends. The total required Δv increases in the tuned filter cases, as conditions worsen from case 1 to case 6. The cases requiring a slightly higher Δv, such as 5, 8, 10, 11, and 15, are cases receiving the highest level of maneuver execution error from the environment model. Only case 16 is seen to require a noticeably larger amount of Δv than the other cases. The reader will recall that this case, as well as case 6, defined the performance envelope of the Precomputed Gain system and performed very poorly. Here the improperly tuned filter in case 16, receiving 4000 m/day of unmodeled accelerations and the largest maneuver execution error model from the environment, only experiences a small rise in the 3-σ deviation from the mean. It will be seen that this harsh environment model does cause several trials of case 16 to enter mosaic mode (5 trials), however the flexibility of the system allows it to get back on track without diverging as the Precomputed Gain examples did. No trajectories diverged in case 6 as they did for the Precomputed Gain system, suggesting that the performance envelope for the Extended system, due to the filter configuration, is slightly more more robust than the parameters for case 6.

In Section 9.3.1, the likely cause for divergence of case 6 in the Precomputed Gain system was given as the nonlinearity introduced by the field of view constraint. Using the Extended system, this is not an issue as described previously. When a measurement is missed, the Extended system simply propagates the error covariance accordingly, and when measurements are resumed, a Kalman gain matrix is calculated based on the proper covariance. Thus, the interpretation of the measurement residuals will be much more appropriate in the Extended system, maintaining sufficiently accurate navigation to prevent divergence.

Recall from the presentation of the nominal case that the position navigation error can be readily presented in two ways: with the trajectories synchronized at the initial time, or synchronized at the time of the final v-bar crossing, t_{zero}. Both types of plots are thus presented in Figures 9-35 through 9-43. The position navigation errors for all 16 cases have been plotted together, with the downrange, crosstrack, and altitude components broken out into separate figures.

Figure 9-35: EX Comparison of downrange position navigation error

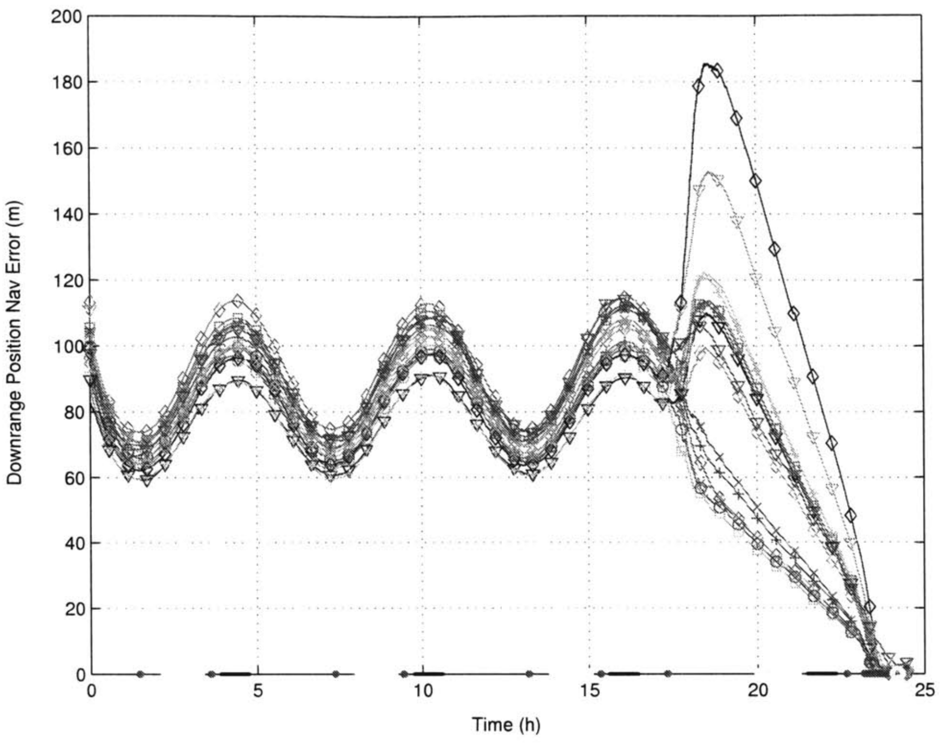


Figure 9-36: EX Comparison of downrange position navigation error, football orbit

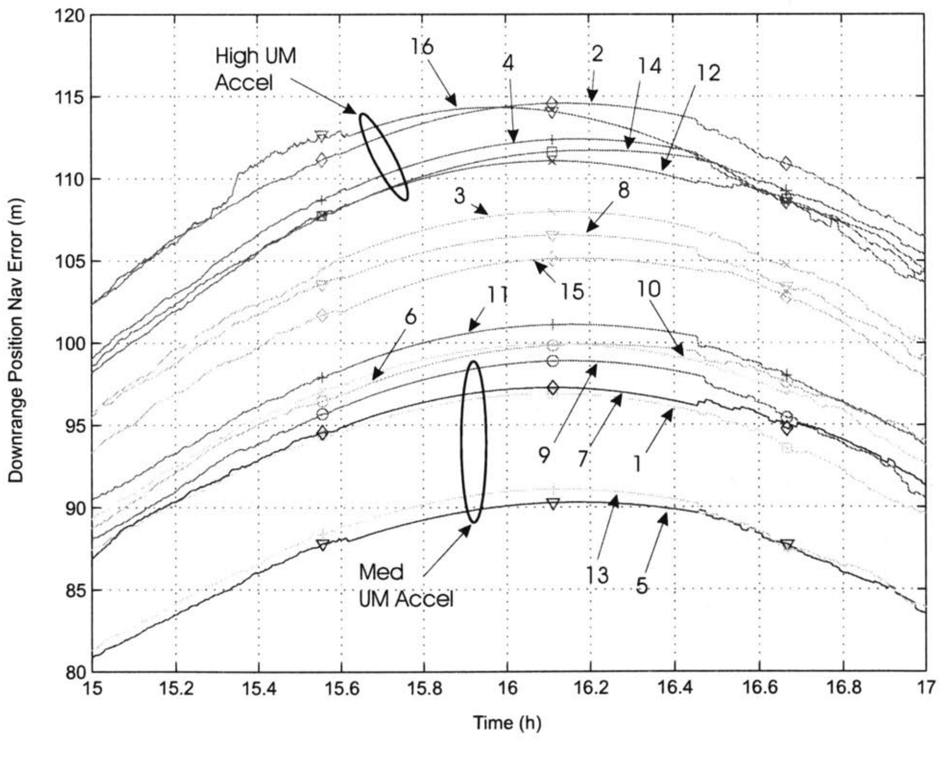


Figure 9-37: EX Comparison of downrange position navigation error, transfer burn

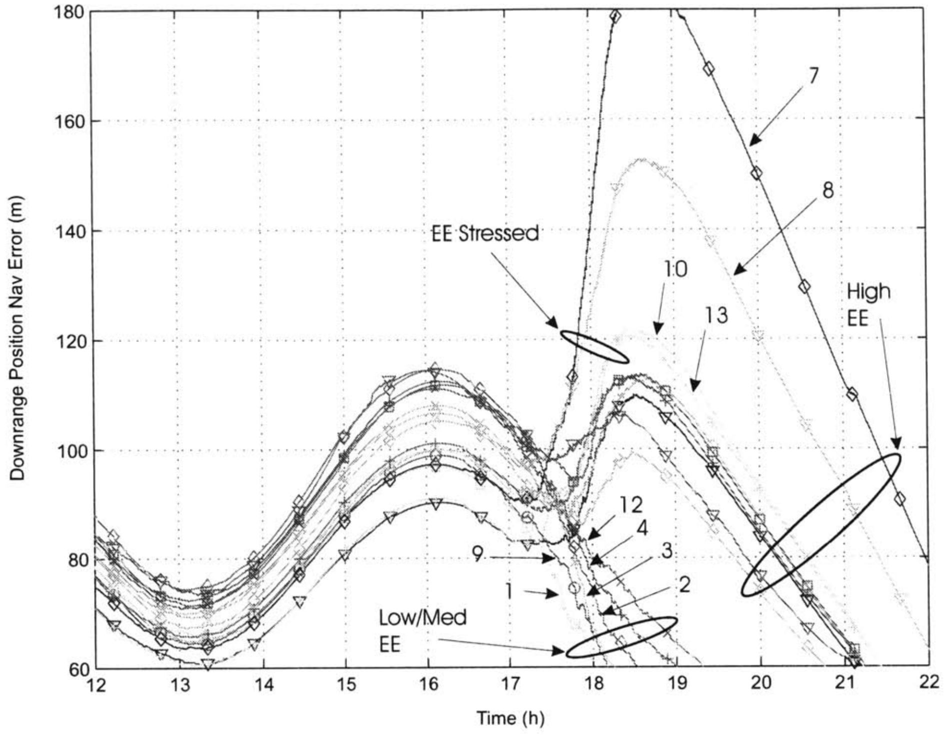


Figure 9-35 presents the downrange position navigation error for all 16 cases, synchronized at the initial time. The football correction maneuvers are commanded at specific absolute times, so it is reasonable to look at this view of the navigation error at least to the maneuver near 17.5 hours. In this figure many of the same trends are apparent that were discussed for the nominal case above. The navigation error in the downrange direction is very closely tied to the range, particularly in the holding football orbit. The cases subjected to increased amounts of unmodeled accelerations in the environment tend to have more navigation error that those that do not. This is shown more clearly in Figure 9-36, which is an enlargement of Figure 9-35 for a period on the football orbit. Trajectories experiencing high and low levels of unmodeled accelerations are indicated on the plot. Notice the period of smooth navigation error surrounded by areas of rather choppy lines. The smooth area occurs when the trajectories are propagating the navigation state through a period of eclipse.

Figure 9-38: EX Comparison of crosstrack position navigation error

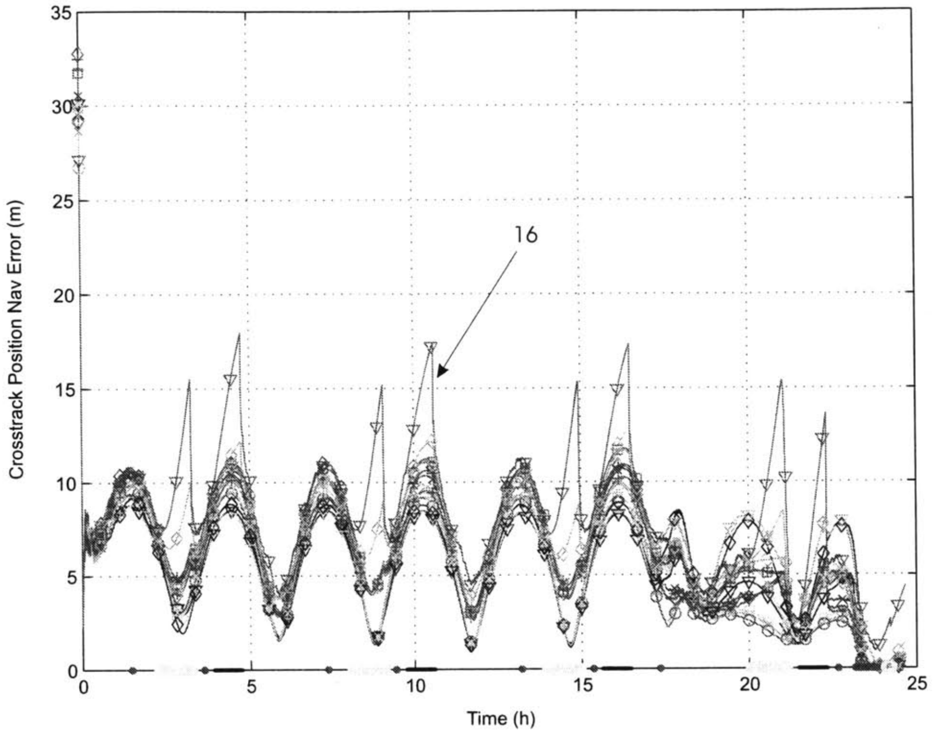


Figure 9-37 is an enlargement of the portion of the trajectory near the maneuver to enter the coelliptic transfer. The curves are labeled with their case number, and several interesting groupings are noted as well. Every case that had the high level of maneuver execution error in the environment shows a jump at the large maneuver to enter the coelliptic, regardless of whether or not the filter is properly tuned. The filters that are not properly tuned, however, have significantly larger navigation errors than the properly tuned filers. It is also worth mentioning that even those properly tuned cases that show a jump at this maneuver time still have a very similar level of error compared to the tuned filters in the Precomputed Gain implementation. Figures 9-38 and 9-39 show the crosstrack and altitude navigation error, respectively. These curves show similar trends to the downrange navigation error, however, note that case 16 experiences exaggerated spikes during constraints in both plots, due to the very high level of unmodeled acceleration it experiences from the environment.

Figure 9-40 is a plot of the downrange position navigation error for every case, where now the trajectories are synchronized at t_{zero}, the time of the final v-bar cross-

Figure 9-39: EX Comparison of altitude position navigation error

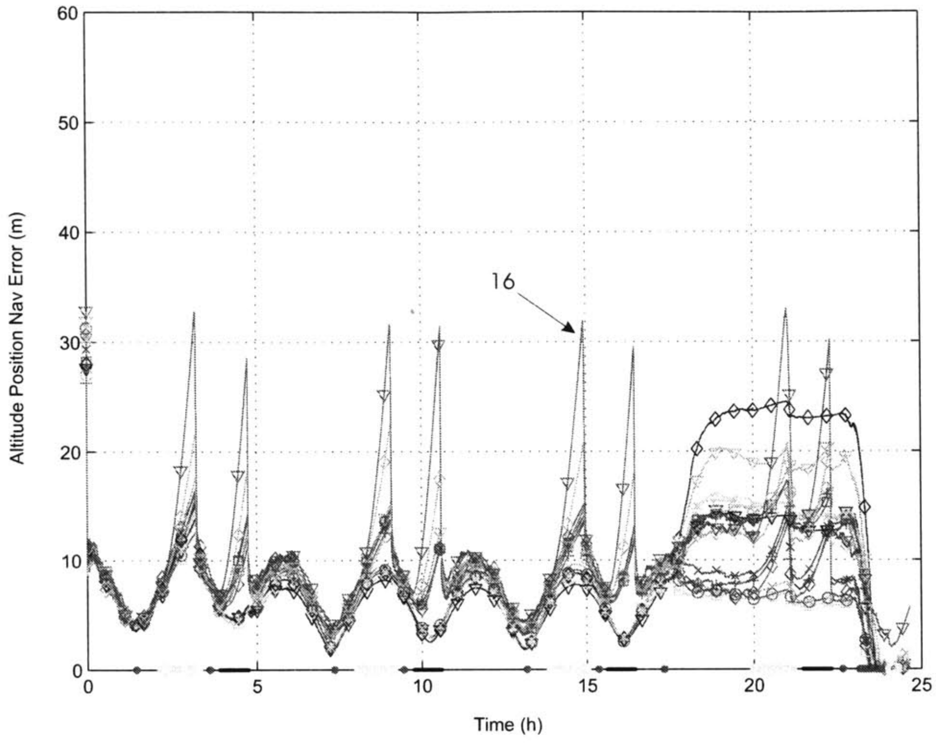


Figure 9-40: EX Comparison of capture correlated downrange navigation error

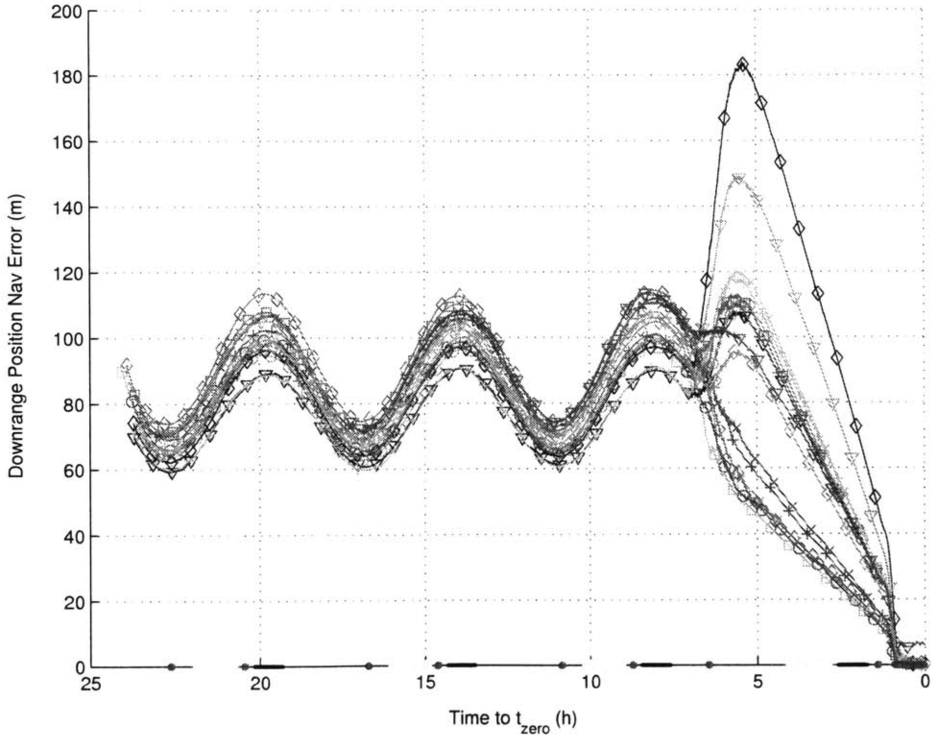
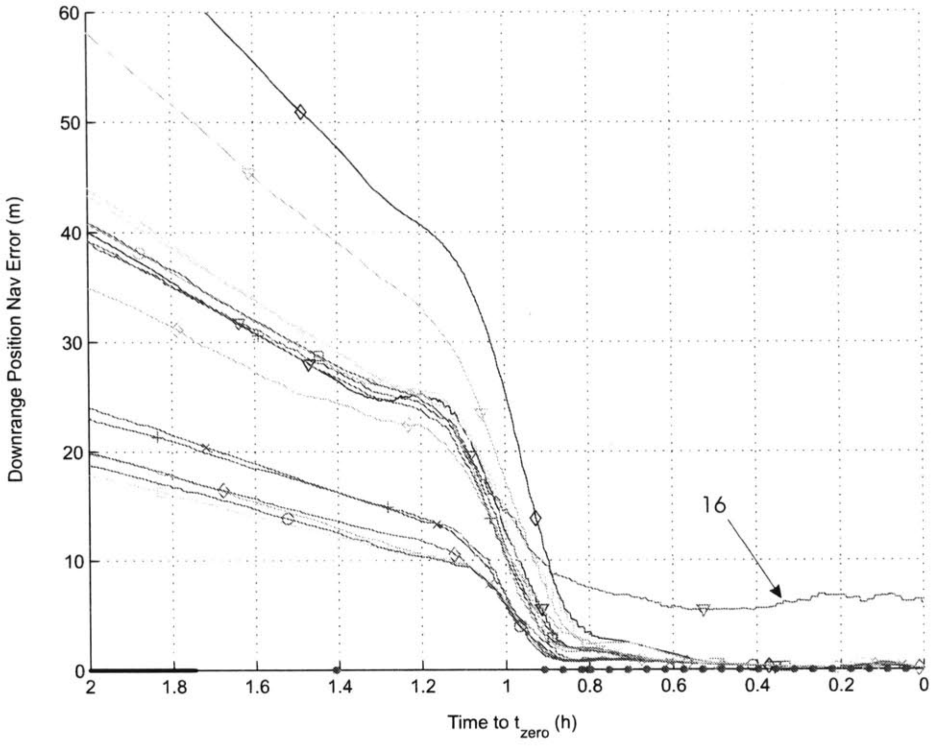


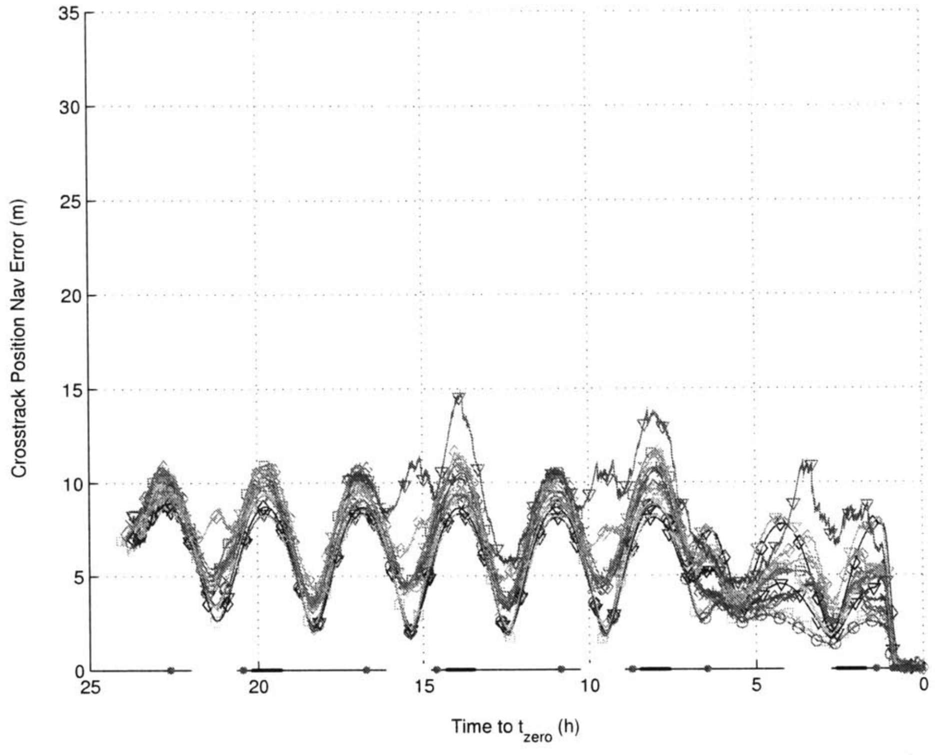
Figure 9-41: EX Comparison of capture correlated downrange GS navigation error



ing. Because the three large maneuvers in this trajectory (coelliptic insertion, fast transfer insertion, glide slope insertion) can occur at different times based on the geometry of the particular trajectory, the t_{zero} synchronized navigation error plots seem like the best way to evaluate the performance on the glide slope. Figure 9-41 is an enlarged view of the glide slope portion of this plot.

The first thing to notice about Figure 9-41 is the curve for case 16, marked with a triangle. While it is not a particularly large error, this case has a downrange error several times larger than the nearest case. The reason for this large downrange position error is actually a consequence of the simulation setup and plotting routine. Case 16 has 5 trajectories that require the use of the mosaic mode, two of these going relatively far astray before returning the the proper v-bar approach. While this is a tribute to the robustness of the targeting algorithms used in the Extended system, the deviations cost time and one of these trajectories did not finish before the simulation ended. This trajectory was left in the data because it is important and instructive

Figure 9-42: EX Comparison of capture correlated crosstrack navigation error

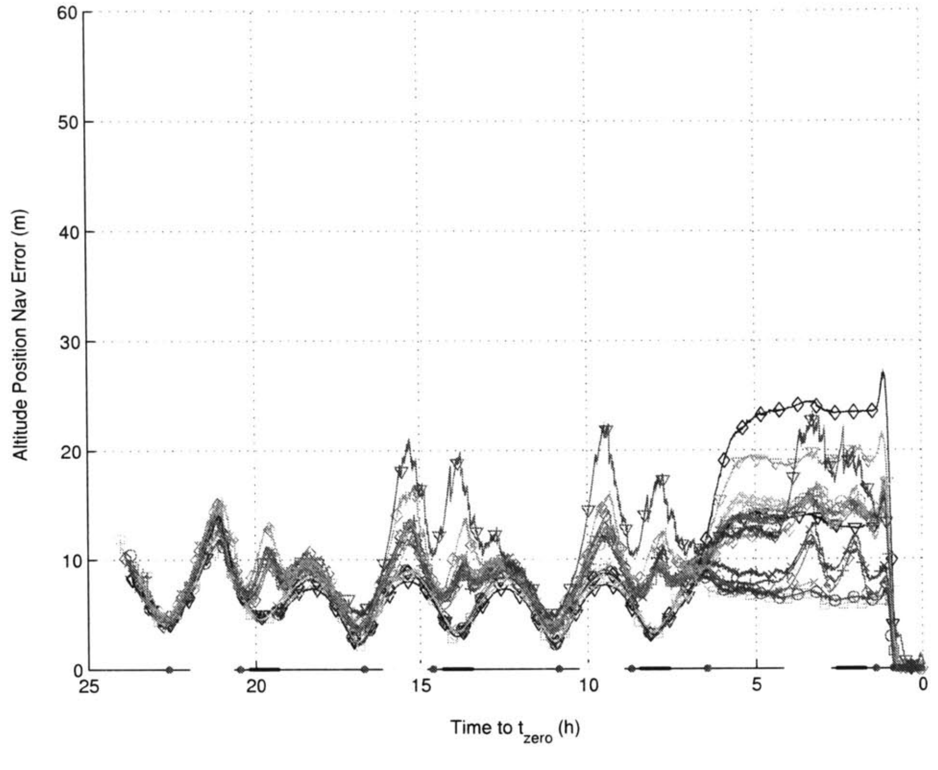


to see how all of the trials perform, and it had consequences on the statistics for the navigation error.

Figures 9-42 and 9-43 show the crosstrack and altitude navigation error, respectively. These plots show similar trends to the downrange position plot. Note that case 16 again shows large errors through the constraints, however those errors are muted because the trajectories are synchronized at the end of the run, misaligning the peaks of the errors on the coelliptic.

Figures 9-44 and 9-45 show the mean and 3-σ standard deviation of the target pointing error for the Extended system after constraints 7 and 8, respectively. The reader is referred to the plot of the reference trajectory shown in Figure 3-7 as a reminder of where the nominal constraints are located. Constraints 7 and 8 can be found near the end of the coelliptic approach, and have been shown in the nominal case of both the Precomputed Gain and Extended systems to be the worst of the 8

Figure 9-43: EX Comparison of capture correlated altitude navigation error

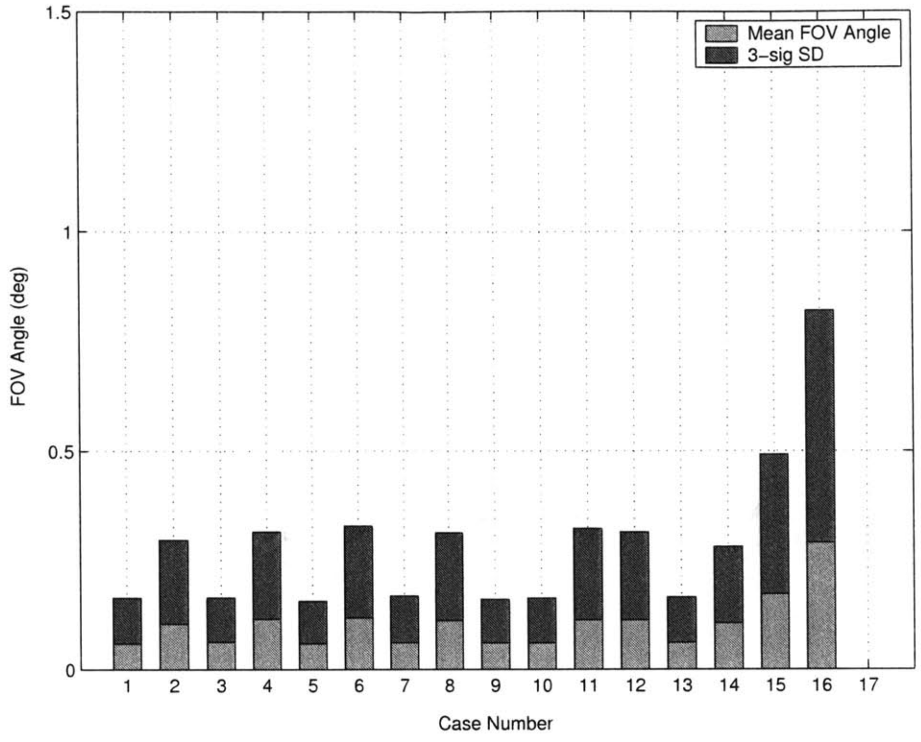


constraints for target pointing error.

The reader will immediately notice in Figure 9-44 that the pointing errors for the Extended system are significantly smaller than for the Precomputed Gain system. There is essentially no variation in the pointing error with changing maneuver execution error models, and a direct correlation with the amount of unmodeled acceleration. In cases 1 through 14, the cases with 100 m/day of unmodeled accelerations have about half the pointing error as those with 1000 m/day. Of the two cases that have larger pointing errors, case 15 has 2000 m/day of unmodeled accelerations while case 16 has 4000 m/day. The fact that the level of maneuver execution error does not effect the target pointing error compliments the result found for the navigation error.[9] In contrast to the Precomputed Gain system, the navigation error for the Extended system was shown to behave very well during the Sun angle and target shadowing constraints. Low navigation error translates into a good desired pointing solution,

[9]Indeed, the target pointing error can be seen as just another measure of the navigation error.

Figure 9-44: EX Comparison of target pointing error after constraint 7

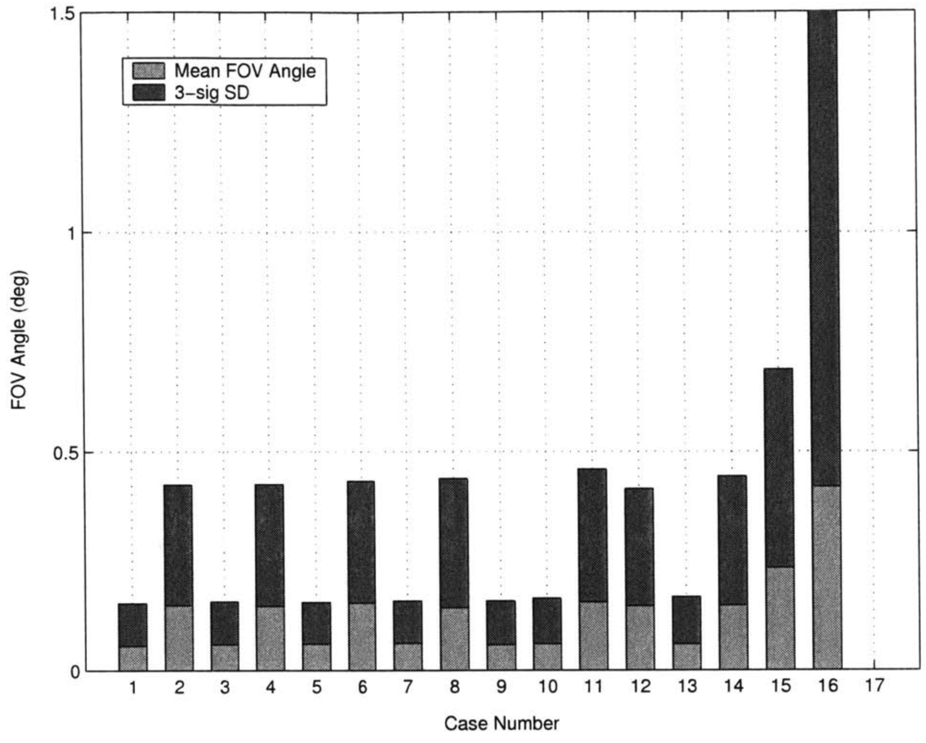


and hence a low pointing error.

The 3-σ pointing error in case 16 is larger than 0.7 degrees, the maximum allowable angle to still obtain a useful measurement. This means that there is a decent possibility of the chaser entering mosaic mode after constraint 7 in case 16, which does not bode well for constraint 8. Indeed, looking at Figure 9-45, case 16 sees a very good chance of a target pointing error greater than 0.7 degrees, and it was mentioned that 5 trajectories did in fact require the mosaic mode for this case. Fortunately, the Extended system with onboard targeting algorithms is robust enough that these mosaic trials were able to recover the proper trajectory and not diverge as in the Precomputed Gain system. The scale of Figure 9-45 was set to match Figure 9-18 for easy comparison. For reference, the 3-σ target pointing error for case 16 after constraint 8 is 2.4 degrees.

It can also be seen that after constraint 8 the trend from the previous plot is exacerbated. That is, those cases with large unmodeled acceleration levels now have

Figure 9-45: EX Comparison of target pointing error after constraint 8



larger pointing errors, while those cases with low levels remain the same. This suggests that the few measurements that are received between the two constraints are not sufficient, for the cases with high levels of unmodeled acceleration, to bring the navigation error back down to the level it was before constraint 7. Thus, the higher level of navigation error is propagated through constraint 8, resulting in larger pointing errors.

When discussing the nominal case, dispersion plots were introduced for the Extended system. As a reminder, these plots are generated by placing a mark on the figure at an epoch time for each trajectory. For the Extended system, the epoch times were chosen to be the actual maneuver times, which can vary from trajectory to trajectory. Figure 9-46 shows dispersion plots for the full trajectory for the first four cases with properly tuned flight computers, cases 1, 2, 3 and 4.

For the nominal case, the cross-shaped structure on the football orbit was explained to be a consequence of the small maneuver constraint. Now, in context with cases of other execution error models, the nature of this phenomenon can be seen more clearly. Cases 1 and 2 have the lowest amount of maneuver execution error, and also the "smallest" small maneuver constraint. Thus, based on the initial conditions of each trajectory, most trials require a maneuver at the first opportunity to match energy with the target. This maneuver is allowed by the small maneuver constraint, and not much execution error is applied. Thus, the dispersions at that point stay tight through all three circuits of the football orbit, and the football orbits do not have a tendency to drift towards the target (as indicated by the top-most dispersion ellipse, which is biased away from the target). The ellipses grow slightly larger for case 2, the upper-right plot, due to larger unmodeled accelerations "pushing" the vehicles around during the rendezvous. Cases 3 and 4 have larger execution errors and larger small maneuver constraints, resulting in larger dispersions, the cross-shaped structure on the football, and the possibility of football orbit drift towards the target.

Figure 9-47 presents another four full trajectories with dispersion ellipses at actual maneuver times. The four cases shown in this figure are for cases 13, 14, 15, and 16, which are improperly tuned cases based on the expected nominal parameters. Unmodeled accelerations in these cases are 100 m/day, 1000 m/day, 2000 m/day, and 4000 m/day, respectively. This increase can be clearly seen on the plot, where in case 16 the cross-shaped structure on the football orbit caused by the small maneuver constraint is completely overshadowed by the dispersions from the unmodeled accelerations.

Figure 9-48 shows four dispersion ellipse plots where the glide slope portion of the trajectory has been magnified. The cases chosen for this plot, cases 1, 3, 6, and 16, where chosen because they are each cases of particular interest. Case 1 is a tuned filter with the best possible conditions. The entire glide slope is contained in less than $\pm$ 0.5 meters of altitude. Case 3 is the nominal trajectory, and many of the rest of the cases have a very similar dispersion plot to the one shown for this case. Case 6, the lower left plot, is shown in contrast to the equivalent plot in the Precomputed Gain

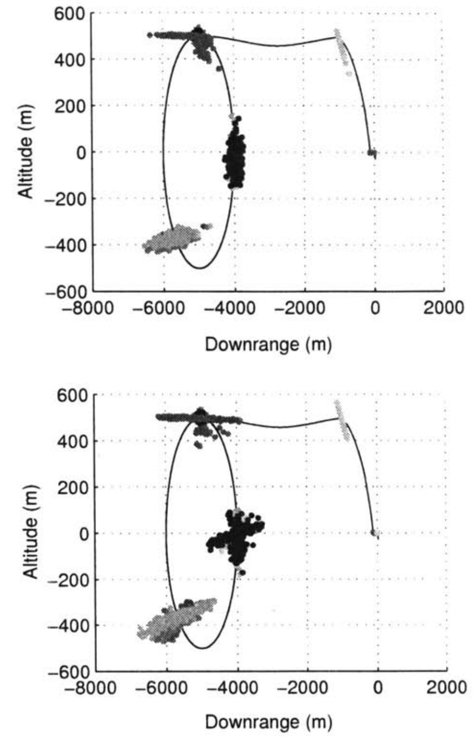


Figure 9-46: EX Comparison of position dispersions: Cases 1, 2, 3, 4

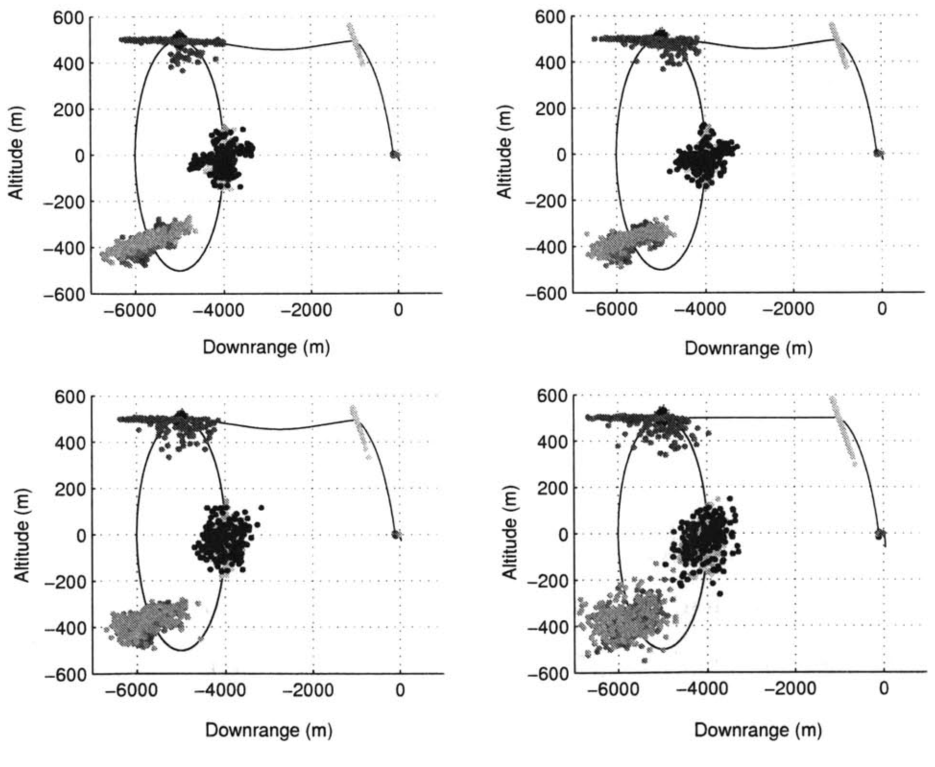


Figure 9-47: EX Comparison of position dispersions: Cases 13, 14, 15, 16

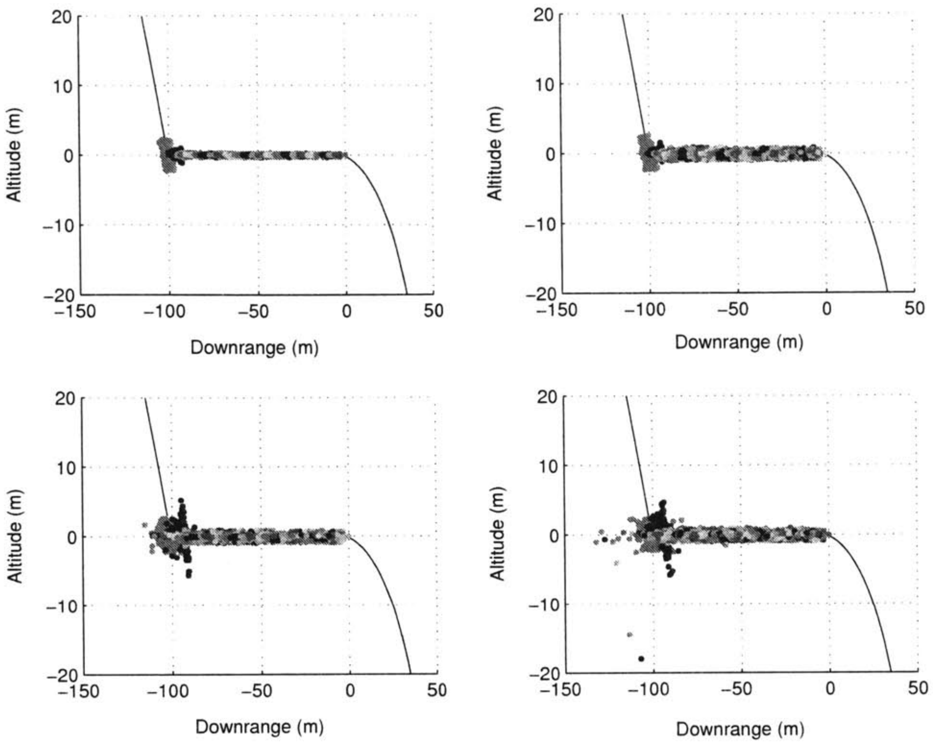


Figure 9-48: EX Comparison of position dispersions, glide slope: Cases 1, 3, 6, 16

system. Recall that in that system, one trajectory was divergent and its dispersions were dotted around the figure. Using the Extended system, this case is well contained, and no mosaic modes or divergent trajectories were encountered. Finally, case 16 is of interest because it *did* include several mosaic trajectories, which can be seen in the dispersions in the figure as errant dots behind and below the glide slope. The important note, however, is that the system was able to recover sufficient navigation accuracy to use the Lambert targeting to get back on track. This is the essential statement about the Extended system, and a suitable conclusion to this section.

9.3.3 Comparison

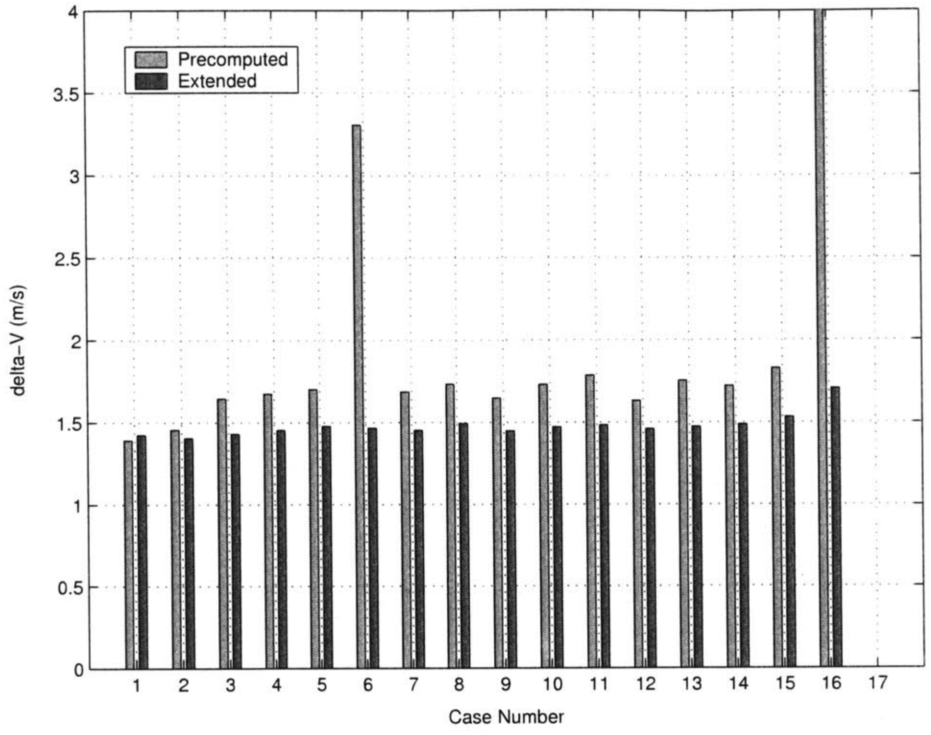
Sections 9.3.1 and 9.3.2 show the results of the analysis of the Precomputed Gain and Extended systems in great detail. Naturally, some amount of comparison was made between the systems as the data was presented, but this section will pull the analysis together for a concise comparison of these two algorithms as implemented.

Figure 9-49 is a plot of the total Δv required by each algorithm (mean plus 3-σ standard deviation), for every case studied in this work. This is the same data that was presented in the previous two sections, now combined into one figure for comparison. It is quickly seen that the Extended system outperforms the Precomputed Gain system in every case, except case 1 by a near margin. In cases 6 and 16, which partially define the performance envelope for the Precomputed Gain system, the Extended system clearly is a more favorable approach. For the other cases, however, the Δv saved is approximately 0.25 m/s, which may or may not be a level that is considered important to the mission design.

Figure 9-50 is a comparison plot of the target pointing error for the Precomputed Gain and Extended systems, for constraints 7 and 8. For each case, the left bar is constraint 7, and the right bar is constraint 8. Recall that these are the final two constraints in the trajectory, and occur near the end of the coelliptic approach. The data shown is the same as what was presented in the two sections above, with the pointing error for the Extended system plotted on top of the Precomputed Gain data. The height of each bar represents the mean pointing error plus the 3-σ standard deviation. The scale of this plot has been adjusted to show the most important details, and the final bars have been allowed to run off of the plot. Recall from the previous sections that, for constraint 8 in case 16, the Precomputed Gain and Extended systems have 3-σ target pointing errors of 2.2 and 2.4 degrees, respectively.

It is immediately clear from this plot that the Extended system is able to provide better pointing than the Precomputed Gain system when exiting these last two con-

Figure 9-49: Precomputed Gain vs. Extended: Required Δv



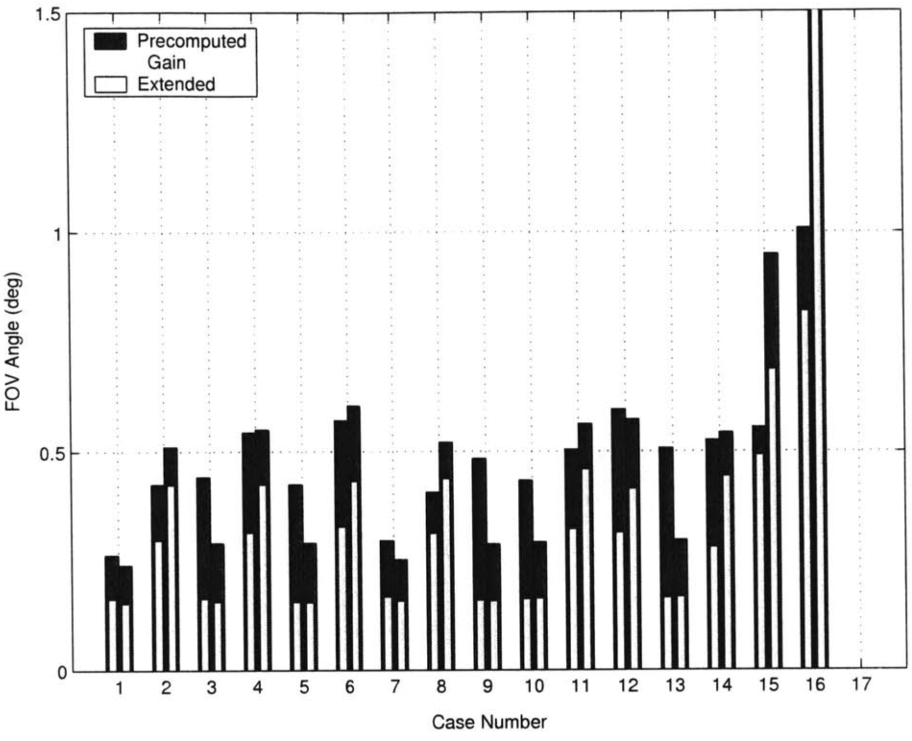
straints.[10] Both systems have difficulty with cases 15 and 16, indicating that there is a level of unmodeled acceleration that can cause any navigation system to have a large pointing error if not allowed to take measurements for extended periods.

Figures 9-51 through 9-53 give a comparison of the position navigation errors for the Extended and Precomputed Gain systems, broken down into the downrange, crosstrack, and altitude directions. In these plots, the light lines represent all 16 of the Precomputed Gain cases studied, and the dark lines represent all 16 of the Extended cases studied. For these plots it was necessary to choose a plot style for the Extended system, between trajectories synchronized at the initial time or the time of the final v-bar crossing. The author chose the former, as this data lines up better (in a purely graphical sense) and also correlates well for the most visible part of the figures.

It can be seen from the navigation error figures that, as a general trend, the Ex-

[10]It was shown in Sections 9.3.1 and 9.3.2 that the pointing error when leaving these two constraints is a worst-case picture of the pointing for the entire trajectory.

Figure 9-50: Precomputed Gain vs. Extended: Target pointing error



tended system provides better navigation than the Precomputed Gain system. In particular, the reader will of course notice the two cases that diverge in the Precomputed Gain system, while no such divergence appears in the Extended navigation. This reinforces the trend seen above in Figure 9-50, where one would expect a system with improved navigation to also have improved pointing.

It was mentioned in Section 2.2 that a discussion would be provided on the performance of these two algorithms in eccentric orbits. The simulation was reconfigured with a target orbit eccentricity of 0.04, and the chaser was initialized with the desired relative position and velocity (to initiate the holding football orbit at 5 km, even with the eccentric orbits), and perturbed initial conditions as before. In the Precomputed Gain system, this required creating a new reference trajectory with the proper eccentricity, and recomputing the required nominal maneuvers, maneuver gains, and measurement gains for the new trajectory. In the Extended system, targeting algorithms that were robust to non-zero eccentricity were sufficient.

Figure 9-51: Precomputed Gain vs. Extended: Downrange navigation error

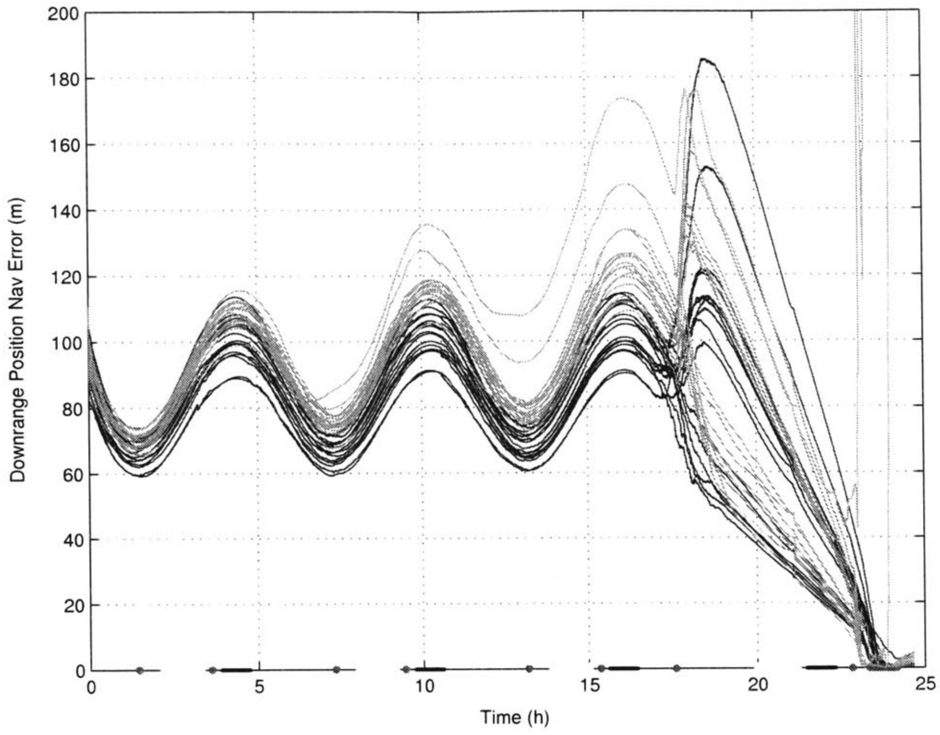


Figure 9-52: Precomputed Gain vs. Extended: Crosstrack navigation error

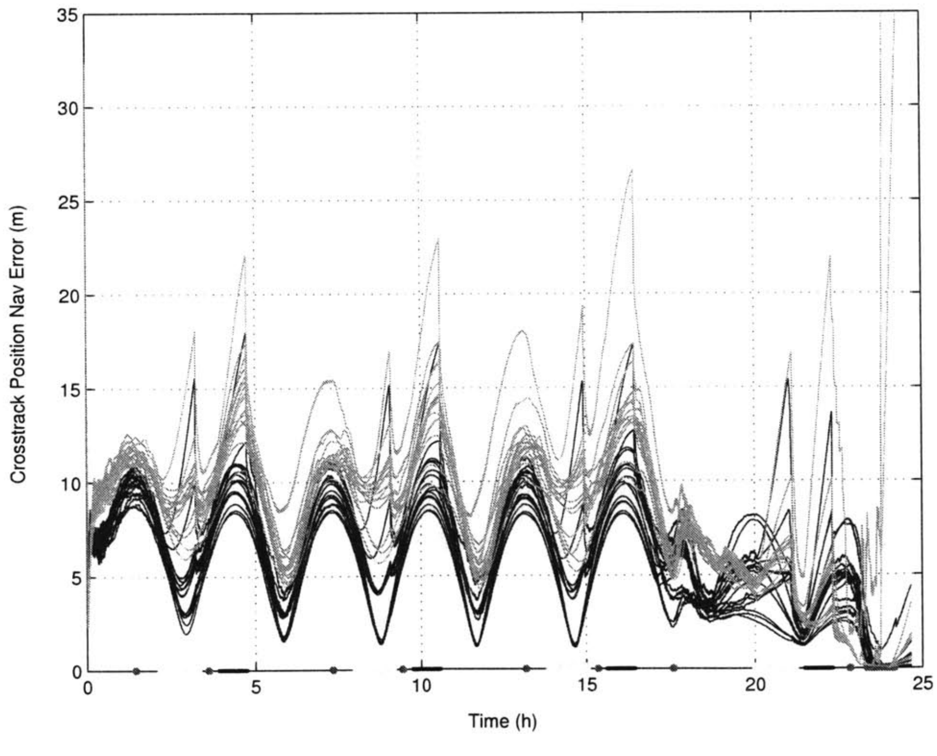
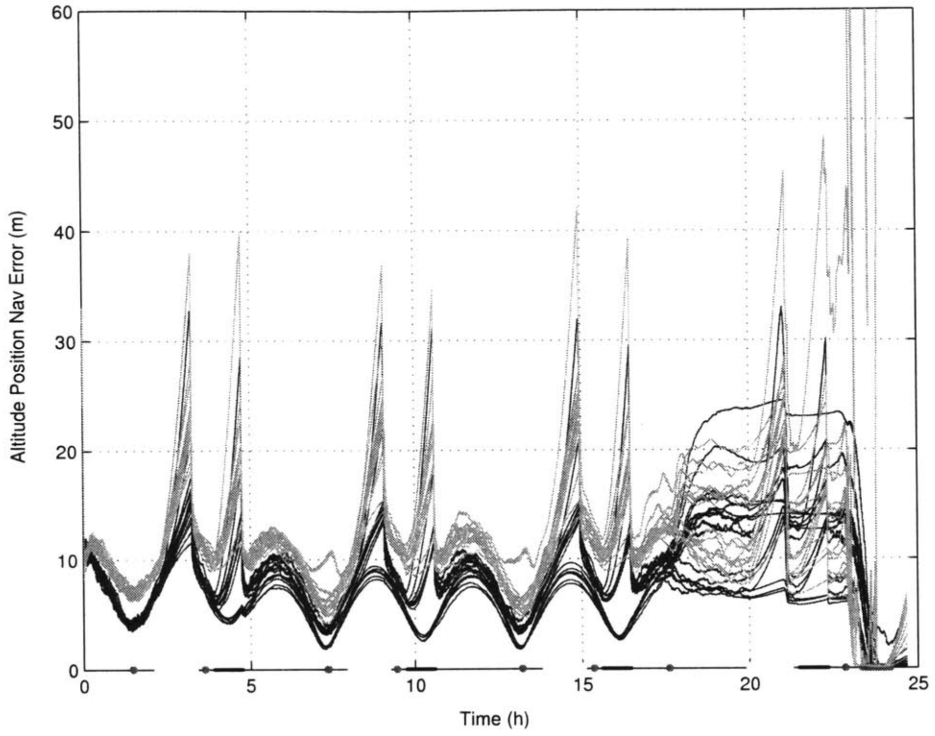


Figure 9-53: Precomputed Gain vs. Extended: Altitude navigation error



Two cases were run on both implementations, the nominal case and a stress case. Referring to Table 9.12, cases 3 and 14 were re-run with an eccentricity of 0.04. Interestingly, both systems performed very well under these conditions. In every case, both Precomputed Gain and Extended, there was a small increase in navigation and pointing errors, as well as Δv required, over the equivalent circular case. The Precomputed Gain system saw slightly larger increases than the Extended system. Still, the result of these simulations is that both systems were able to complete the rendezvous 200 out of 200 times, even for case 14 which is fairly rigorously stressed.

For the original circular version of case 14 using the Precomputed Gain system, the mosaic mode was never required. Now in the eccentric case, 3 trajectories required the mosaic mode, with one trajectory exhibiting slightly erratic behavior at the end of the fast transfer. This trajectory *was* able to recover sufficient navigation accuracy to return to the nominal trajectory before straying too far and diverging. It was perhaps expected that using the linearized Lambert targeting in the Precomputed Gain system, when the orbit was slightly eccentric, would cause problems for this algorithm, and cause some of the trajectories to stray from the nominal and diverge. This behavior was not seen in the simulation results, however, so the conclusion is that both of these systems are capable of performing the rendezvous for elliptical orbits and reasonable environmental inputs. It is expected that, if an elliptic version of Table 9.12 was populated for the Precomputed Gain system, the performance envelope would be slightly smaller than for near circular orbits.

The reader is sure to ask at this point: so which system is *better*? The answer, quantitatively, is that the Extended system is. It has a larger performance envelope, flexible targeting algorithms that help prevent errant cases, lower statistical Δv requirements, better navigation, and better pointing when exiting constraints. Of course, this is not the end of the story. The author would have the reader also ask: how *much* better is it? As shown in the plots earlier in this section, the improvements are perhaps not of critical importance to designers. A difference of 0.25m/s of Δv; an expanded performance envelope, but expanded outside of the most conservative bounds; and, navigation and pointing improvements, but over values that were already well within reason. The following 3 chapters will discuss some less quantitative but still important considerations that might aid in a decision between these two unique algorithms.

[This page intentionally left nearly blank.]

Chapter 10

Technology Readiness Level

10.1 Description and Definitions

An important consideration when evaluating the merits of the navigation and targeting algorithms under study, is the level to which they have been developed, tested, and used in actual space systems. A relevant metric for this type of consideration is the *Technology Readiness Level* (TRL) of each algorithm. A straightforward definition of TRLs is given by John Mankins in a paper for the NASA Office of Space Access and Technology [9]:

> Technology Readiness Levels (TRLs) are a systematic metric/measurement system that supports assessments of the maturity of a particular technology and the consistent comparison of maturity between different types of technology.

In this case, a comparison "between different types of technology" is not required, however a systematic metric to asses algorithm maturity is precisely what is desired. The current standard definitions for the various TRLs can be found in a Department of Defense document on mandatory procedures for Major Defense Acquisition Programs (MDAPS) [12]. These definitions are worded in a necessarily general way, to be applicable to any type of technology that might be procured. TRL definitions that are more relevant to this study were found in a NASA document applying the nine levels of technology readiness to software systems [1]. These levels are listed below and form the basis of the comparison at hand. The following sections examine each algorithm to determine its TRL, and a comparison is made between their respective "readiness levels."

TRL 9 Actual system is "mission proven" through successful mission operations. Thoroughly debugged software readily repeatable. Fully integrated with operational hardware/software systems. All documentation completed. Successful operational experience. Sustaining software engineering support in place. Actual system fully demonstrated.

TRL 8 Actual system completed and "mission qualified" through test and demonstration in an operational environment. Thoroughly debugged software. Fully integrated with operational hardware and software systems. Most user documentation, training documentation, and maintenance documentation completed. All functionality tested in simulated and operational scenarios. V&V completed.

TRL 7 System prototype demonstration in high-fidelity environment (parallel or shadow mode operation). Most functionality available for demonstration and test. Well integrated with operational hardware/software systems. Most software bugs removed. Limited documentation available.

TRL 6 System/subsystem prototype demonstration in a relevant end-to-end environment. Prototype implementations on full scale realistic problems. Partially integrated with existing hardware/software systems. Limited documentation available. Engineering feasibility fully demonstrated.

TRL 5 Module and/or subsystem validation in relevant environment. Prototype implementations conform to target environment/interfaces. Experiments with realistic problems. Simulated interfaces to existing systems.

TRL 4 Module and/or subsystem validation in laboratory environment. Stand-alone prototype implementations. Experiments with full scale problems or data sets.

TRL 3 Analytical and experimental critical function and/or characteristic proof-of-concept. Limited functionality implementations. Experiments with small representative data sets. Scientific feasibility fully demonstrated.

TRL 2 Technology concept and/or application formulated. Basic principles coded. Experiments with synthetic data. Mostly applied research.

TRL 1 Basic principles observed and reported. Basic properties of algorithms, representations and concepts. Mathematical formulations. Mix of basic and applied research.

10.2 Results

10.2.1 Precomputed Gain

Even with the concise definition of the TRLs given above, placing the Precomputed Gain algorithm on the scale is still not easy. Certainly, scientific feasibility has been fully demonstrated for this system. Work by Geoffrey Huntington [5], the Optical Navigation group at JPL, and the author has all demonstrated that this system is in fact a viable option for autonomous rendezvous. Further, the deterministic simulations used in this analysis and at JPL have validated the Precomputed Gain filter and targeting algorithms in simulation, using "full scale problems" as required by TRL 4.

Technology readiness level 5, according the the ARC/GSFG document, requires "model validation in a relevant environment." Additionally, it requires an implementation that conforms to the actual environment, with simulated interfaces, and experiments with realistic problems. The Monte-Carlo simulator used in this analysis would seem to meet all of these requirement, depending on the reader's interpretation of the "relevant environment" and "realistic problems."[1] It is the author's view that, while the Precomputed Gain Kalman filter and targeting algorithms meet the require-

[1]That is to say, the Precomputed Gain system has not been tested in a realistic physical environment, but it *has* been tested in a reasonably high-fidelity simulated environment.

ments for TRL 4, there might not be full justification to move to TRL 5. Thus, the Precomputed Gain algorithms are assigned a technology readiness level somewhere between TRL 4 and TRL 5, with strong emphasis on the latter.

10.2.2 Extended Kalman

The Extended Kalman filter and onboard targeting algorithms, generally speaking, have significantly more heritage than the Precomputed Gain system. The Space Shuttle uses a version of the Extended Kalman filter for navigation during rendezvous, and the Deep Space 1 probe also used an Extended filter for rendezvous operations. Thus, the general Extended Kalman filter would receive a technology readiness rating of TRL 8 or TRL 9 – mission qualified and mission proven.

The implementation of the Extended Kalman filter and onboard targeting algorithms used in this study, however, is unlike any that has ever flown before. Speaking specifically about the 20 state Extended algorithm, used for relative navigation in autonomous rendezvous operations, no such system has been implemented on hardware. In this respect, the Extended algorithm used in this study would receive a technology readiness rating of TRL 4 or TRL 5, suggesting that it has been validated on full scale problems and could perhaps shown validated in a "relevant environment." Without hardware implementation, TRL 6 cannot be achieved.

10.2.3 Comparison

The preceding sections rate both the Precomputed Gain and Extended implementations used in this study as TRL 5 or a high TRL 4. These rankings on the scale of technology readiness represent a significant amount of effort and expense that would be required to space qualify either algorithm for use in a real mission. Both systems have been demonstrated to have scientific feasibility, have been validated with full scale problems, and are implemented in a way that conforms to the expected environment. Neither has been implemented on hardware. On the other hand, it must be noted that the general Extended Kalman filter does have a space-qualified heritage in space. It would be expected that the time and cost required to prepare this particular implementation for an actual mission would be lower than for the Precomputed Gain system. Thus, while the actual ratings might suggest that the two algorithms are on equal footing for technology readiness, the conclusion of this chapter is that the Extended system is significantly more developed than the Precomputed Gain system.

[This page intentionally left nearly blank.]

Chapter 11

Complexity and Computational Burden

The Precomputed Gain and Extended algorithms exhibit large differences in the amount of software and computation that is required onboard the flight computer. The Precomputed Gain algorithm was specifically designed to have a simple implementation where the most complicated operation is matrix multiplication.[1] Thus, it is expected that the Precomputed Gain system will demonstrate significant advantages over the Extended system in terms of the characteristics studied in this chapter.

Section 2.2.3 introduced the concepts of **complexity** and **computational burden**. Increased complexity in onboard programming has consequences in terms of software bugs, computer memory and storage resources, and development time and cost. Increased computational burden, of course, cuts into processing time that could be used for science or other spacecraft functions, and possibly drives the selection of more powerful hardware. Unfortunately, it is difficult or impossible to quantify these ideas in a general way for these two algorithms. Instead, the author has used an approach that focuses specifically on the implementations created for this thesis. This approach will be immediately relevant to the MTO rendezvous demonstration and MSR mission, and should also be of use to designers considering either of these

[1]This is not entirely true. There are a few additional calculations required, including a square root in the calculation of the range residual. The bulk of the required processing, however, is involved with matrix multiplication

algorithms for future use. The analysis method will be described in Section 11.1, and then results will be given for the Precomputed Gain and Extended algorithms in the sections that follow.

11.1 Description and Definitions

As defined above and in Section 2.2.3, the *complexity* of an algorithm is directly related to the amount of programming required for the implementation. Thus, a straightforward way to evaluate the complexity of two existing implementations is to simply measure the amount programming required for each. In this analysis, the metric for measuring the 'amount of programming' is the number of lines of code required to create each simulated flight computer. Since all of the software has been written by the author and colleagues[2] in a reasonably consistent style, a relative comparison between the Precomputed Gain and Extended implementations is very reasonable using this metric.

Both the Precomputed Gain and Extended implementations have been created in MathWorks' MATLAB/Simulink [6]. Simulink proved very useful in implementing the environment model described in Chapter 5 (largely due to the ease with which numerical integration can be performed), and quickly mating it with the desired "flight" algorithm. Fortunately, the flight algorithms themselves are largely coded in MATLAB and interfaced to Simulink through function call blocks. Thus, "lines of code" for the flight algorithms can be defined as lines of MATLAB code, and any operations that are performed in Simulink have been re-coded in MATLAB to establish their equivalence. Likewise, some functions that were implemented as Simulink "S-functions" (compiled C code) were re-coded in MATLAB to establish their equivalence. Only code required by the filter and targeting algorithms was considered in this analysis. For example, pieces of code used to generate output in the LVLH coordinate system for plotting were not counted in the total. The breakdown for

[2]Special thanks to David K. Geller, who has provided many tools and insights that form the backbone of the software implementations.

the Precomputed Gain and Extended implementations can be found in the following sections.

The idea of *computational burden* is a way to think of the demand that an algorithm places on the computational resources available to it. There are many ways to quantify computational burden, including: required operations for each iteration; processing time required per iteration; or, percentage of available processing resources consumed. For this thesis, the easiest way to measure computational burden is in terms of the required processing time to complete each iteration. Since both algorithms are implemented in MATLAB/Simulink, it is relatively straightforward to use the internal computer clock to time the duration of each Monte-Carlo trial. Of course, it is necessary to implement some controls to make the best possible estimate, and these were instituted as follows.

The hybrid MATLAB/Simulink nature of the simulation makes it difficult to separate the environment model from the flight computer models, but the execution speed of the environment model is not in question here. Thus, a dummy flight computer model was created that contained no calculations and returned mock pointing and maneuver data to the environment model. The full simulation was then run for an extended "Monte-Carlo" trial to determine the processing time required by the environment model, for the baseline trajectory under study. With the calculation time required for the environment model now known, the Precomputed Gain and Extended simulations were run with identical parameters[3] on the same computer, and the mean trial execution time calculated for each. With the environmental bias removed, the total flight computer processing time can be divided by the total number of time steps to determine the processing time required for each iteration of each filter. The results from this analysis are shown in the sections to follow.

[3]Case 1 parameters were used.

MATLAB	Lines
Propagate navigation states	27
Update navigation states	17
Compute required $\Delta \boldsymbol{v}$	14
Simulink	
Check measurement validity	7
Boresight Pointing	
Calculate $\boldsymbol{T}_{I \rightarrow c}$	11
Matrix to quaternion	5
Overhead estimate	10
Total	**91**

Table 11.1: Lines of code required by the Precomputed Gain implementation.

11.2 Results

11.2.1 Precomputed Gain

The number of lines of code for the Precomputed Gain system break down according to Table 11.1. Again, the programming of interest here is only the "flight computer" portion of the implementation. A substantial amount of code is required to generate the precomputed data that must be uploaded to the orbiter, but these calculations can be made and checked on the ground by very powerful computers, with operators in the loop. A small amount of 'overhead' has been added to the Simulink portion of the analysis, representing features provided by the Simulink interface such as unit delays and block interfaces.

Remarkable as it seems, a full autonomous rendezvous can be performed using precomputed data and less than 100 lines of code! It should be remembered that this is MATLAB code, making full use of built-in functions such cross products and matrix multiplication, and implementation in a lower level language such as C would be substantially longer. It is the author's view that the number of lines of code would scale linearly between the two algorithms under study, if the code for the built-in functions were considered, or the algorithms were coded in another language.

MATLAB	Lines
Navigation filter main function	78
Propagate navigation states	129
Update navigation states	85
Targeting algorithms	101
Simulink	
Check measurement validity	7
Boresight Pointing	
Calculate $\boldsymbol{T}_{I\to c}$	11
Matrix to quaternion	5
Overhead estimate	10
Total	**426**

Table 11.2: Lines of code required by the Extended implementation.

For the Precomputed Gain Kalman filter and targeting algorithms implemented on the computer used for this analysis, the mean processing time required for each trial is 20.9 seconds. Removing the bias of 4.5 seconds calculated for the environment model, a total of 16.4 seconds are required for the flight computer in each Monte-Carlo trial. Since each trial is 88800 seconds long, or 8881 time steps, the processing time is found to be 1.85 ms per time step for the Precomputed Gain system, on the author's 1.5 GHz Intel Pentium 4 based computer.

11.2.2 Extended Kalman

The lines of code in the Extended system break down according to Table 11.2. It is assumed here that the initial covariance matrix is uploaded to the spacecraft (see Chapter 12) rather than initialized onboard. Onboard initialization would require the addition of code implementing the algorithm of Section 6.3, and would further increase the disparity between the Precomputed Gain and Extended algorithms. Again a small amount of 'overhead' has been added to the Simulink portion of the analysis, representing features provided by the Simulink interface such as unit delays and block interfaces.

It is no surprise that the Extended system requires substantially more code than that Precomputed Gain system, as the entire Kalman filter and targeting algorithms are carried onboard. While using the full algorithm improves other aspects of the system's performance, it certainly has a cost in terms of complexity. A comparison between the Precomputed Gain and Extended implementation follows in Section 11.2.3.

For the Extended Kalman filter and onboard targeting algorithms implemented on the computer used for this analysis, the mean processing time required for each trial is 70.2 seconds. Removing the bias of 4.5 seconds calculated for the environment model, a total of 65.7 seconds are required for the flight computer in each Monte-Carlo trial. Since each trial has been run for 88800 seconds, or 8881 time steps, the processing time is found to be 7.40 ms per time step for the Extended system.

11.2.3 Comparison

Tables 11.1 and 11.2 give the breakdown of the code required for the Precomputed Gain and Extended implementations used in this study, respectively. The Precomputed Gain system uses approximately 91 lines of code, while the Extended system uses approximately 426 lines. It hardly bears further telling that the Extended implementation is significantly more 'complex,' by the definition given in the description above. While the numbers presented are based on a particular coding by a particular author, it is nonetheless clear that the Extended system requires several times more code than the Precomputed Gain system for a complete implementation. The Extended system will thus be more prone to software bugs and will take longer to develop and test, at greater expense.

Some might argue that the number of lines of code do not tell the whole story for the complexity of the Precomputed Gain algorithm. In full application this algorithm would require protocols for the generation and transmission of large quantities of precomputed data. It would mate a distant and largely static flight computer with ground-based, easily updated tools for the generation of the required data. The Precomputed Gain algorithm is *split in half*, and the thoughtful designer has reason to wonder if the amount of code in the flight computer is the only measure of complexity for this system. The author proposes, however, that all of these challenges are things that can be easily overcome; hard problems on the ground are still easier than small problems on a distant orbiter.

The sections above describe a time step processing requirement of 1.85 ms for the Precomputed Gain system, and 7.40 ms for the Extended system. Since both algorithms were run for the same number of time steps, the time to run one trial is an equivalent metric in larger units. The Precomputed Gain system requires 16.4 seconds to complete one trial, while the Extended system requires 65.7 seconds. Thus, for these particular implementations, the Extended system requires almost exactly four times as much processing time as the alternative. This is certainly consistent with the complexity analysis above, where the Extended system was likewise found to require about four times as much code.

Once again it should be noted that the numerical values presented for computational intensity were calculated for a particular trajectory, generated on a particular personal computer, by executing code created by the author. As a comparative analysis, however, these values represent a reasonably consistent trial, and could be applied as a ratio to future designs under consideration. The conclusion of this section is, therefore, that the Precomputed Gain system is much less complex than the Extended system and requires significantly less onboard processing power. For a mission calling for the simplest onboard system possible, it is ideally suited in this regard. The Extended system, while showing strengths in other areas, should be considered relatively complex and computationally intensive between these alternatives.

[This page intentionally left nearly blank.]

Chapter 12

Data Uplink Requirements

Chapter 2 introduced the notion of a large gap in the data uplink requirements for the two sets of algorithms under study. By its nature, the Precomputed Gain algorithm calls for a large upload of data required by the flight computer, while the Extended Kalman filter and associated targeting algorithms require only basic configuration parameters to perform the rendezvous. The effect that these large data files will have on the orbiter's communication budget and onboard storage are important things for a designer to consider.

The amount of data required by the Precomputed Gain system of course depends on the particular rendezvous mission, including the rendezvous trajectory, number of sensors, maneuver frequency, and measurement frequency.[1] This chapter will deal specifically with the mission outlined in this thesis. Specifically, the analysis looks at the baseline trajectory described in Chapter 3, combined with the camera model referenced frequently throughout this work and the measurement frequency described by Table 9.3. It will be assumed that data requirements for general spacecraft function, such as telemetry and tracking data, will be common between the two implementations, and so will not be considered.

Section 12.1 will briefly describe the method by which data requirements were calculated for the Precomputed Gain and Extended implementations. Sections 12.2.1

[1]Algorithm configuration can also have a significant impact on the data requirements. For example, the integration time step used in this analysis is 10 seconds. A 5-second time step would nearly double the data requirement.

and 12.2.2 present the results for the Precomputed Gain and Extended algorithms, respectively. Finally, Section 12.2.3 will provide a comparison between the two algorithms.

12.1 Description and Definitions

As mentioned above, data requirements have been calculated specifically for the MTO mission baseline trajectory discussed extensively in this thesis. The data file size is highly dependent on the reference trajectory, measurement frequency, and integration time step, but from the example given here a designer could generate a reasonable estimate for his or her particular mission.

In the MATLAB/Simulink implementations of the two navigation filters and associated targeting algorithms, all variables (and vector and matrix components, etc) are represented by 64-bit double precision numbers. Thus, a suitable system for calculating minimum data requirements for the two implementations is to simply find the total number of values required by the flight computer, and multiply this by 8 to find the bytes of storage required for the raw data. This will be the amount of storage required to transmit the exact data used in this analysis, to the exact same level of precision.[2] The details of file structure, etc, will be ignored in this analysis since they are the same for both implementations. The result of this analysis for both algorithms is shown in the following two sections.

12.2 Results

12.2.1 Precomputed Gain

The data requirements for one rendezvous mission using the Precomputed Gain system are shown in Table 12.1. The recurring value of 8881 indicates the number if time steps in the simulation (88800 seconds with a time step, Δt, of 10 seconds,

[2]Lower precision values could certainly be used to save space, but this would have consequences for the numerical accuracy of the algorithms and has not been tested here.

Required Variables	Dimensions	Doubles
Reference trajectory	$[12 \times 8881]$	106572
State transition matrices	$[12 \times 8881]$	
Target	$[6 \times 6 \times 8880]$	319680
Chaser	$[6 \times 6 \times 8880]$	319680
Measurement (Kalman) gain matrices	$[27 \times 3 \times 1060]$	85860
Maneuver gain ($\boldsymbol{C}^*$) matrices	$[29 \times 3 \times 12]$	1044
Nominal maneuvers	$[29 \times 3]$	87
Maneuver times	$[1 \times 29]$	29
Measurement times	$[1 \times 1060]$	1060
Initial state	$[12 \times 1]$	12
Minimum maneuver size (SMC)	$[1 \times 1]$	1
Total		**834025**
Bytes	Total $\times$ 2	**6672200**
Megabytes	Bytes/1024^2	**6.36**

Table 12.1: Data requirements for the Precomputed Gain implementation.

including time $t_0 = 0$ seconds), and the value 1060 indicates the total number of measurements taken. Detailed descriptions of the precomputed data can be found in Section 8.1.2. The table indicates that a total of approximately 834,000 double precision numbers must be transmitted to the rendezvous orbiter, for a total of approximately 6.36 megabytes. For the MTO demonstration, the orbiter is of course a communication satellite, where data transfer is not likely to be a problem. For the MSR mission or other future missions, however, the large data requirement is certainly a consideration.

It should be noted that the 6.36 megabytes listed in the bottom of the table is not necessarily the final word in the the data that must be transferred. Very simple data compression could reduce this file size dramatically. More sophisticated algorithms could be created that make use of the properties of the data itself, such as fitting the vehicle dynamics to higher order polynomials or reducing possibly sparse gain matrices. This analysis assumes the simplest possible flight computer with no decompression routines available, so the value from Table 12.1 stands as shown.

12.2.2 Extended Kalman

The data requirements for one rendezvous mission using the Extended system are shown in Table 12.2. Unlike the Precomputed Gain system, which was designed to use precomputed data, the Extended system propagates state dynamics and computes measurement gains onboard. The variables required for this system are initialization values and terms to characterize the expected sources of error during the mission (such as execution error, unmodeled accelerations, and measurement angle error). As was mentioned in Section 11.2.2, it is assumed that the initial covariance matrix is transmitted to the orbiter rather than initialized onboard. The approximate total number of parameters required by the Extended system is only 484 (including 400 from the covariance initialization), resulting in a total storage requirement of approximately 3,872 bytes. This value will be discussed in comparison to the Precomputed Gain total in the following section.

12.2.3 Comparison

The total raw data requirements for the Precomputed Gain and Extended implementations are shown in Tables 12.1 and 12.2 above. The Precomputed Gain system was found to require approximately 6.36 *megabytes* of data, while the Extended system only required approximately 3872 *bytes*. The difference here is more than three orders of magnitude, not a small consideration by any means. As mentioned above, certain levels of compression might reduce the Precomputed Gain number dramatically, but the conclusion will still be clear: a designer must consider a very large increase in data transmission requirements when using this system.

Additionally, recall the design of the Precomputed Gain system, where state transition matrices, Kalman gain matrices, maneuver gain matrices, and nominal maneuvers are uploaded to the orbiter in inertial coordinates *for every unique mission.* If the rendezvous mission epoch time is missed (even by a few minutes), or a nominal maneuver is changed, or the level of expected unmodeled accelerations changes, for example, the entire data set must be uploaded again. Contrast this with the Extended

Required Variables	Dimensions	Doubles
Initial covariance	$[20 \times 20]$	400
Initial state	$[20 \times 1]$	20
Measurement ranges	$[2 \times 1]$	2
Measurement Δt at ranges	$[2 \times 1]$	2
Markov state time constants	$[8 \times 1]$	8
Markov state standard deviations	$[8 \times 1]$	8
Initial time (t_0)	$[1 \times 1]$	1
Integration time step (Δt)	$[1 \times 1]$	1
K_{OS}	$[1 \times 1]$	1
K_{Orb}	$[1 \times 1]$	1
Measurement angle variance	$[2 \times 1]$	2
Execution error model parameters	$[3 \times 1]$	3
Minimum maneuver size (SMC)	$[1 \times 1]$	1
Targeting parameters	$[34 \times 1]$	34
Total		**484**
Bytes	Total $\times$ 2	**3872**
Megabytes	Bytes/1024^2	**0.0**

Table 12.2: Data requirements for the Extended implementation.

system, where the new parameter (or an entire new data set) can be uploaded with very low data uplink requirements. These are certainly things that a designer must consider when choosing between Precomputed Gain and Extended implementations for navigation and targeting algorithms.

Chapter 13

Summary and Conclusions

This thesis began with an introduction to Kalman filtering and its use in spaceflight navigation, in particular autonomous navigation for a Mars Sample Return mission. The scope of the document was outlined, and the objective of the analysis was said to be a quantitative understanding of the performance of the Precomputed Gain and Extended algorithms. A large portion of the document was dedicated to background information for the analysis, including: design of rendezvous trajectories; the baseline rendezvous trajectory; discrete Kalman filtering; the environment model; the Precomputed Gain and Extended implementations of the Kalman filter; targeting algorithms; and, the design of the simulated flight computers for the Precomputed Gain and Extended systems. Finally, several chapters were provided to deal with the performance of the two implementations, including: the performance envelope; technology readiness level; complexity and computational burden; and, data uplink requirements for each algorithm. It is hoped that this work is as educational for the reader as it was for the author.

This brief chapter will summarize the main points from Chapters 9 through 12, and make some final comments about the applications that these two filters might be suited for. In addition, there is no shortage of future work that could be done to expand or extend the results presented here. Many of these ideas are named in Section 13.1 below.

Several basic conclusions come out of Chapter 9 on the performance envelope. The Precomputed Gain system was found to have very clean position dispersions inside of its performance envelope, due to targeting algorithms designed to restore the trajectory to the nominal. It was found to be limited by a conservative filter configured with the worst case execution error model and 1000 m/day of unmodeled accelerations (case 6 in Table 9.12). It was also found to be limited by an environment model providing the worse case maneuver execution error and 4000 m/day of unmodeled accelerations, while the filter was configured nominally. The Extended system was not seen to be limited by the two cases above. Unmodeled accelerations of 4000 m/day caused the mosaic mode to be required several times, however flexible targeting algorithms allowed the completion of the rendezvous anyhow. The position dispersions for the Extended system were found to be larger than for the Precomputed Gain system, however this was merely a consequence of the targeting algorithms selected for this study. These larger dispersions resulted in a larger 3-σ required Δv for nominally large maneuvers, however gains made by using energy control on the football orbit still resulted in a net reduction in *total* required Δv.

The use of the full estimated state in the Extended system, and filter linearization about this estimate, resulted in improved position and velocity navigation in the Extended system over the Precomputed Gain system. As a direct result of improved position navigation, target pointing errors were found to be somewhat lower than those seen for the Precomputed Gain system.

It may be suggested that the use of flexible onboard targeting algorithms in the Extended system does not provide a fair comparison between these two algorithms. The author argues that the ability of the Extended system to use these flexible algorithms is a *significant strength* over the Precomputed Gain system, perhaps the most important distinction. Thus, rather than considering it an unfair comparison, it is a conclusion of this document that the Extended system is found to be more versatile, due largely to the targeting algorithms allowed by the implementation.

Several conclusions should be drawn from Chapters 10 through 12, as well. While these two systems were found to be tied near TRL 5 for their technology readiness, it was acknowledged that the Extended Kalman filter has space heritage in other applications, and thus has a significant advantage over the Precomputed Gain system in this metric. The Extended system was also found to have data uplink requirements that were orders of magnitude lower than the Precomputed Gain system, which could certainly prove significant on a mission with a tight communications budget. On the other hand, the Precomputed Gain system was shown to have very simple onboard flight software, which was of course by design. It is expected that the flight software for the Precomputed Gain system will be shorter, simpler, and require fewer computational resources, which could perhaps reduce the expense of development of onboard software by a significant amount.

In general, the Extended system was found to be more robust, with a larger performance envelope. Flexible targeting algorithms allow closed-loop corrections even when the chaser significantly leaves the "nominal" rendezvous trajectory. The Precomputed Gain system, while bound by more restrictive operating parameters, still functioned well within all reasonable bounds expected for the actual mission. Performance gains for using the Extended system over the Precomputed Gain system are marginal, while for the Precomputed Gain system the onboard flight software is fast and simple. The final conclusion is this: for robustness, the author recommends the Extended Kalman filter with onboard targeting algorithms, but for onboard simplicity the author recommends the Precomputed Gain Kalman filter with precomputed maneuver corrections.

13.1 Future Work

There are many things that could be done to expand or extend the results presented in this thesis. The following is a short list of things the author sees as potential sources of future study.

Improved Targeting Algorithms The Precomputed Gain system uses a linearized Lambert targeting algorithm to perform position and velocity corrections and stay near the nominal trajectory. This was found to be an appropriate solution with reasonable results. The Extended system, on the other hand, uses a large variety of targeting algorithms, and many comments were made in Section 9.3.2 about possible improvements to the algorithms selected. It is believed that further work on the targeting algorithms for the Extended system could reduce the amount of total Δv required for the rendezvous. In addition, modifications to the glide slope targeting will improve the position dispersions after the final maneuver.

Equivalent Targeting Algorithms Another interesting variation on the targeting algorithm strategy would be to provide linearized Lambert targeting for the Extended system, and then compare the simulation results between the Precomputed Gain and Extended systems. This would remove the targeting algorithms as a variable in the analysis, and allow a more direct comparison of a filter linearized about a nominal trajectory (Precomputed Gain system) with one linearized about the state estimate (Extended system).

MSR Trajectory This analysis was performed for an orbit and baseline trajectory suitable for the MTO spacecraft, which is seen as the most likely candidate for the first rendezvous demonstration. A necessary extension to this work is an analysis for the orbital elements and baseline trajectory anticipated for the actual MSR mission.

Rotational Dynamics The chaser vehicle in this study was modeled without attitude dynamics. Thus, certain pertinent issues cannot be discussed, such as the ability of the spacecraft to slew between attitudes required for maneuvering and taking measurements. Thus, attitude dynamics would be a useful addition to the rendezvous simulation.

Improved Gravity Model This analysis assumed a spherical planet of uniform mass, knowing that higher order gravity terms will have a small effect on the relative motion of the two spacecraft, particularly on the time scale studied. Still, the fidelity of this analysis could be improved by including higher order gravity terms, such as the J2 term.

Full Precomputed Gain State Estimate It is mentioned in the preceding chapters that only 12 states (position and velocity for both vehicles) are estimated in the Precomputed Gain flight computer. It was expected that this capability would be added to the simulation should the 12-state estimate prove inadequate, however the current system functions well as-is. Still, it would be interesting to update the flight computer to use the full state estimate (even if this requires some additional onboard processing and complexity), and see what gains could be made.

Additional Cases A final improvement that could be made using the same analysis tool used in this thesis would be to fully populate Table 9.12 with cases, and perhaps extend it, to completely define the performance envelope of both algorithms. In this study, time constraints prevented such a thorough examination, but the results would certainly be useful to mission designers.

[This page intentionally left nearly blank.]

Bibliography

[1] NASA ARC/GSFC. Technology Readiness Levels Applied to Software. June 1999.

[2] Richard H. Battin. *An Introduction to the Mathematics and Methods of Astrodynamics, Revised Edition.* American Institute of Aeronautics and Astronautics, Inc., Reston, Virginia, 1999.

[3] CNES. Maneuver Execution Accuracy and Knowledge. Notes from Teleconference, September 2002.

[4] Arthur Gelb, editor. *Applied Optimal Estimation.* The MIT Press, Cambridge, Massachusetts, 1974.

[5] Geoffrey T. Huntington. Trajectory Design and Analysis of a Mars-Orbit Rendezvous Flight Experiment. Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, February 2004.

[6] The MathWorks Inc. MATLAB/Simulink. Software Package, 2002.

[7] Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[8] William M Lear. Kalman Filtering Techniques. Technical Report JSC-20688, NASA Mission Planning and Analysis Division, Houston, Texas, 1985.

[9] John C. Mankins. Technology Readiness Levels: A White Paper. Technical report, NASA Advanced Concepts Office, April 1995.

[10] Richard Mattingly, Steve Matousek, and Frank Jordan. Mars Sample Return, Updated to a Groundbreaking Approach. *IEEEAC*, (1392), November 2002.

[11] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 141 of *Mathematics in Science and Engineering.* Academic Press, New York, 1979.

[12] Office of the Secretary of Defense. Mandatory Procedures for Major Defense Acquisition Programs (MDAPS) and Major Automated Information System (MAIS) Acquisition Programs. Technical Report DoD 5000.2-R, Department of Defense, Washington, DC, April 2002.

[13] Peter V O'Neil. *Advanced Engineering Mathematics.* PWS Publishing Company, Boston, fourth edition, 1995.

[14] Warren R. Reid, editor. *Public Papers of the Presidents of the United States: John F. Kennedy.* United States Government Printing Office, 1963. Containing the Public Messages, Speeches, and Statements of the President.

[15] Ralph Roncoli, Bill Strauss, and Dolan Highsmith. Mars Exploration Rover Project Planetary Constants and Models - Version 2. Interoffice Memorandum, November 2002.

[16] George M. Siouris. *An Engineering Approach to Optimal Control and Estimation Theory.* John Wiley & Sons, Inc., New York, 1996.

[17] David A. Vallado. *Fundamentals of Astrodynamics and Applications.* McGraw Hill, New York, 1997.

[18] James R. Wertz, editor. *Spacecraft Attitude Determination and Control.* D. Reidel Publishing Company, Boston, Massachusetts, 1986.