AGENT AND MODEL-BASED SIMULATION FRAMEWORK FOR DEEP SPACE NAVIGATION DESIGN AND ANALYSIS

A Thesis
Presented to
The Academic Faculty

by

Evan Anzalone

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
August 2013

AGENT AND MODEL-BASED SIMULATION FRAMEWORK FOR DEEP SPACE NAVIGATION DESIGN AND ANALYSIS

Approved by:

Prof. Dimitri Mavris, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Carrie Olsen
Marshall Space Flight Center
National Aeronautics and Space Administration

Dr. Jason Chuang
Marshall Space Flight Center
National Aeronautics and Space Administration

Dr. Russell Peak
School of Aerospace Engineering
Georgia Institute of Technology

Prof. David Spencer
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: 17 May 2013

To Tarra

ACKNOWLEDGEMENTS

This work was made possible by the financial support of NASA External Training Requests and the Air, Space, and Missile Defense Association 2011 Mayor Loretta Spencer Scholarship. Many individuals contributed to help and support me while working through this dissertation and I am grateful for all of them.

First of all, I would like to express my gratitude and appreciation for the members of my committee, Profs. Mavris, Spencer, and Peak, and Drs. Olsen and Chuang. I appreciate the time spent discussing this work with me, reviewing the research, providing your comments, and guiding me through this process. I have a great amount of respect for each of you and am extremely grateful for the time and energy you have invested in me.

I would like to specifically acknowledge Prof. Mavris for his years of support and advising, from offering me the opportunity to perform research in the Aerospace Systems Design Laboratory to supporting me in my pursuit of a PhD and serving as my advisor as I entered the post-graduate workforce in Huntsville. I appreciate the emails, phone calls, and meetings we had as this work was coming to a close. My engineering skills and design approach are strongly rooted in what I learned under your tutelage.

I must also acknowledge Dr. Olsen for her continued support of my PhD as I hired into her branch at MSFC. Thank you for pushing me forward, providing encouragement, and helping me to get my ideas out. I am also grateful to Dr. Olsen for making the drive to Atlanta and serving on my committee.

I also owe gratitude to Dr. Chuang. On my first day at MSFC, Dr. Chuang had already provided me with a great paper on strapdown inertial navigation. That paper and all of the work we have done together at MSFC have continued to feed my interest and fascination with space navigation approaches. I also am grateful to him for making the journey to Atlanta twice, advising me on the research at Huntsville, providing an outlet to brainstorm navigation with, and serving on my proposal and defense committees.

I also must acknowledge Prof. Spencer for serving on my proposal and defense committee. I am grateful to have your excellent feedback from your exceptional experience in deep space mission analysis and design. Your feedback helped steer this thesis into the work that it is, providing support for the focused analysis and presentation of the results.

I also must recognize Dr. Peak for his help and support through emails, reviews of my models, and also for serving on my defense committee. I am grateful for your time and recommendations in support of my modeling implementation and analysis.

Also, a great deal of thanks is due to the administrative staff at ASDL, specifically Ms. Loretta and Allison for helping to get all of the paperwork processed. Your assistance was invaluable and I appreciate your time and help.

This thesis was also supported by multiple reviews by Donna Hamby, who helped read over the document and provide editorial support. Thank you for helping to catch all of the comma splices and grammatical errors.

I also must acknowledge my gratitude and appreciation for my friends and classmates at Georgia Tech. From that first year of long nights in the basement of ESM, to the qualifier study nights in the house on Mount Royal, to the continued visits. You have been with me through this journey from my early classes, through qualifiers, and on to my thesis defense. All of the long days and long nights we worked and played made this journey incredible, and also created everlasting memories.

I also owe a large debt of gratitude to my parents, for always supporting me in the pursuit of my dreams, and trusting me to chart my own course. You have always supported me and given me unending encouragement and understanding. Thank you for always being there for me, and all the trips and moves over the years.

Lastly, I owe a special debt of gratitude to my wife, Tarra. You have supported me, encouraged me, and been at my side since we met that wonderful summer of 2009. Thank you for your understanding, patience, support, and love on the long nights and weekends I worked on this dissertation and followed my dreams.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS OR ABBREVIATIONS

ABM	Agent-Based Modeling.
b	Clock Bias.
c	Speed of Light.
CB	Central Body.
CCSDS	Consultative Committee for Space Data Systems.
C&DH	Command and Data Handling.
COBE	Cosmic Background Observer.
CRAIVE	Concept, Analysis, Implementation, Verification, and Evaluation.
d	Clock Drift.
DEAP	Distributed Evolutionary Algorithms in Python.
DSN	Deep Space Network.
DTN	Delay and Disruption-Tolerant Networking.
ET	Ephemeris Time.
F	Applied Force.
f	Filter Dynamics Model.
GNSS	Global Navigation Satellite System.
GPS	Global Positioning System.
$\mathbf{G}_r$	Receiver Gain.
$\mathbf{G}_t$	Transmission Gain.
(H)	Hypothesis.
h	Measurement Model.
H	Measurement Observation Model.
$\mathbf{h}_{-1}$	Flicker Frequency Noise.
$\mathbf{h}_{-2}$	Random Walk Frequency Noise.
$\mathbf{h}_0$	White Frequency Noise.
$\mathbf{h}_1$	Flicker Phase Noise.
$\mathbf{h}_2$	White Phase Noise.

HEAO	High Energy Astrophysics Observer.
HPOP	High Performance Orbit Propagator.
INCOSE	International Council On Systems Engineering.
IPN	Interplanetary Internet.
K	Kalman Gain.
$\mathbf{k}_{Boltzman}$	Boltzman Constant.
λ	Wavelength.
LEO	Low Earth Orbit.
$\mathbf{L}_r$	Receiver Line Losses.
$\mathbf{L}_{space}$	Space Transmission Losses.
$\mathbf{L}_t$	Transmission Line Losses.
LTP	Licklider Transmission Protocol.
MBSE	Model-Based Systems Engineering.
MER	Mars Exploration Rover.
MGA	Medium Gain Antenna.
MRO	Mars Reconnaissance Orbiter.
MSL	Mars Science Laboratory.
μ	Gravitational Constant.
N	Noise Power.
NNAV	Network-Based Navigation.
N(x,y)	Random Noise Variable with x Mean and y Standard Deviation.
OMG	Object Management Group.
P	Covariance Matrix.
PLGA	Parachute Low Gain Antenna.
$\mathbf{P}_{Received}$	Power Received.
$\mathbf{P}_t$	Power Transmitted.
Q	Process Noise.
R	Measurement Error Matrix.
r	Position of Agent.

(RQ)	Research Question.
S	Spectral Density of Clock Noise.
s	Transmission Distance.
σ_r	Error in Position.
σ_v	Error in Velocity.
σ_y^2	Allan Variance.
SNAPE	Space Navigation Analysis and Performance Evaluation.
SNR	Signal to Noise Ratio.
SysML	Systems Modeling Language.
τ	Time Period of Clock Averaging.
TDB	Barycentric Dynamical Time.
TDRS	Tracking Data and Relay Satellite.
x	State Vector.
$\mathbf{T}_n$	Noise Temperature.
TONS	TDRS Onboard Navigation System.
TWTA	Traveling Wave Tube Amplifier.
UML	Unified Modeling Language.
v	Dynamic Noise.
v	Velocity of Agent.
VLBI	Very Long Baseline Interferometry.
w	Random Noise in Model.
XNAV	X-Ray Based Navigation.
y	Measurement.

SUMMARY

As the number of spacecraft in simultaneous operation continues to grow, there is an increased dependency on ground-based navigation support. The current baseline system for deep space navigation utilizes Earth-based radiometric tracking, which requires long duration, often global, observations to perform orbit determination and generate a state update. The age, complexity, and high utilization of the assets that make up the Deep Space Network (DSN) pose a risk to spacecraft navigation performance. With increasingly complex mission operations, such as automated asteroid rendezvous or pinpoint planetary landing, the need for high accuracy and autonomous navigation capability is further reinforced.

The Network-Based Navigation (NNAV) method developed in this research takes advantage of the growing inter-spacecraft communication network infrastructure to allow for autonomous state measurement. By embedding navigation headers into the data packets transmitted between nodes in the communication network, it is possible to provide an additional source of navigation capability. Simulation results indicate that as NNAV is implemented across the deep space network, the state estimation capability continues to improve, providing an embedded navigation network.

To analyze the capabilities of NNAV, an analysis and simulation framework is designed that integrates navigation and communication analysis. Model-Based Systems Engineering (MBSE) and Agent-Based Modeling (ABM) techniques are utilized to foster a modular, expandable, and robust framework. This research has developed the Space Navigation Analysis and Performance Evaluation (SNAPE) framework. This framework allows for design, analysis, and optimization of deep space navigation and communication architectures. SNAPE captures high-level performance requirements and bridges them to specific functional requirements of the analytical implementation. The SNAPE framework is implemented in a representative prototype environment using the Python language and verified using industry standard packages.

The capability of SNAPE is validated through a series of example test cases. These analyses focus on the performance of specific state measurements to state estimation performance, and demonstrate the core analytic functionality of the framework. Specific cases analyze the effects of initial error and measurement uncertainty on state estimation performance. The timing and frequency of state measurements are also investigated to show the need for frequent state measurements to minimize navigation errors. The dependence of navigation accuracy on timing stability and accuracy is also demonstrated. These test cases capture the functionality of the tool as well as validate its performance.

The SNAPE framework is utilized to capture and analyze NNAV, both conceptually and analytically. Multiple evaluation cases are presented that focus on the Mars Science Laboratory's (MSL) Martian transfer mission phase. These evaluation cases validate NNAV and provide concrete evidence of its operational capability for this particular application. Improvement to onboard state estimation performance and reduced reliance on Earth-based assets is demonstrated through simulation of the MSL spacecraft utilizing NNAV processes and embedded packets within a limited network containing DSN and MRO. From the demonstrated state estimation performance, NNAV is shown to be a capable and viable method of deep space navigation. Through its implementation as a state augmentation method, the concept integrates with traditional measurements and reduces the dependence on Earth-based updates. Future development of this concept focuses on a growing network of assets and spacecraft, which allows for improved operational flexibility and accuracy in spacecraft state estimation capability and a growing solar system-wide navigation network.

CHAPTER I

INTRODUCTION

From the earliest nomadic tribe to the latest world traveler, the capability of determining and tracking one's location has enabled longer and further journeys, from coast to coast and ground to sky. Knowledge of one's current position in relation to starting and ending points allows for the definition of the trajectory required to reach a certain destination. The guidance and navigation process is defined by knowing where you are and using that to inform the direction of travel needed to reach your endpoint. These techniques are used constantly in one's life and by a multitude of species from the tiniest ant to largest whale, traveling from breeding grounds to food supplies and back again. This can occur over ranges of a few meters to thousands of kilometers, from the depths of the oceans to the vastness of space. The focus of this dissertation is on the continued development of navigation algorithms and techniques for spacecraft traveling beyond Low Earth Orbit.

1.1 Overview

This research presents a new approach for deep space navigation that takes advantage of the growing communication infrastructure, specifically utilizing increased spacecraft-to-spacecraft transmissions. The layout of this thesis is based upon proven Systems Engineering and design methods [27] [68]. The research approach and organizational structure used is captured by the acronym CRAIVE, with individual steps focused on capturing the **C**oncept of Operations, **R**equirements of the navigation system, **A**nalysis and modeling of the system, **I**mplementation of analytical tools, **V**erification of their operation, and **E**valuation of the proposed concept.

This can be visually displayed by a v-shaped diagram, as given in Figure 1, with references to specific chapters which focus on each. The left side of the diagram focuses on the description and modeling of the navigation system and analysis framework. This captures the decomposition from the high level concept to the analysis requirements to the modeling

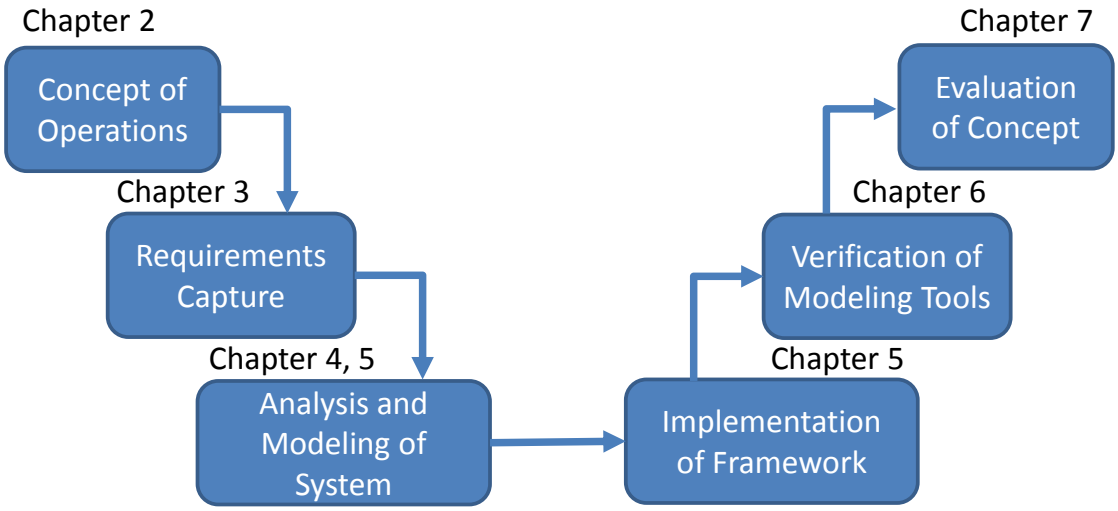


Figure 1: CRAIVE Structure of Research

of the specific navigation system under study. The right side of this model begins at the lowest level and provides an implementation of the system models. This modeling and simulation framework is then utilized to perform verification of its capabilities to demonstrate that the implemented framework meets the analysis needs. Finally, this verified framework is used to evaluate the navigation concept and provide numerical analysis of its capabilities.

The specific focus of each chapter is given in Table 1. This chapter discusses the need and importance of navigation capability to mission performance and success for deep space and human exploration programs. The Network-based Navigation (NNAV) architectural foundation and concept are described in Chapter 2. With the concept defined, Chapter 3 develops an approach to modeling of deep space navigation systems and compares it to other current approaches. The proposed method integrates aspects of Model-Based Systems Engineering and Agent-Based Modeling to capture the complexities and intricacies of the navigation systems design space. To provide increased insight into the analysis process, Chapter 4 describes the analytical background in terms of the underlying modeling and estimation assumptions.

Chapter 5 documents the development and implementation of a Space Navigation Analysis and Performance Evaluation (SNAPE) Framework to capture the architecture, design,

Table 1: Structure of Thesis

Chapter	Focus
1	Introduction to Deep Space Navigation
2	Proposed Network-Based Navigation Concept
3	Analysis Requirements, Methods, and Approach
4	Navigation Analytical Background
5	Implementation of Modeling and Simulation
6	Verification of Analysis Framework
7	Evaluation of Network-Based Navigation
8	Summary of Research, Contributions, Future Work

and capabilities of a deep space navigation system. This chapter documents the development of Model-Based Systems Engineering techniques using SysML tools to capture the analysis framework requirements, processes, and structure, which are derived from the requirements and needs of a generic navigation system. This chapter also demonstrates the linkage of these developed models to the Python-based software implementation of the modeling and simulation aspects of the SNAPE framework. With the analysis tool developed, verification test cases utilizing standard tools are described in Chapter 6. These results capture high-level trends in navigation system design that demonstrate the capabilities of the simulation framework and its application to system analysis requirements. Additionally, these tests serves as validation of SNAPE's functionality.

SNAPE is then executed to analyze the capabilities of and validate NNAV. Chapter 7 documents the test cases for the proposed navigation architecture and presents a series of analyses to capture the characteristics of NANV using the implemented modeling and simulation framework. The results of these test cases demonstrate the applicability of NNAV to deep space navigation, particularly capturing its benefits in terms of robustness to delays in ground-based tracking and onboard navigation capability, with operation autonomous of Earth-based ground support. This thesis concludes in Chapter 8 with a summary of the presented research, its contributions, and forward research directions and future work.

1.2 Need for Navigation in Space-Based Applications

Since the first successful insertion of a man-made object into orbit about the Earth, engineers and scientists have continued to develop more complex satellites performing increasingly complex missions. The functionality of space-borne assets has increased dramatically from the simple beacon transmission of the Sputnik satellite. As the capability of launch systems and satellite components increase, it is possible to put larger and more complex hardware into orbit. With each mission and continued advancement of spacecraft technology, missions have ventured farther out from Earth to our planetary neighbors, such as the Messenger mission to Mercury, and distant locales, such as the New Horizons mission to Pluto and Charon. As the missions and spacecraft have become more complex, the requirements on spacecraft navigation become more stringent[80]. Advances in navigation techniques have also proven to be an enabler for new missions[99]. The main drivers for spacecraft navigation are a function of the spacecraft's trajectory from orbital entry and cruise, its onboard hardware capabilities, planetary orbit entry errors and uncertainty, and scientific observation requirements.

1.2.1 Orbital Correction Maneuvers

Even with advanced celestial dynamics models and navigation measurements, it is not possible to perfectly predict spacecraft ephemeris (trajectory over time). This is due to the complexity of the system being modeled, and assumptions used in the modeling of dynamic effects in deep space. As such, there is inherent error in predicting and propagating a spacecraft's trajectory. This is limited due to a range of issues from gravitational model uncertainty to finite precision computations.

Due to these effects, most spacecraft's mission concepts include several trajectory correction maneuvers (TCM) in order to tweak the spacecraft's flight path. These corrections are needed to ensure correct planetary flyby conditions, orbit insertions, and correct for initial launch orbit errors. The primary information used to plan a TCM is the navigation data. Again, the accuracy of the TCM is limited by the accuracy of the initial estimated state. Orbit observations allow for an analysis of the observed trajectory compared to the

planned. Ground analysis compares the observed to the predicted state and desired state to ascertain if any thrust is needed. Improvements in both navigation accuracy and state update rate will increase knowledge of the spacecraft's position. This will further reduce the need for large TCMs by the minimization of initial state error and increasing the effectiveness of corrections[129]. Overall error is still limited by the assumptions made in the predicted dynamics, which high precision navigation dynamics models can improve.

1.2.2 Landing Requirements

Some of the most difficult missions involve landing a probe or rover on an extraterrestrial planetary surface. There have been many successes and several failures. For stationary assets, such as the Phoenix Martian Lander, the science data collected is limited to the probe's location on the planetary surface. Rovers, such as the Mars Exploration Rovers (MERs) Spirit and Opportunity, contain the capability to traverse the planetary surface. This increases the scientific range of these assets to kilometers over large lengths of time. Even so, this capability is limited and much deliberation and analysis goes into choosing a landing site.

For scientific missions, it is desired to arrive in a general vicinity of scientific interest or region. But as man begins to push outwards from Earth, the need to deliver supplies to a Lunar or Martian outpost will become very frequent. It will be increasingly important to accurately land these resources close to a predefined location (nearby to the human presence, or within range of any local surface vehicles) to minimize the time and effort required to retrieve the supplies.

Typical landing systems have relied on techniques involving aerobraking or parachutes to gently land with minimal control during descent. As such, it is paramount to have knowledge of the spacecraft's state well ahead of planetary entry. This knowledge allows ground operators to predict the spacecraft's entry vector and estimated landing site. With increasing accuracy of the navigation knowledge, the operators can issue more precise thrusting commands to the spacecraft to tune its entry trajectory. This increased navigation performance allows for the capability of precise directed planetary entries and landing.

Martian probes provide a good example of the accuracies involved in planetary entry conditions. For the MERS, the navigation error at Martian atmospheric entry was 9 kilometers, which propagated to 80 kilometers of surface landing uncertainty and error[74]. As Lightsey and Mogensen [74] describe, even systems with in-atmosphere correction such as the Mars Science Laboratory, are predicted to have best-case error on the order of 10 kilometers. As such, the landing error is directly related to navigation knowledge. Due to transmission time delay and on-ground processing time, the last control update is limited to six hours prior to atmospheric entry [74]. Further trajectory corrections must be executed by onboard navigation and guidance systems, with the accuracy directly affecting landing capability.

1.2.3 Scientific Pointing Requirements

Modern deep space probes typically contain instrumentation allowing radar ranging, complex spectroscopy, and high definition imagery. In order to use such powerful high-resolution instruments, stricter requirements are placed on spacecraft pointing performance. To meet these needs, onboard attitude state determination has steadily improved to enable focusing observations on specific areas of interest.

With the growth in quality and quantity of data, it is important to link the measurements accurately to a physical location (and site observed). This is needed both to link various independent data sets (from multiple spacecraft for example), as well as to build up repeated measurements of a specific area. The spacecraft's state is also needed to process any observations. The combination of these two data sets allows scientific researchers to link the observations to an exact physical location on a planetary body. This is important both for Earth-based and extraterrestrial observations. Reducing the error of on-orbit positioning reduces the error in data collection, allowing maintenance of the required high accuracy position information on measurements with increasing resolution and capability.

1.3 Complexity in Space Navigation

The navigation problem of ascertaining one's current position and velocity is a very complex problem. This is due to uncertainties in dynamics models, measurement accuracy,

and resolution limitations. For ground-based navigation, this can be performed relatively accurately with imprecise measurement (such as simply using a map, compass, and observations of the landscape). But as spacecraft travel farther and farther from Earth, and the distances traveled increase to such large values, navigation becomes increasingly difficult. This is due to limited observation data, and finite precision. For example, to maintain a relative positioning accuracy as a spacecraft's distance from Earth increases requires a continual improvement in observation resolution. At the most fundamental level, numerical precision and computational accuracy limit this capability. The driving effects are due to many parts, including uncertainty in the dynamic models, physical spacecraft limitations, and the measurement process.

1.3.1 Gravitational Uncertainty

No calculated or predicted ephemeris is perfect. Simulation of celestial dynamics is a very complex problem, and it is impossible to capture every effect. Additionally, simulation capability is limited at the lowest level on numerical round-off errors. Analytical models of general relativity paired with ground observations do allow for an accurate determination of planetary ephemeris.

Even so, there is still some associated error. From the observations and dynamic models, a planet's mass can be estimated. Propagating a spacecraft in that planet's orbit with an accuracy on the order of meters or centimeters requires much more detailed information on the planet's mass and gravitational potential distribution. Such models exist for Earth[89] and Mars[59], which capture the data to a very high degree. Additional effects such as drag in the interstellar medium and solar radiation pressure must also be accounted for to enable high accuracy state propagation.

The prediction capability is directly related to the knowledge of the dynamics being modeled. For principal investigators of science instruments, the primary information required is not predicted trajectory but actual traveled trajectory (and location of spacecraft at the time of observation). This data is generated by post-processing navigation data, which can be more accurate then forward-prediction due to the use of smoothing functions

on large data sets. Even with advanced filtering and least squares estimation approaches, the accuracy of this information is limited by navigation measurement uncertainty [124]. Due to this limitation, it is important to capture high quality navigation measurements in order to capture all effects in the true dynamics and aid in trajectory prediction and reconstruction.

1.3.2 Signal Travel Time

One of the main complexities in generating this navigation data is a product of the environment itself and the distances involved. Typically, deep space navigation is performed by Earth-based assets. As seen in Figure 3, the time for a signal to reach its destination can vary from several minutes to several hours based on the geometry involved. For the New Horizons probe with corresponding trajectory given in Figure 2, the round trip light time increases with distance. This data was generated using the published trajectory of New Horizons from January 19, 2006 to January 1, 2010 [1]. As the probe reaches Pluto and Charon, the one way light travel time ultimately reaches nine hours[8]. This large distance drives the signal strength required from Earth that the spacecraft is able to detect Similarly, very large and sensitive receivers are required to receive any signals being transmitted in return, especially due to the spacecraft's limited transmission capability.

The transmission travel time, along with the time required for post-processing and analysis on the ground, produces a latency in any state measurement based on this observation. As such, a calculated navigation solution is a measurement of where the spacecraft *was* and not where it *is*. This delay also affects transmission capabilities, in that the ground-based assets must be pointed based on the predicted delay and where the satellite *will be* when the signal has traveled such a distance. The ground assets must therefore track ahead of the spacecraft. Additionally, the efficiency of transmitting to a spacecraft is driven by knowledge of the spacecraft's position, which can drive pointing losses. As navigation fixes are generated, errors in pointing and data transmission can be reduced.

[1] http://naif.jpl.nasa.gov/pub/naif/pds/data/nh-j_p_ssspice-6-v1.0/nhsp_1000/

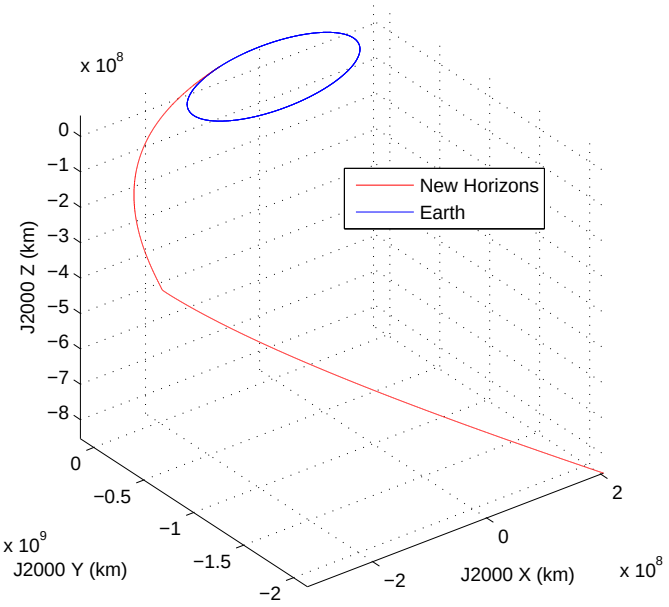


Figure 2: New Horizons Trajectory

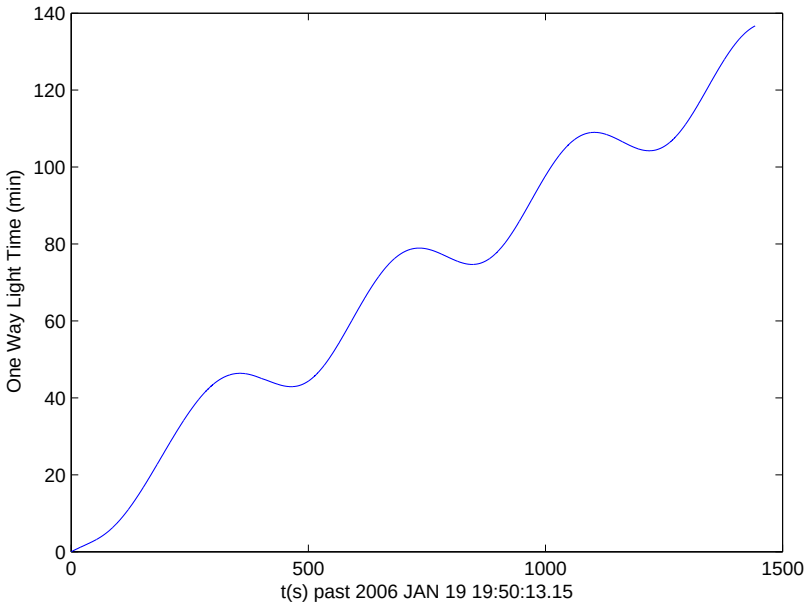


Figure 3: Light Travel Times to Earth

1.3.3 Spacecraft Subsystem Limitations

Due to the issues involved with signal travel and deep space communication, the ideal solution involves the spacecraft performing navigation autonomously on-board. There are several limitations to using onboard satellite systems to perform complex navigation and state estimation routines. Some research has been performed to begin to implement this, but in limited scope[20]. There are limitations both in the field of algorithm development and the required hardware and computational systems.

These limitations are intrinsically linked. Due to the long lead time for deep space missions and focus on flight-proven systems, the amount of computing power limits the implementation of advanced algorithms. The hardware available is further reduced by requirements on radiation hardening. This also limits on-board memory storage. Due to these limitations, it is difficult to develop autonomous algorithms of sufficient capability to be run on memory- and processing-constrained systems. As more powerful computers are flown, there is a possibility for improvement of these algorithms. Even so, any algorithm must be incredibly efficient to be used on-board during flight.

Additionally, the spacecraft itself is physically constrained. Due to launch vehicle limitations, both the volume and mass are limited. This constrains which and how many instruments can be installed on the spacecraft. To transmit information back to Earth, a large directional antenna is required. For example on Cassini, seen in Figure 4, the communications dish is maximized to the allowable launch volume, with a diameter of four meters. The size of the dish and receiver determine the strength of any transmitted signal. This also limits available power reception capability and required signal-to-noise ratios SNR for communication. The transmission strength is also limited by the limited onboard generated power available.

As a spacecraft travels farther from the sun, the solar flux reduces proportionally to the square of the distance. Solar panels are thus decreasingly useful the farther a spacecraft travels from the sun. The Juno spacecraft, destined for study of Jovian system, in Figure 5, is the first spacecraft of its size to generate power solely via solar arrays at a distance as far as from the Sun as Jupiter. It contains three solar panels, with a total effective solar cell

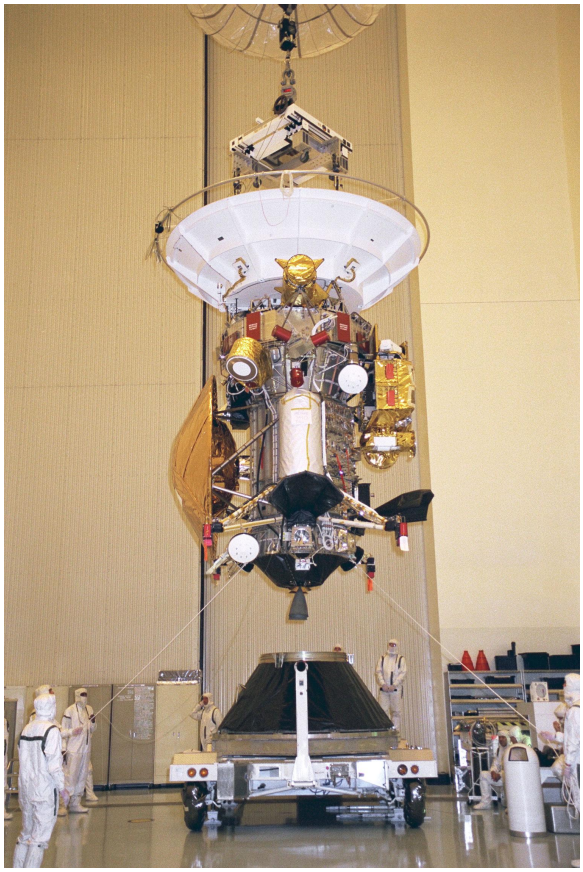


Figure 4: Cassini Spacecraft (NASA/JPL)

Figure 5: Juno Spacecraft[53]

area of forty-five square meters[53].

Typically, radioisotope thermoelectric generators (RTG) are used to provide power for deep space missions. Even when these generators are used to provide steady power, the spacecraft consumption rate is still limited. Therefore onboard processes and instrumentation must be very efficient, and operations must be coordinated and scheduled ahead of time in order to stay within operational bounds and maintain power margins.

As such, any onboard navigation hardware needs to have limited power requirements and a minimal affect on other spacecraft operations. Due to these factors, spacecraft navigation is inherently difficult due to the environment involved, the signal delay, and the tight spacecraft physical and operational constraints. Many methods of navigation are currently used, most external to the spacecraft. But as autonomy and navigational accuracy requirements increase, it is necessary to shift these functions onboard.

1.4 State of the Art Space Navigation Systems

As satellite and space technology has evolved, the capability for space navigation and tracking has also increased. The earliest methods involved optically tracking an object across the sky. Optical observations led to the earliest solar system models and continue to aid in developing and maintaining planetary body ephemerides. Observations of the motion of fixed stars in the sky also informed early models of the Earth's orientation.

Current navigation methods utilize a range of signals and observation techniques to perform in-spacecraft positioning and external tracking. Some positioning techniques involve the use of ranging signals from beacon satellites, optical observations of planetary bodies, and internal tracking of the spacecraft's state. External tracking has been performed using primarily radio waves to both measure an asset's position in the sky (right ascension and declination) as well as direct measurement of the distance to and radial rate of the spacecraft.

1.4.1 Optical Navigation

The measured ephemerides of planetary bodies have been utilized for deep space navigation of spacecraft to the outer solar system[24] [38] via the method of Optical Navigation for missions as early as Mariner. The use of optical cameras on spacecraft is the key enabler for utilizing these techniques. The primary optical observation instruments are often used in lieu of a dedicated navigation camera. Performing this analysis requires three main external pieces of information: the spacecraft's attitude during the observation, the planetary body's physical dimensions, and the current position of the body being observed. This data is compiled in a range and bearing-type method[130]. Other implementations of optical navigation measure the angle between a guide star and the planetary object being tracked [24].

The attitude is obtained by analysis of the image generated. The planetary body must not fill the entire observation, in order to gain a viewing of the background stars. Post-processing on the ground uses techniques similar to star trackers, which look for known constellations or groupings of stars and then uses this information to obtain an inertial pointing

reference. The key assumption is that the celestial background is relatively inertially-fixed with respect to our solar system. The resulting attitude is then joined with further analysis of the body's state to obtain a range vector.

Multiple methods exist to analyze the observed planetary body[130]. Information about the size and shape of the body must be assumed. Typically, this amounts to knowledge of the body's diameter, which has been estimated by celestial dynamics models or direct observation (i.e., linking optical observations from other spacecraft using other navigation methods).

This analysis works best for bodies with very well known shape and dimensions. The accuracy of this method is limited by the knowledge of the body. Due to this fact, this method is usually limited to observations of planets, which have well defined shapes that are easy to capture.

Determining the observed dimensions is done by analysis of the observed shape. The centroid and outer shape of the planet is determined by fitting curves to the observed planetary images. This is limited by the resolution and quality of the camera used and its capability is linked to distance to the object. As the spacecraft approaches the body, a greater number of pixels are used, providing more data for analysis. Conversely, as the spacecraft is very near the body, the whole shape is not observed and horizon tracking techniques must be used to fit the observed arc of the planet's shape. Additionally, higher quality optics can allow the ground-based analysis team to distinguish between atmosphere and surface to improve the results.

Sextants have also been used for optical navigation. During Apollo 8, sextants were tested to obtain navigation fixes by measuring the angle between stars and Lunar surface features [20]. A diagram showing operationally how this was performed is given in Figure 6 [122]. An expansion of this to autonomous use is the Space Sextant [47]. This device was developed to be used for high altitude spacecraft in Earth orbit. It involves a gimballing platform with two telescopes. The instrument tracked a star's observed position in relation to the limb of the moon. Integrating this angular information for two stars with ephemeris knowledge of the moon provides enough information to calculate a position fix[19]. Studies

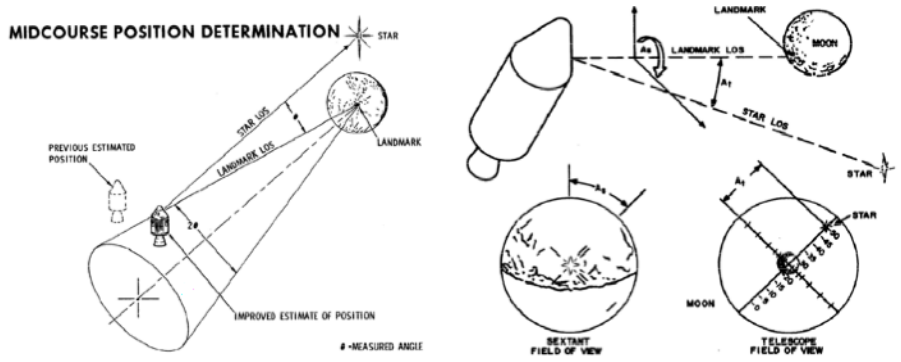


Figure 6: Orbit Determination via Lunar Observation [122](NASA)

predicted this approach would have an accuracy of 800 feet in position and .01 feet per second in velocity [47].

As seen, navigation by optical observations is a well-used and proven method of deep space navigation. Its application, though, is limited by the geometry of the trajectory. The combination of fixed camera optics and planetary ephemerides limit the use of optical navigation to only during certain observation-capable periods. Additionally, a large amount of ground analysis and support is required to calculate the actual navigation solution. This ground support comes at a high cost. Other methods instead utilize Earth-based measurements to track the spacecraft.

1.4.2 Radio-based Tracking

Radio signals can also be used as navigation sources. Worldwide systems such as LORAN-C have seen continuous use for at-sea tracking. Simple tracking involves a bearing-bearing method where measures of angles to multiple radio sources provide a two dimensional position fix [55]. A simple implementation of bearing determination utilizes a loop antenna [60]. The received power is proportional to the orientation of the plane of the antenna to the signal. When the two are orthogonal, no power is received and a heading can be determined. With multiple measurements to different beacons, a position fix can be calculated.

More advanced radio navigation systems have been developed for use in orbit. Early space navigation techniques involved tracking of a beacon signal emitted by the spacecraft. Radio systems are thus able to obtain a fix on the signal and determine its position in the

Figure 7: Deep Space Network Antenna (NASA/JPL)

sky. Analysis of a series of observations can provide orbital information [7]. As radio and computing technology has advanced, the techniques have evolved to include the spacecraft itself and multiple ground stations on Earth.

The Deep Space Network (DSN) [121] is the state-of-the-art technology for radiometric tracking. An overview of the architecture is given in Figure 7 [103]. The network consists of 3 large 70-meter diameter radio telescopes/transmitting stations outside of Madrid, Spain; Canberra, Australia; and Goldstone, California. These sites were selected to allow for continuous coverage of any deep space craft; however, with only one asset in the Southern Hemisphere, total sky coverage is limited. With the proven capability of DSN, some spacecraft missions have switched to using only radiometric tracking for navigation to reduce spacecraft development costs [121].

Several radiometric techniques are available via DSN: one-way, two-way, and three-way tracking at several frequencies, which are given in Table 2 [121]. One-way tracking is the method mentioned above, with the ground station tracking a beacon generated by the spacecraft to obtain right ascension and declination. Over time, this integrated information

Table 2: Deep Space Network Operating Frequencies

Band	Uplink Frequency (MHz)	Downlink Frequency (MHz)
S	2110-2120	2290-2300
X	7145-7190	8400-8450
Ka	34,200-34,700	31,800-32,300

can be used to determine orbital parameters, similar to way a series of optical observations of planets led to proven planetary ephemeris.

Two-way tracking can be utilized for spacecraft reasonably close to Earth. The two-way tracking consists of both a ranging code and a radial velocity tracking. This method is limited to use when the total length of observation with one site is greater than the round-trip signal transmission time. It is desired to minimize the number of stations used in this method of tracking in order to reduce errors, such as timing errors between ground stations.

In this ranging method, a predefined tone is transmitted to the spacecraft. Upon reception of the signal, it is re-transmitted back to the ground station. The ground site contains reference hardware to emulate the return signal. This reference is compared to the received tone, and the difference in phase represents a round-trip travel time. From this measurement and knowledge of the speed of light, the range to the spacecraft is determined. The round-trip time is used to reduce errors from one-way transmission.

As a spacecraft travels farther from Earth, and a transmitted signal cannot be received within one ground pass at a particular site, the ranging is handed off to the next station, thus becoming three-way ranging. To enable this transfer, the clocks and reference signals of the selected stations must be very well synchronized. This is typically done by having similar hardware and high bandwidth connections linking the ground stations together. Additionally, highly accurate clocks are used to minimize clock drifts between stations.

The main error source of this ranging technique is clock and oscillator instability [121]. Any difference between the actual transmitted signal and reference signal will result in an error. For example, in one-way ranging, the spacecraft produces the reference signal. The most stable space-qualified crystal oscillators fluctuate about one part in 10E13 over averaging intervals of 1000 seconds. In two-way tracking, ground-based crystals and oscillators

are used that are stable to a few parts in 10E15 over a typical round trip travel time. As such, there is a clear benefit to using two-way ranging, and one-way tracking is limited by the accuracy of the onboard oscillators.

In addition to generating ranging data, the DSN also measures the Doppler shift of signals received from the spacecraft. The Doppler shift is a measured change in frequency due to a difference in velocity between the observer and the tracked object. Doppler data is typically recorded continuously during a tracking pass at the DSN station. From a single pass, it is thus possible to determine spacecraft radial velocity, right ascension, and declination. Velocities normal to the line-of-sight are inferred from several days or more of Doppler data (tracking right ascension and declination change).

The primary source of error is frequency stability. With current ground-based clocks this error is on the order of one millimeter for objects at a distance of one Astronomical Unit. But these timing synchronization errors have a larger effect on three-way ranging. For example, if two stations clocks are offset by ten nanoseconds, they can produce an error of three meters [121].

Recent advances in DSN methods [121] utilizing Very Long Baseline Interferometry (VLBI) have vastly increased the resolution of one-way signal tracking [132]. The technique is called δ Differential One-way Ranging (DOR) which can achieve as high as 40 nrad resolution in angular position resolution. VLBI techniques utilize simultaneous observations of a signal from multiple sites to determine precisely the angular position of the source. These methods require very high accuracy time synchronization in order to combine the observation data. The use of combining data from spacecraft separated by such great distances allows for greater angular resolution of the observations using interferometry techniques. Delta VLBI introduces a second, very well-known quasar source to compare the original signal to, in order to improve calibration and reduce error. The main limitation is the availability of local strong sources to use as references. An angularly close source with a very well known position is required. Explicit differencing of observations from nearby sources removes or substantially reduces the effects of common observation errors (such as atmospheric noise). This can be used to account for clock and instrumentation timing errors,

which is very important due to the need for the observation times to be synchronized to high precision in order to overlay and integrate the data from multiple ground stations. Additional improvements in error analysis and estimation have increased the accuracy of these positioning systems [33].

The New Horizons mission to Pluto and Charon uses top-of-the-line DSN procedures with δDOR. This mission presents some of the most demanding constraints on radiometric tracking [77]. The mission utilizes δDOR three-way ranging from DSN to provide Earth-based observations of range and range-rate, which integrated over time can give heliocentric orbital parameters. Because Pluto and Charon are at such a great distance and their individual masses are not exactly determined, (though the dynamics give the mass ratio fairly well), it is difficult to ascertain where exactly New Horizons is in relation to the celestial bodies. Observations of the bodies are required to analyze their location to obtain a relative position. Additionally, the optical imagers onboard the spacecraft will be used to image Pluto and Charon and to give the spacecraft orientation during the passes. Over time, as New Horizons nears the bodies, its onboard capability will outperform ground observations of Pluto and Charon, with observations at higher quality than is possible from Earth. The ground-based filters will include optical observation data and radiometric tracking to solve for parameters relating to the physical properties. The tracking data and ephemeris of the spacecraft will further refine the two bodies' orbits, both locally and heliocentrically, as well as provide increased information about their rotation, prime meridian, mass, etc. Due to the large range of mission, when ranging signals are received from Earth, there is more signal than noise. To correct this, the New Horizons engineering team implemented a pseudonoise signal base to allow them to better distinguish the actual data and utilize a series of filters to increase signal quality on retransmission to Earth [25]. This shows a great example of where optical and radio tracking can be combined to form better navigation information and also demonstrates the strength of optimal estimation techniques.

Some current and developing methods utilize Doppler tracking for inter-spacecraft navigation [29] [28] [74]. Such experiments have been performed using the Tracking Data and

Relay Satellites (TDRS) and the Cosmic Background Observer (COBE) satellite [29]. Experiments have also been conducted using the Explorer Platform on the Extreme Ultraviolet Explorer Mission [52] in testing of the TDRS Onboard Navigation System (TONS). The TDRS satellites are used to relay data between ground assets and space assets to allow for always-connected/always-on data connections. This is used for data communication with the International Space Station and Hubble, as well as other highly utilized assets. It was also used extensively during Shuttle operations to enable constant communications.

The satellites have the capability to participate in Doppler and one-way ranging to other assets and to act as a repeater for ground sites. Several experiments in the late 1980s and 1990s studied their application. The TDRS relays forward data from and to the spacecraft acting as intermediaries for ground-based tracking and ranging. True spacecraft to spacecraft tracking and ranging is limited to the accuracy and stability of onboard crystals and oscillators. As such, many of these early studies focused on the application and use of Ultra-Stable Oscillators [29] to reduce the timing uncertainties.

1.4.3 Global Navigation Satellite Systems

Some methods implementing simultaneous ranging solutions have also been developed. The current Global Positioning Systems (GPS) using ranging measurements to multiple satellites to obtain a position fix. This is currently extensively used for air, ground, and sea assets. Additionally, this has been shown to work well for Low Earth Orbit (LEO) satellites. Currently, it is becoming standard practice for satellites to include a GPS receiver. The current GPS system is a constellation of 24 satellites in highly inclined Earth orbits. The design of their orbits is such that at any time, four satellites are in view of any location on the surface of Earth. Several other Global Navigation Satellite Systems (GNSS) are being implemented and put into similar constellations, increasing the available number of assets able to be observed.

The GPS satellites act as a series of beacons that continually transmit specified codes on their predefined frequencies. GPS is an implementation of a ranging positioning method. Upon reception of a signal, a receiver computes a time difference between the GPS satellites

and its internal clock. This information is used to calculate a range to a satellite. Each receiver also tracks the ephemeris of the satellites. Multiple simultaneous range measurements are combined to calculate the receiver's position. Typically four satellites are used with the last measurement included to correct for clock error. As seen before, clock error and stability are very important and drive the accuracy. But for local Earth systems, the accuracy is on the order of meters, which is well within operational-desired capability. Using high fidelity GPS ephemeris (typically post-processed) and advanced error correction techniques (such as carrier-phase analysis), the accuracy of GPS can be increased to 15 centimeters.

The primary downside to GPS is the large number of satellites involved to build a usable constellation. Several studies have looked at building such a network around another planetary body [91] [74]. To calculate a three-dimensional fix to reasonable accuracy requires simultaneous observations from four satellites. Once such a network is in place, though, it is extremely powerful.

1.5 Current Research

To meet the increasing demands on spacecraft positioning and the drive towards autonomous spacecraft, several methods are being investigated [20]. Several different approaches are being taken in this field of research. These techniques include performing Doppler and ranging between spacecraft in local orbits, moving optical navigation analysis methods onboard, and a new method utilizing observations of high energy pulsars.

1.5.1 In-space Radio-based Ranging

As previously mentioned, landing accuracy and knowledge (even with the capability to perform a controlled descent) is limited by planetary entry navigation accuracy. Due to round trip travel time and the time required for ground-based analysis and mission planning to support any last minute orbital corrections, the last navigation command can be sent at eight hours prior to atmospheric entry (for a Martian mission). Thus, initial knowledge and ground-based landing estimates are driven by these errors.

There are two primary aspects to increasing navigation knowledge and capability for

improving landing capability. These are the time required for analysis and the round trip travel times of the Earth-based measuring method. The ideal solution addresses both of these issues. Increasing spacecraft autonomy and computational power can provide a reduction in the time for the last command point. The window for commanding is thus reduced, with the only limit being the time required on the ground to obtain a position fix and the few minutes required to return the navigation information. By utilizing onboard navigation capabilities, the role of humans in the loop is reduced and reduces the responsibility of the ground support crew and transfers it to the spacecraft. Additionally, the software should be very powerful and well verified.

To obtain additional accuracy in navigation knowledge and reduction of response time, additional assets must be used that are closer to the landing site. This requires forward planning and the placement of assets in orbit in preparation for such a landing. In terms of landing on Mars, this has been taken into account via the Electra payload on the Mars Reconnaissance Orbiter (MRO). An example of this method is given in [74].

The Electra payload is a software-defined radio engineering package [44]. The unique features of this device are that it can be reprogrammed over the course of its life to communicate via new protocols and coding schemes, as well as new software-driven functionality. Electra also includes the capability to perform Doppler ranging. As such, it is planned to aid arriving vehicles destined for the surface of Mars, by providing local navigation and relay services. It can also be used to track an object's entry and perform post-landing trajectory analysis. This capability has been tested by tracking the MERs.

This asset also includes an ultra-stable oscillator to allow for increased accuracy in ranging. As such, the spacecraft is able to provide high-accuracy navigation solutions. The primary limiting factor is the reception and transmission power of other spacecraft, which may not have as powerful communications systems. Mission operations were developed to use the Electra payload as a guiding asset for the planetary entry of the Mars Science Laboratory [74] which successfully launched on November 27, 2011.

This was intended to serve as an additional verification of its capabilities and show the benefit for use of local in-orbit navigation-aiding spacecraft. MRO, in acting as a

relay between Earth and the growing number of Martian assets, shows the capability of implementing and investing in a local orbit communication relay satellite. As such, MRO can be considered to be the first node of an inter-planetary communication and navigation network.

However, there are a few drawbacks to this approach. The current Martian planetary exploration plan has been to launch rovers and satellites on alternate two-year periods. This is due to the celestial mechanics aligning in such a way as to minimize travel time between Earth and Mars. For such an active planetary network, implementing an Electra-type payload is very cost-effective and has a very high benefit due to the steady stream of incoming missions. But this is not so for more remote destinations, such as the Juno mission to Jupiter, where such an infrastructure does not exist. Additionally, missions to the outer parts of the solar system are increasingly constrained due to power requirements and the reductions in the solar flux at such distances. As such, these missions currently tend to be one-off missions to explore new areas. Placing the precursor assets in place is as much of a challenge as placing the probe. For some missions, such as the New Horizons mission to Pluto and the Charon, there remains uncertainty about the two objects' orbits about each other, and one of the main science objectives is to track the spacecraft's observed flight to determine planetary properties of the two Kuiper-Belt Objects [77]. When traveling to new locales where there is not a steady manifest of planned missions, it is not cost effective to include specific payloads that occupy weight and volume and do not achieve their true benefit until additional spacecraft arrive. This mass may be much better partitioned for increased fuel, allowing a longer scientific mission, increased development margins, or additional scientific payloads, increasing the knowledge return and overall value of the spacecraft.

1.5.2 Autonomous Optical Navigation

The Electra payload offers a clear benefit for orbital locations with a high density of operating and planned spacecraft as part of a long-term exploration plan. But for independent missions to new areas, where there may not be additional infrastructure available to perform

navigation duties, a powerful option is to use observations of the space environment from the spacecraft and perform the position-fixing onboard. The most plentiful, historically used, and easily observable element of deep space is the celestial background. The positions of the stars are relatively fixed compared to our solar system due to the vast distances separating galaxies within the proper motion of the Milky Way galaxy. Star charts have served sea-faring vessels very well before the times of radio navigation and GPS. Due to the location of the craft in deep space, optical observations can be very successful.

As mentioned previously, optical navigation has been extensively used in deep space exploration on programs such as Mariner and Voyager [24]. An image containing a star background with observations of a celestial object can be used to give the orientation of the spacecraft as well as the distance to the object. This analysis is performed on the ground, due to the complex methods required to identify the star background, identify the planetary body, and perform the needed calculations with the planetary ephemeris. For early spacecraft, with limited memory space and computational power, this analysis was relegated to ground analysis teams, with the spacecraft focused on data collection and transmission.

With the rapid increases in computation power and storage capability onboard spacecraft, it is now possible to move these algorithms onboard the spacecraft with a technique called Autonav [96]. Onboard autonomous navigation systems are desired for spacecraft which operate at large round trip light times, such as New Horizons [8]. This has been implemented with the micro Advanced Stellar Compass and other star cameras to observe planetary bodies against a star background. To enable the algorithms to identify the celestial body and perform the object centroiding and recognition to obtain a measured or apparent size, the spacecraft is pre-programmed to know where over its trajectory to point in which direction. For the recent implementation with Dawn and Deep Space-1 [97], the main observables were asteriods.

As such, the spacecraft stored onboard the ephemeris of the asteroids it expected to see. According to its onboard mission profile, it knew where to look and when to look, in order to determine what object it was observing. This produced an interesting artifact in the

Figure 8: Deep Space 1 Image of Comet Borrelly [10](NASA/JPL)

flight navigation performance due to the selection of targets. Due to the size of the optics required to obtain reasonably scaled images of planets, observations of asteroids needed to be used as opposed to planets [11]. The asteroids have fairly well known ephemeris from Earth-based optical and radar observations. The main difficulty with the process is in the complex algorithms required to centroid the asteroids, which can be quite oblique. An example of an image captured onboard is shown in Figure 8 [10] of Deep Space 1's encounter with the comet Borrelly.

Additionally, even though there have been radar observations of some of these asteroids from Earth, there are still gaps in knowledge of the complete shape. Unless the body is undergoing rotations such that it is entirety visible to Earth ground assets over time, the radar generated capture is limited to the face pointing towards Earth. Additionally, the dynamics and pointing is very different between an Earth-based asset and the spacecraft. As such, parts of the asteroid that may be visible to the spacecraft are not visible to Earth. The optical observations onboard are much more sensitive to lighting conditions, due to the pointing relative to the sun, which radar is imperious to. Due to the complex nature and incomplete information on these asteroids, it is reasonable to expect some errors in the centroiding algorithms, thus producing erroneous navigation fixes. This nuance was

captured in-flight on the Deep Space 1 [10] and Deep Impact [45] mission performance reports [96].

Studies have additionally been performed for navigation at Jupiter [115]. This concept utilized only bearing measurements to capture position relative to the Jovian moons through observation and has been shown to be viable. Additional work has been performed at JPL to advance the algorithms to include precise landmark tracking for landing on celestial bodies such as asteroids [98].

This method has been successfully used in flight and continues to be developed and improved with current work focusing on improving the analysis algorithms and integration [96]. The method does have its limitations, primarily in the required pre-planning of observations of targets, as well as potential gaps in information about the targets being observed. To allow the camera system to capture and image larger planetary bodies at greater distances will require more complex optics, directly affecting the instrument's size and limiting the times of application to when there are available bodies that meet the instrument's viewing capability.

1.5.3 Advanced Deep Space Timing Sources

The development of improved clocks and onboard oscillators is another method under analysis [92]. Current research efforts were recently awarded funding to enable a year-long mission to characterize and flight-test an advanced Mercury ion deep space atomic clock[2]. This research effort focuses on improving the stability of a spacecraft's oscillators. With this capability, spacecraft will be able to perform one-way ranging with ground-assets. Thus, during a navigation pass, instead of requiring a two-way tone transfer, the measurement of phase offset can be performed onboard. This is enabled by the improved stability over the observation. This capability will enable additional methods and realtime onboard processing of the navigation measurements.

[2]http://www.nasa.gov/mission_pages/tdm/clock/index.html

1.5.4 X-Ray-Based Pulsar Navigation

Other approaches exist to navigate using celestial observations. Instead of utilizing measurements in the radio or optical bands, X-Ray Navigation (XNAV) observes in a higher energy band of the electromagnetic spectrum. The main sources used for XNAV are rapidly rotating, highly magnetized neutron stars [30]. These sources produce regular oscillating emissions with very stable frequencies. It has also been suggested using these for timing standards and atomic clocks due to this regularity. Alternate observations of these sources could be in the radio. In order to focus enough signal a very large dish or very high gain and low noise amplifier would be required.

Instead, the focus is on the X-ray emissions. This is desirable due to the low noise in this part of the spectrum and the relatively small sensor size. The trade-off is that very long integration times are required to gain sufficient signal to noise ratios. Also, complex statistical processes are required to fold the data from multiple observations to capture the phase and frequency of the observation [105]. Initial results show that a square meter detector with observations integrated over a thousand seconds can provide an accuracy of one kilometer [105]. The time required and detector size are inversely related, as one increases the other decreases to maintain a set accuracy.

Studies have analyzed the performance of XNAV for attitude [57], timing [105], absolute position [106], and relative position [30]. By measuring the observed position relative to background stars, it is possible to determine attitude, similar to a star tracker. Using the High Energy Astrophysics Observatory (HEAO), XNAV techniques provided a .012 degree attitude uncertainty in spacecraft roll [57]. Studies were also performed with the Advanced Research and Global Observation Satellite Unconventional Stellar Aspect high energy detector. Sheikh [106] was able to demonstrate calculation of orbital position from observation data and obtained an error of two kilometers. Errors in the GPS onboard solution (which was being used as the true inertial position) increased the calculated error to 15 kilometers [105].

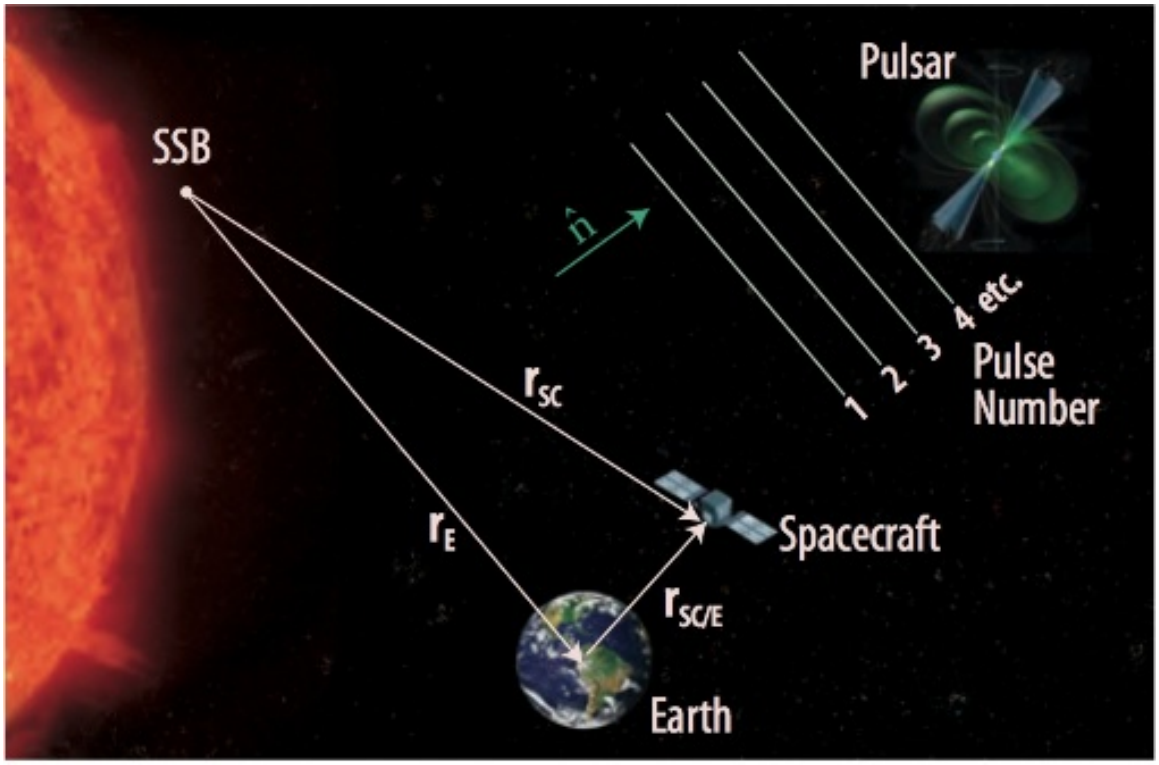


Figure 9: XNAV Positioning Concept (NASA)

Position estimation using XNAV [54] is shown in Figure 9[3]. In order to perform positioning, a highly detailed pulsar timing model is required to determine the time of arrival of pulses from a pulsar at the solar system barycenter (labeled SSB). By observing a series of pulses, it is possible to capture the difference in measured arrival time and modeling arrival time at the barycenter. This information is then used to determine a relative location, using pulsar angular position [105].

By observing multiple sources, it is possible to ascertain full three-dimensional positioning [54]. The performance is limited by timing accuracy onboard, which affects the binning capability of long integrations, attitude pointing, which maximizes SNR of an observed source, and gravitational modeling accuracy. This is important as the trajectory is propagated via optimal estimation techniques to determine the position-fixing over the course of the long integration [106].

[3]http://gcd.larc.nasa.gov/projects/deep-space-x-ray-navigation-and-communication/

The limitations to these methods are the required detector sizes coupled with long integration times. The spacecraft launch volume limits the available detector size, but as detector technology continues to improve, it will be possible to use smaller instruments. For long integration times, complex gravitational models are required in estimating the spacecraft's trajectory. This also limits the applicability of this method to certain parts of the trajectory with well-defined gravitational properties. Applications of XNAV that are unaffected by these parameters have been discussed. For example, Sheikh suggests developing beacon satellites that are in very stable orbits and can perform long integrations to track their position to a very high degree of accuracy and also to act as pulsar timing models. These satellites could be used to operate as timing and position references [105]. The procedures to support XNAV are in active development and are documented in [99] as being a promising future development path for deep space navigation. A large focus of this work is on detector development, building pulsar libraries, and developing optimal estimation algorithms, working towards eventual mission implementation and flight qualification.

1.6 Comparison of Current Methods

The previous sections provided a summary of the current navigation methods and ongoing research. It is important to compare and contrast the strengths and weaknesses of each to gain some higher level insight to the implemented approaches. The methods can be broken down into the two categories as put forward by Groves [55], positioning and tracking. Positioning is the spacecraft's internal determination of its location by observation of some physical entity, such as optical observations of stars or acceleration. In contrast to positioning, tracking is the use of external measurement assets, such as DSN, to determine the object's position.

Table 3 breaks down how the methods fit into each category. Even though optical navigation is based on observations of the spacecraft's environment, it is listed as an external measurement due to the requirement of ground-based analysis. Similarly, due to its autonomous nature, Autonav is considered an internal method. DSN tracking, inter-spacecraft tracking (both TDRS and Electra), are all considered external methods. X-ray

Table 3: Overview of Navigation Methods

Method	Type	Observation
Optical Navigation	External	optical imaging of celestial bodies against a star background analyzed on ground
AutoNav	Internal	optical imaging of celestial bodies processed onboard
DSN Tracking/Electra	External	round trip light travel time to spacecraft and frequency shift of received signal
TONS	External	ranging and Doppler measurements relayed through TDRS
X-ray Navigation	Internal	time of arrival of radiation pulse from pulsars
GPS	Internal	range relative to multiple GPS satellites by measuring dt from GPS broadcast signals
Dead Reckoning	Internal	angular velocity and acceleration integrated to propagate position

Navigation, dead reckoning, and GPS systems are internal methods, due to the spacecraft itself performing the position fixing.

Each of these navigation systems has their own unique advantages and disadvantages. The characteristics of each design also provide insight into the overall requirements and needs for a navigation system. Additionally, they give direction into how to design a new navigation system. Optical navigation (including Autonav) is the use of observations to provide for navigation fixes. Observation of the environment is important in that it is independent of Earth assets. The drawback to this method is incomplete information about the sources used. There is inherent uncertainty in a planet's ephemeris, and in some planet's physical properties. Strict observation rules and defined assumptions must be present when correlating flight observation data to previously known values. Autonav proved to be beneficial in cases where exact ephemeris may not be known and local onboard guidance must be used to navigate relative to a target, but also demonstrated the effects on navigation with incomplete object data. These navigation techniques are also very beneficial in their use

of onboard scientific instruments to perform the observations, taking advantage of dual-use payloads.

Similarly X-ray navigation takes this same approach, but focuses on high energy particles. X-rays were chosen to take advantage of new advances in high energy detectors. In comparison to planetary observations, there is a much more spread out pattern of X-ray pulsars, whose observation is independent of location in the solar system. Yet similar to optical observations, this method begins to break down with assumptions in the modeling, particularly of the signal's arrival times at the barycenter. The pulsars themselves also present some problems. Even though their emissions have proven to be highly repeatable, it is still possible for random outbursts or other events to change the spectrum or phase of the emitted signals. As such, they must be tracked and models updated over time. This is also pointing dependent, and the spacecraft must have highly accurate attitude knowledge to be able to point at a particular source. Long-term pointing capability is important to maintain observations during long data collection times to obtain sufficient accuracy.

Radiometric tracking has proven to be very successful and very capable for deep space navigation. Many recent spacecraft have chosen to use purely DSN for navigation purpose, eschewing traditional optical observations, to minimize development cost and time [121]. This works particularly well for missions with very well understood dynamics, such as Earth-Mars transfers, due to the existing highly detailed Martian observations. The high resolution of δDOR observations can be used to solve for force models and to understand more information about planetary physical properties. For example, this will be used for the New Horizons mission to link observed ephemeris of the spacecraft to optical observations in order to better understand the orbit and mass distribution in the Pluto-Charon system [77].

The primary drawback to this method is the reduction in accuracy and noise in returned signals with distance from Earth. Additionally, observations can be limited or occluded by other objects between Earth and the spacecraft, such as the sun or other planets. The Electra radio addresses this limitation by placing radiometric tracking assets farther out into the solar system. This allows for Doppler and ranging and bounds error as a function

of distance from the closest asset. The primary bounds on tracking are due to the geometry and transmission distances.

Spacecraft volume constraints also limit the size and capability of the radio equipment, primarily the antenna dish. Deployable antennas for radar use have been used on orbit with sizes from 4.9 to 12 meters [62]. Current research into deployable antennas is working towards a 35 meter diameter Ka-band dish[62]. Operational power limitations in deep space also reduce the amount of power available for transmission to assets. Additionally, the spacecraft's oscillator stability directly affects the accuracy of the method and will increase errors with tracking distance. As such, the Electra radio provides an expansion of radiometric tracking farther out, but its operation and performance tie directly into the spacecraft's requirements and design trades made. TDRS inter-spacecraft tracking follows the same analysis. These spacecraft operating as deep space radiometric tracking hubs must have onboard timing sources to maintain this capability, which affects other subsystem designs.

GPS operates as a mix between external and internal. GPS receivers observe the timing signals sent out by the satellites in the constellation, and the measurement of location is performed relative to these sources. As such, the transmissions are artificially creating an observable, that are used similar to planets in optical navigation. This method shows that it is possible to accurately and repeatedly perform position-fixing at high rates with high accuracy by tracking these broadcast signals. The drawbacks to this method are that the satellites must be very well synchronized and a large body of knowledge about them (such as ephemeris) must be tracked, updated, and broadcast to all elements in the network to maintain its accuracy. Additionally, building out the constellation requires a large investment to design, build, launch, and maintain the large number of satellites required. They must utilize very accurate onboard clocks and oscillators to maintain the integrity of their transmissions. This affects both operational support and lifetime, as well as subsystem trades early in the design process.

These methods encompass multiple development paths and options to implement navigation systems. It is important to capture the key features that make each unique. The

observational techniques utilize the background environment to perform navigation. This is important in that it requires no external or special signals generated to perform this operation. The fundamental requirement to enable observational navigation is detailed knowledge of the source. This refers to its position and physical characteristics. Libraries must be developed to capture this information for the sources, whether they are celestial or planetary.

External navigation tracking signals have been used to achieve very high accuracy state tracking, though at the cost of very complex analysis and ground support. There has been work to automate these processes and place them on orbit, but the main limitations are on spacecraft requirements and sizing constraints. The location of the transmission source must be well known and a priori knowledge about the target available in order to successfully track. As such, regardless of the method used, some assumptions must be made to provide a navigation fix, but that information varies depending on the method used. In all of the methods, these assumptions affect the state estimation error as much as the measurement sources.

1.7 Summary of Current Methods

This chapter has presented the need for deep space navigation. It is crucial to continue to develop and advance the capabilities in order to support autonomous vehicle operations, complex missions that require real-time navigation support, and improve the capability to land planetary asset to maximize scientific return. With the implementation of extraterrestrial habitats and human exploration beyond low Earth orbit, the need for advanced autonomous navigation techniques is increasingly important to support both in-space maneuvering and trajectory planning. Additionally, the need to support high accuracy pinpoint landing is important not just for scientific return bu also to support a sustained human presence by delivering supplies directly where they are needed. The various approaches have been summarized, and provide a variety of methods to develop deep space navigation systems. Each of the proposed paths either depend on complex hardware development and testing or ground-based orbit determination efforts and human feedback to ensure continued

operation.

CHAPTER II

NETWORK-BASED NAVIGATION (NNAV)

2.1 Confluence of Navigation and Communication

As shown in the previous chapter, current proven methods of deep space navigation rely upon ground updates and support. The advanced computational flexibility coupled with the ever-expanding data storage capability of ground assets enables a greater level of vehicle orbit determination. Additionally, performing this analysis using ground assets and personnel also enables debugging and continual improvement of analysis routines much more easily than going through the required validation, testing, and uploading required for on-orbit software updates. As such, communication with Earth is crucial to deep space navigation and time tracking, both for maintaining contact and for performing the measurements required.

The enabling data for current methods involve the observation of change in phase between the transmission and reception using a predefined tone signal or series of tones. This allows for a measurement of range to the spacecraft. The two-way range is thus measured. This is used rather than a one-way ranging signal (from Earth to the spacecraft or vice versa due to the limited stability capability of spacecraft oscillators). The two-way measurements allow for a reduction in measurement range by averaging two observations. Recent advances in spacecraft atomic clock design such as the Deep Space Atomic Clock [92] [1] will allow for the additional use of one-way ranging signals. The ranging carrier tone can also potentially be mixed with the data ranging signal. As such, the capabilities of ranging and communication are inherently linked to ground-based measurements. These measurements also drive the requirements for post-processing, verification, and generation of state updates to the spacecraft. The inherent delay between measurement and state update is due to human-in-the-loop error checking, tweaks to gravity and dynamics models, as well as advanced filtering routines running over a very large set of data.

[1] http://www.nasa.gov/mission_pages/tdm/clock/index.html

For current methods, limitations include the amount of available time on the ground stations, as well as the cost to support ground operations. As additional spacecraft utilize the existing ground work, the communication infrastructure will become increasingly constrained due to increased users. Along with increased data fidelity, this will limit available time to each asset to gather the required observations. There is some initial work done to increase the frequency of communications to Ka-band [119] as well as optical communications. The move to these frequencies will enable greater bandwidth at the cost of greater power systems and receiver systems (due to increased space loss at higher frequencies, insert space loss equation below). Due to the fact that ground-based measurements are only possible when communication is also possible demonstrates the intrinsic link between communication and navigation. This will be discussed below in addition to the opportunities it enables for advanced autonomous deep space navigation.

2.2 Communication Architecture Research

As more spacecraft are launched with greater reliability and improved scientific instruments return higher resolution data of a greater quantity, the communication system bandwidth begins to become constrained. In order to meet the growing data requirements from individual spacecraft and the overall increase in network traffic to deep space, the current communication system must be advanced in order to meet this growing demand. There are several ways to address this. A simple approach is to develop data compression schemes to reduce total data transmitted. But there is a limit to the effectiveness of this, and currently methods are fairly mature. A more productive method is switching to higher frequency standards in order to operate at increased data rates. This has been implemented with the switch from early S-Band to X-band and currently to Ka-Band communication standards which allowed for increased bandwidth. These higher frequencies required advancements in radio processing, receivers, and transmitters both on the ground and in space.

To achieve higher data rates by frequency alone requires a fundamental shift in the underlying physics. This change is from transmission and reception of electromagnetic waves to generation and detection of photons generated by an optical source. This method

is known as optical communication and has been in development as an alternative to radio communication [58] [12]. Much work has been done in this area to improve data rates, transmission power, and detection sensors. Recently, the NASA Office of Chief Technologist Technology Demonstration Mission program funded a proposal to fly a next generation optical communications system [116]. Additionally, the NASA Lunar Environment and Dust Environment Explorer (LADEE) satellite, with a planned launch in 2013, includes the Lunar Laser Communications demonstration[2]. This payload will implement a laser communications terminal to demonstrate high bandwidth Lunar orbit to Earth surface communications. The drawback to such a well-performing receiver is the requirement of new satellite hardware, as well as ground infrastructure. In terms of the communication method, development is reaching a point where new spacecraft or hardware may begin to be built into or replace some of the existing infrastructure. For a built-in navigation system, it would be preferable to be independent of transmission medium in order to be applicable across any communication method.

A different approach is to build out a network architecture into the solar system, breaking down the network into a series of local networks and utilizing relays with high bandwidth data trunks [18]. However, this is not the ideal solution for small networks with a limited number of assets. This is due to increased complexity of designing and operating an additional relay satellite. Coupled with a limited lifetime and power limitations on transmissions, it quickly becomes a very complex problem. This is in comparison to putting large communications receivers on Earth, which has ample power and space and manpower to operate it and very stable clock sources.

But as the number of space assets in operations continue to grow and a growing extraterrestrial surface presence is initiated, the network architecture will increase and mature along with it to support full operations and data transfer between networks. These relays are important due to severe power and size limitations for any surface exploration system.

An operational example of this is the Mars Reconnaissance Observer (MRO). This spacecraft contains a very powerful transmission capability utilizing powerful Ka-band and

[2] http://www.ll.mit.edu/news/lunarlasercomm.html

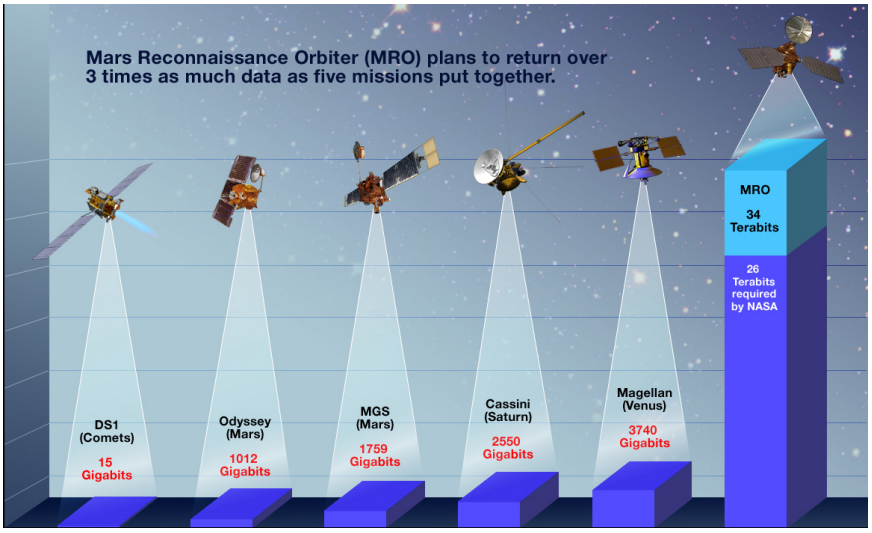


Figure 10: MRO Data Return (NASA/JPL-CalTech) [119]

X-band transmitters. Figure 10 shows the vast increase in data capability enabled. One of the satellite's missions is to operate as a relay for the Mars Exploration Rovers and other Mars surface assets. All of the MER's data is forwarded from the surface of Mars to in-orbit relays and from there to Earth. The relay concept has also been used for the Phoenix lander through use of the Proximity-1 protocol [118]. The Promixity-1 protocol is defined by the Consultative Committee for Space Data Systems CCSDS, an international standards body for space transfer protocols focusing on point-to-point networks. This protocol is implemented on multiple Mars orbiters that act as relays, including MRO [119].

Another example of implementing a dedicated Earth-Mars high bandwidth data trunk is the Mars Telecommunications Observer (MTO) [44]. This satellite was planned for a 2009 launch and 2010 Mars arrival with over 10 years of operations including extended missions. Its primary operational goal was to act as a dedicated communications relay in Martian orbit. Primary payloads included the Electra radio, with a Proximity-1 implementation, as well as a laser communications package to demonstrate optical communications from Mars. Given the communication system, it was expected to be capable of delivering 10 gigabits of data per day for an 8-hour DSN contact [44]. Autonav capabilities were also being planned for implementation for on-orbit technology demonstration. The mission, though, was canceled in 2005, due to funding constraints within NASA's budget to support other

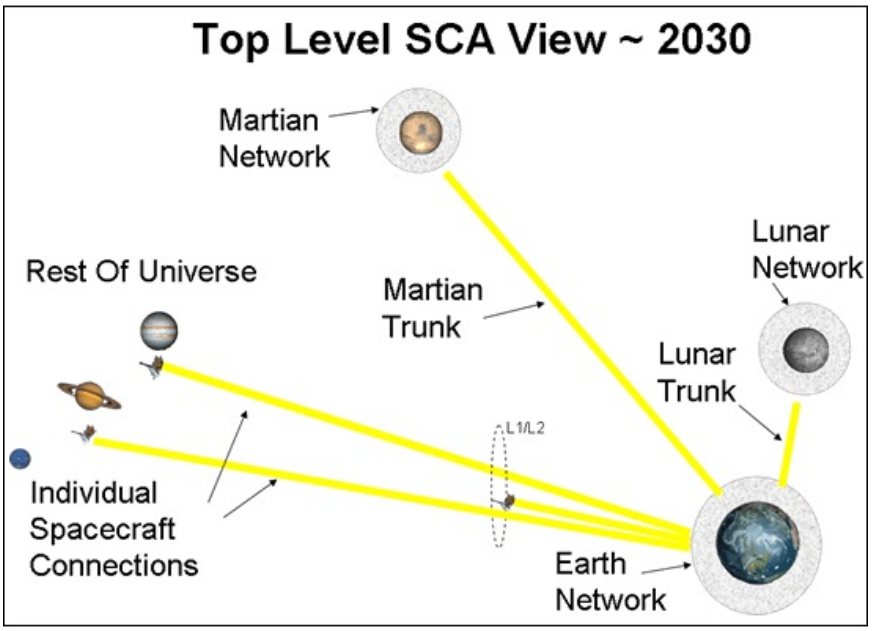


Figure 11: SCAWG Space Communication Architecture [103](NASA)

programs[3].

Additional studies have been conducted on expanding the assets in Martian orbit into a Mars Network for data communications amongst assets and to Earth [14] and using these sources for navigation [74]. As more surface assets are utilized, the need for a dedicated data trunk between local orbit and Earth will increase. The use of relays for local communications and rovers has been well proven and will continue to be an aspect our deep space communication architecture.

This can be seen in the current plans for the development and immediate growth of the current space communications architecture. The NASA Space Communications Architecture Working Group developed a set of heuristics to develop a growth plan to meet data return requirements and frequency constraints in the immediate timeframe [103]. The resulting architecture is given in Figure 11 [103] and displays a combination of direct satellite communications combined with data relays and trunk links as they become cost-effective to handle increasing data traffic. The architecture thus provide pathways for additional data relays as part of an evolving and growing infrastructure.

[3]http://www.spaceref.com/news/viewpr.html?pid=17424

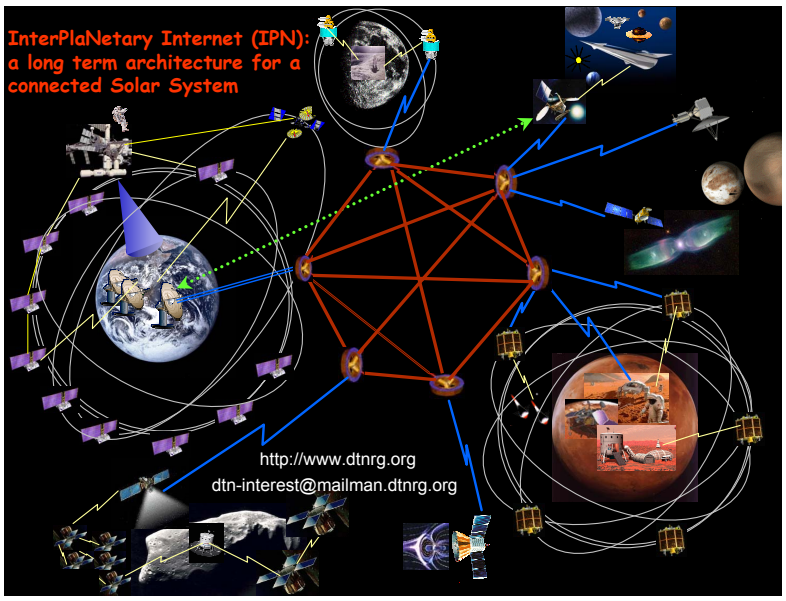


Figure 12: Interplanetary Internet Concept Architecture[15]

2.2.1 Delay Tolerant Networking

The concept of data relays is the base of the Interplanetary Internet (IPN) concept, which is shown in Figure 12 [15]. The IPN studies focused on how to implement a Internet-like approach to networking for space systems. In order to achieve networking in space, a new data transfer protocol is required in order to take into the long light travel times between nodes. Using TCP/IP data protocols with such large delays causes timeout errors. Most ground system network protocols assume near-instantaneous data transfer capability. But this amount of time is barely long enough to enable communication with the moon [35]. This inspired the creation of Delay and Disruption-Tolerant Networking (DTN), protocols, such as the bundle protocol and Licklider Transmission Protocol (LTP).

Farrel and Cahill give a thorough overview of the protocols and results from ground testing in their book [35]. The bundle protocol is designed for a large network of intercommunicating assets. As the nodes communicate, bundles are passed from node to node until it reaches its final destination. A bundle is defined as a closed group of data that combine both reported data and status. Additionally, a bundle can include a series of commands

and settings meant for a scientific payload or rover. The progress of the bundles is tracked through the network, as is the ownership of the packet, to insure it has reached its destination before it is cleared from the source's memory. The protocol is also developed to optimize retransmissions due to lost pieces of the bundles and efficiently only retransmit what is needed, reducing redundant data transfers and allowing for increased throughput.

The bundle protocol proves incredibly useful in integrating data from several distant sources such as commanding and receiving data from a scientist on Earth to a Mars rover, which goes through several hops through relays and ground processing networks. DTN methods have also been implemented on the Commerical Generic Bioprocessing Apparatus payload on the International Space Station [81][51] in processing data from orbit to primary investigators on the ground.

Similarly, this work has also led to new developments in spacecraft point-to-point transmission protocols. LTP is an implementation of delay tolerant networking for point-to-point communication for long travel time deep space links. This protocol is developed in contrast to standard CCSDS file delivery protocols which are typically implemented mission to mission. LTP was developed to be a standard development of a transfer protocol implementing many of the same functions as CCSDS's file delivery protocols. The main concept of LTP is the use of freezing timeout timers, when a spacecraft knows that it cannot receive data due to planetary eclipses for example. This reduces unnecessary timeouts and retransmits of data. Additionally, each bundle is broken down into a series of individual packets based on max transmission size. Each piece of data can be prioritized as to reliability desired, to ensure the important information is transferred whereas less important, potentially redundant data is not transferred or retransmitted. To reduce retransmits, upon reception of the end-of-message packet, the receive station tracks which packets were successfully received and transmits that data back to request any missing data. It is also possible to add headers to these messages to contain information such as a digital signature and host information. This user-defined header, and also bundles, provide a great vehicle to transfer additional information to be used in a navigation system. Several implementations of LTP are currently under development. One project is the Interplanetary Overlay Network being

developed by NASA, John Hopkins University Applied Physics Laboratory, and Ohio State University which includes the bundle protocol, LTP, and other space standards in an open source package[4].

The combination of LTP and DTN protocols enable a very efficient method of transferring data at large distances, increasing data throughput my minimizing redundant transmissions. These become especially useful as the deep space communication network continues to grow and additional surface and space assets are integrated into the network. There has been ongoing development and growing flight experience in deep space, such as EPOXI [67] and the Deep Impact Network Experiments [104]. Experiments were also performed with the Disaster Monitoring Constellation satellite sensor network utilizing the bundle protocol to download images from orbit [64]. These protocols have proved their value in point-to-point communications as well, increasing efficiency, and provide a communication standard that can be implemented across transmission mediums to effectively transfer data. The inherent structure of the bundles (and breaking down the data sets into transmittable chunks) provides a great asset and potential for the standard integration of additional information to support space operations, such as dynamic routing network information or navigation aids.

2.3 Possible Paths Forward

To develop a new navigation to meet the needs of current and future missions, which will require increased autonomy and increased accuracy, one can identify several options for forward development in terms of the current research. For optical navigation, there are multiple paths forward including improvement and use of variable optics to improve the range and accuracy. Additional standardization of processing algorithms will also lead to improved results and increased implementation of Autonav. X-ray navigation can also be improved by advanced model development and sensor technology upgrades.

Improvements to DSN Earth-based tracking include additional assets to increase capacity and throughput. Additional research into VLBI techniques utilizing large arrays of

[4]https://ion.ocp.ohiou.edu/

smaller satellites also shows promise [65]. The Electra payload can provide improvements in several aspects as mentioned above.

There have been several studies looking at implementing GPS constellations around other celestial bodies such as the Marco Polo Martian navigation system [91]. The main drawback to this is the large investment and long development time to implement such an infrastructure. Other studies are looking at building a Navigation Receiver [131], to enable satellites in High Earth Orbit (HEO) to better observe the GPS satellites and obtain a fix. Current GPS implementations are evolving to include transmission channels from the receivers as well as transmission for use during emergencies such as Search and Rescue[5]. These advances though are limited to improvements in local orbit, and are not directly applicable to deep space navigation.

Due to these options, the most robust algorithms operate in a middle ground between observation and tracking. Being observationally-focused is important to allow for inherent onboard spacecraft navigation regardless of external signals. But the incorporation of an external source allows for highly sensitive information about information such as ephemeris, which can be used in the development of position-fixing algorithms. The main initial disadvantage of radio tracking payloads, such as Electra, and a planetary GPS system, is their reliance on initial infrastructure. GPS has also shown that a very limited data set, ephemeris and time, can be used by multiple observations to attain an position fix (range method). Additionally, optical navigation techniques have proven that an observation and an attitude can also give a fix (range and bearing method). These show the capability of using each method.

A potential path forward is to combine the two methods, in that the spacecraft is observing a signal both in terms of the signal and its source, by embedding additional information. Ideally a signal used for navigation is an inherent observable deep space. In terms of potential signals to use, one that has become prevalent in our continuing operations is communication and data transmissions to spacecraft. For the foreseeable future, communication with spacecraft will be via means of electromagnetic waves, whether radio, X-ray, or

[5]http://www.gsa.europa.eu/go/home/galileo/applications/

photonic. The planned advancements and current research into the deep space communication architecture provide an even more prevalent communication signal that can be utilized for positioning in a range and bearing approach. This can combine the advantages of an observational and tracking-based navigation system.

2.4 Proposed Navigation Approach

As opposed to ground station infrastructure updates and technology investments in communications subsystems, this thesis proposes a different solution. Similar to the ranging tones currently in use during communication, this method considers the integration of navigation measurements within the information transmitted via the communication data structure. As opposed to processing the measurements on the ground, the data is processed onboard the spacecraft, enabling autonomous updates of state during a communication pass. The operation is considered to be autonomous to the state estimation being performed onboard the spacecraft, as opposed to ground-based orbit determination techniques, which provide a state update to the vehicle. The continual improvement in microprocessor computational capability allows for this shift of analysis from large ground-based computing resources to onboard processing.

The expanding sources for positioning will be the relays and spacecraft that will form the InterPlanetary Internet and will integrate into the digital signals from a growing communication infrastructure. The main assumption that will be analyzed as part of this work, is the existence of a growing network of satellites operating as relays, fulfilling roles similar to MRO. The size of the navigation network directly drives its overall performance. This is due to the increased distribution of signal and measurement sources. The driving aspect behind network growth is the assumption that as the total number of assets in operation (such as on or around Mars and on the Moon) saturates to a point in which it is no longer viable to use only Earth-to-asset communications, a growing network will be implemented. Along with the discussed studies, this is accomplished by the integration of a new network relay node spacecraft. The method is applicable to cases involving a minimal number of relays and the capabilities of point-to-point navigation will also be studied. A minor assumption is the

implementation of an LTP-type protocol (or DTN bundle protocol) to increase bandwidth and allow for increased efficiency in communication between assets, which is standardized and implemented across all assets in space.

The proposed concept of NNAV is to embed headers containing navigation information into the communication packets. This information coupled with onboard state estimation of other spacecraft and planetary bodies can be integrated to perform range, position, velocity, and ranging measurements. Due to the high rate of communication packet transmissions, a large number of navigation packets can be integrated to allow for onboard estimation of the navigation state with multiple measurements. Additionally, it is possible for the spacecraft to intercept other navigation packets by observing any communication signals. This capability is directly limited by orbit geometry and signal reception strength. A concept of operations is shown in Figure 13.

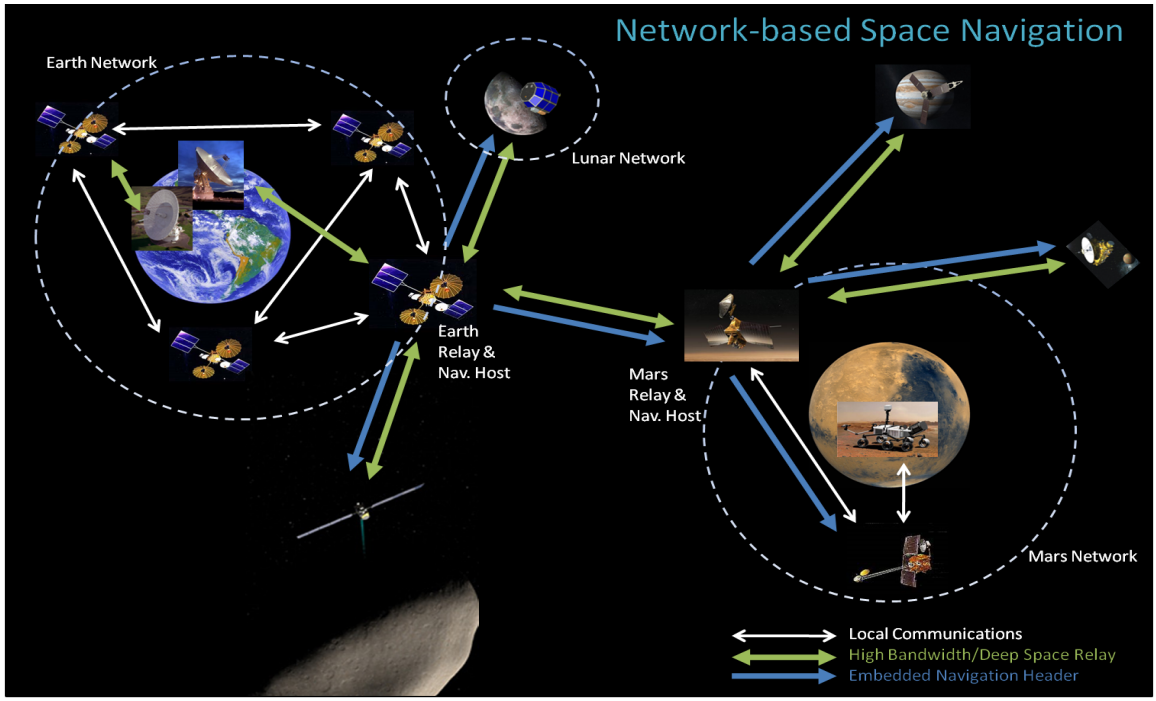


Figure 13: NNAV Concept of Operations

NNAV is an implementation of a multiple ranging method of positioning. By embedding position and timing information into transmitted navigation packets, each relay or ground station essentially doubles as a navigation beacon similar to GPS satellites. The main

difference is that instead of using four independent instantaneous measurements to perform a position fix, repeated observation of range from multiple satellites is used for NNAV. Multiple observations will increase the performance, and this capability grows with the expansions of navigation assets into the solar system. For example, as a craft approaches Mars it may reorient to acquire positioning from a Mars relay, such as MRO, in addition to an Earth orbit-based relay or ground asset.

There are several advantages to embedding navigation packets and performing positioning automatically whenever an asset is in communication. As a spacecraft's orbit becomes increasingly well-defined, supporting navigation methods, such as DSN tracking, become secondary. The DSN assets are then primarily used to debug or in safe mode situations. This frees the Earth-tracking assets to focus on initial Earth-based trajectory characterization, and communication, increasing communication bandwidth and simplifying operations. Additionally, the need for Earth-based navigation analysis is reduced and those facilities can focus on other development and operations. These assets can then be fully utilized as high bandwidth Earth data relays.

The other major component of NNAV is the development and determination of what data to include in the navigation header. One of the main thrusts of this work is to determine the optimal data set that needs to be propagated, in order to minimize positioning error. The basic packet simply includes the inertial position and its time of transmission or simply a transmission location identifier and a time. Combining this with the spacecraft's onboard ephemeris estimates can provide a complete measurement of state. The other information comes from the spacecraft's onboard clock. Assuming the speed of light in a vacuum[6], the difference between the transmission time and reception time can be calculated to ascertain the range. This range calculation is similar to that performed by GPS receivers in calculating psuedorange information. Additionally, the stochastic behavior of the clock must be modeled and integrated into the analysis methods. Together, this information forms the base set that can provide a position fix.

[6]All designed methods of communications involve electromagnetic radiation traveling at the speed of light, just as different frequencies.

Additional information can be integrated to reduce errors and increase accuracy. This information can include calculated error terms or updated ephemeris of the source's trajectory through inertial space. Information on the navigation beacon's attitude can also be transmitted to help bound errors. A central goal of this analysis is to analyze the packet structure to ascertain the ideal header to minimize error, while also taking into account the bandwidth required. Over the course of a conversation between two assets, the traveling asset can also broadcasting its calculated position back to the navigation host. As this information is updated and the two sites continue to generate fixes in relation to each other, the two can converge to a nominal solution. As the two communicate, other data such as time updates and corrections can be inserted into the packet.

2.5 Integration with Current Protocols

This proposed method requires integration with the data transmission protocols in order to meet international space data standards [41]. These standards of both transmission protocols and data packaging are controlled and published by the CCSDS. The Proximity-1 protocol, as mentioned above, serves as a definition of the data packaging, signal generation, and processing for close proximity space data links [40]. This allows for common definition of how to implement and utilize these types of communication links to allow for common operation across multiple vehicles and mission, enhancing inter-operability. This protocol focuses on the how of the data transmission and the structure of the overall packet to allow for tracking and error checking. There also exist several standards for packing the data to put into these transmissions.

The primary international standard is called the Space Packet Protocol [42]. This captures the basic unit of data transfer between assets and is embedded within other data transmission protocols. It consists of a primary header, a potential secondary header, and the user data field. An overview of the Space Packet is given in Table 4 The primary header content is described in Table 5. This is the required data that must be included in every Space Packet in order to be properly processed and adhere to the standards. The total header has a length of sixteen bits and includes a diverse data set: the protocol version,

Table 4: Space Packet Structure

Packet Content		Byte Length
Packet Primary Header		6
Packet Data Field	Packet Secondary Header	0 to 65536
	User Data Field	65536 - Secondary Header Length

Table 5: Space Packet Primary Header

Data Type		Number of Bits
Packet Version		3
Packet ID	Packet Type	1
	Sec. Header Flag	1
	Application Process ID	11
Packet Sequence Control	Sequence Flag	2
	Packet Sequence Count	14
Packet Data Length		16

the packet type, whether there is a secondary header, the identification of the application process, packet sequence flag and count, and number of bits in the total packet. This information allows for successful parsing and decomposition of the transmitted data.

The NNAV approach takes advantage of the capability of the standard protocols to includes additional header information. To enable interoperability with the standard Space Packet, this concept utilizes the secondary header information to include the require navigation data content. Using already defined pieces of the communication protocols fosters integration across multiple missions and vehicles by simplifying method implementation. Additionally, integration into the standard protocols reduces additional design work. Upon implementation, the methods must be thoroughly tested to ascertain data processing latency, to support the navigation measurements.

2.6 Benefits of NNAV

These methods provide many advantages in terms of its lifecycle engineering aspects. If pursuing a new method of navigation, it is desirable to consider the total lifecycle of the implementation, from initial development to upgrades and maintenance. This is to ensure that the method is approached in a manner to allow for long-term use.

An initial aspect of this relates to the difficulty of implementation. In order to gain wide acceptance and use, as is required to fully grow the network and increase the benefit of such a navigation system, it must be easily integrated into existing software and communication infrastructure. This method is inherently robust, as it is digital and operates independent of communication medium. The only upgrade required for existing spacecraft is a software update to enable processing of the header and the algorithms required to acquire a position fix. The only potential limit to implementation is the amount of processing power available, which may be constrained by onboard programming space, computational speed, and power limitations. As given, it does not require any fundamental hardware upgrades to be implemented for existing craft, and is independent of communication method. This demonstrates the practicality of implementation of this approach.

It is also important to develop an approach that can grow and improve over time. This insures that the initial investment in the algorithms and infrastructure developed (i.e., the communication relays) provides maximum return, achieves continued performance gains over time and grows with subsystem technology. As mentioned above, each method of navigation has several paths forward and a majority of these technology development programs can also be applied to NNAV by taking advantage of improving communication technologies or integrating additional measurement sources. Additionally, as advances are made in spacecraft subsystems, such as clock stability, transmission and reception performance, and computational capability, the proposed architecture can expand and integrate to take advantage of the additional resources, by using more numerically intensive algorithms, increasingly complex state estimation algorithms, and improved measurement integration and processing.

2.7 Expected Navigation Capabilities

NNAV integrates and reinforces the current research focused on the improvement of deep space navigation, enabling increased spacecraft autonomoy. This proposed architecture can be used in coordination with any of the other navigation systems both in use and in development. As the communication network evolves and new spacecraft are integrated into the

network, integration of the data headers into the transmission data can be easily attained through software development and standards. As the specific communication methods develop, the packets can continue to be inserted into the communication packets, independent of transmission medium. Additionally, as new measurement approaches are implemented and tested, NNAV will allow for further autonomous navigation integration and augmentation. From the description and concept of operations, several hypothesis, identified by (H), can be developed:

> **(H1)** NNAV, which utilizes embedded navigation packets to enable state updates simultaneously with communication, is a viable method of deep space autonomous navigation.

In order for the navigation system to be deemed viable, it must demonstrate that performance can be gained by the use of these packets to justify their cost in terms of data transmission requirements. Additionally, the methods must show some verifiable enhancement to current approaches to gain a foothold in further research and at a minimum show benefit compared to direct propagation of state. This thesis proposes that the integration of navigation packets into the communication protocol will enable autonomous navigation, by allowing the spacecraft to perform in-space state updates independent of Earth. The development of onboard state estimation algorithms and transmission of navigation packets will enable this capability. Additionally, this is viewed as a simple addition to current communication protocols through the use of standard header formatting.

> **(H2)**Augmenting traditional navigation state update techniques with NNAV will provide improved onboard state estimation capability for deep space missions, especially with a limited network implementation.

As the NNAV architecture is implemented across the growing communication network, all assets utilizing NNAV will benefit from the increased number of measurement sources. For early implementations though with a limited number of communicating assets, the maximum capability of NNAV is as an augmentation of traditional navigation methods.

By integrating with other navigation methods early in its implementation, its capability can be demonstrated and can allow for additional mission navigation redundancy. Through the embedding of navigation packets into the communications, the spacecraft can continue to correct its navigation state providing increased frequency of navigation corrections and updates, minimizing state estimation errors.

> **(H3)** NNAV will reduce the reliance on ground-based state updates and limits the growth of navigation errors between updates.

By the integration of navigation packets with a limited data set, the frequency of ground-based state updates can be reduced from current levels of once a week. Even though the packets may not contain a full state update, their increased frequency of navigation measurements provide a bounding effect on state estimation errors. This is due to daily communication and multiple measurements during a single communication pass. As the NNAV algorithms are demonstrated on-orbit and further refined and improved, their capability will increased. Due to the increase in navigation information and spacecraft state estimation capability, it will possible to increase the time between ground-based state updates and still maintain navigation accuracy.

2.8 Research Question Development

In order to evaluate the navigation method against the declared hypotheses, analysis must be performed to capture the performance of NNAV in relation to current navigation methods. Each of these hypotheses can be linked due to a specific analysis that must be performed. To capture overall performance, an analysis and testing environment must be developed to measure navigation capability. This leads directly into the following research questions which drive the analysis, identified as (RQ):

> **(RQ1)** How can the NNAV architecture be analyzed to capture its navigation capability?

In order to answer the hypothesis a series of experiments must be performed to capture the performance and capability of the analysis method. This approach must be able to both

capture the processing of state measurements as well as packet-based updates. An analysis approach is required that can allow for architecture design studies, system definition, and system evaluation. The end goal of this analysis will be a demonstration of NNAV's capability.

> **(RQ2)** How do the NNAV onboard algorithms analyze the embedded navigation packets into a measurement that can be processed by onboard state estimation filters?

In order to process the packet-based data, the spacecraft must contain orbit estimation and propagation routines. Additionally, a range of algorithms and techniques are available for tracking a spacecraft's state and processing state updates. A method must be chosen and rationale given as to the reasoning for the selection of state estimation procedure. These algorithms will drive the performance of NNAV, as state estimation and propagation are key aspects of deep space navigation. Additionally, analysis is required on how the packet-based information can be converted into a typical state measurement that can be used within the state estimation algorithms.

> **(RQ3)** How are NNAV packets and traditional measurements integrated into a common framework, both operationally and algorithmically?

To address the hypotheses above that focus on integrated performance of NNAV with traditional state updates, the state estimation algorithms must be able to process both type of data within one implementation. The selected state update architecture must allow for a range of data to be processed and analyzed to generate position, velocity, and time updates to be processed by the state estimation routines. These must be integrated into a unified architecture, that can handle a range of data formats and content.

> **(RQ4)** What is the sensitivity of onboard state estimation performance change to NNAV navigation packet information content?

In order to ascertain the capability of the navigation system, the ideal format of the embedded navigation header must be ascertained. Additionally, the performance of update

algorithms should be captured to show the trade-off between data content, packet size, and overall navigation system accuracy. The developed analysis must be able to thus capture a range of potential content information.

> **(RQ5)** What specific NNAV use cases must be analyzed to demonstrate reduced reliance on ground-based updates and verify operation in an initial space demonstration?

To capture the performance, a series of well-defined analysis cases must be defined in order to allow for the capture of the capability of the navigation system. This provides for an environment in which to exercise the experiments to capture system performance and capability. The chosen use cases must also coincide with expected initial implementation and operation of the developed algorithms in a flight environment and mission scenario.

2.9 Summary of Navigation Concept

Due to the complexity of deep space navigation and need for increasing levels of autonomy to enable increasingly complex missions, there exists a wide swath of research methods to define, evaluate, refine, and evolve the current state of the art in deep space navigation and measurement. The current methods are well-defined and executed but have a large requirement on ground resources to support state measurement, analysis, and onboard updating. The current research looks to move the processing to the spacecraft computer systems with additional sensor measurements, whether by optical, X-ray, or in-space radio instruments.

This thesis proposes a new approach to deep space navigation that combines aspects of multiple research approaches to develop a novel integration of the technologies. For the standard radiometric tracking techniques, these signals are embedded into the communication transmission by means of toning signals embedded into the spacecraft. These signals are not processed by vehicle, but rather are simply transmitted back to allow for two-(or three-)way ranging and ground-based orbit determination. A large amount of ranging and Doppler observations is required to calculate an accurate state estimate. This new technique looks to change how these navigation signals are embedded. Rather then at the analog

signal level, the proposed method embeds this information into the data being transmitted, allowing for the onboard processing to process the board, perform a navigation measurement, and update its state autonomously using its onboard state estimation processes. This allow for state estimation independent of ground-based orbit determination and reduces the need for long navigation passes. Additionally, by moving the navigation processing to the spacecraft, the latency of state updates is reduced due to the time required by ground-based data processing and analysis.

The navigation architecture envisions the growing in-space communication infrastructure, in particular the data relay nodes, additionally serving as a navigation network. This expands on other research focused on inter-spacecraft ranging techniques [34] [113]. By utilizing a physically dispersed group of hosts for measurements of range and range-rate, this method includes some aspects of GPS. The main difference is the usage of embedded navigation packets into the transmission packets as opposed to the use of a signal processing approach. GPS receivers measure range by observation of the phase of a predefined ranging sequence using embedded psuedorandom sequences, known as Gold Codes [87]. Interspacecraft ranging has also been used for gravity determination and characterization [93], such as the Gravity Recovery and Climate Experiment Mission [117], though the orbit determination and gravity analysis are done via complex ground analysis models. These missions and techniques provide the legacy implementations and inspirations to the proposed communication and navigation integrated architecture.

The proposed communication-based packet architecture provides an alternate path forward via the integration of navigation data into the transmitted data content. Prior research in interspacecraft and communication network design provide a strong foundation for the NNAV architecture. Through the embedding of these navigation packets, this method augments traditional navigation methods allowing for onboard state estimation and processing. Spacecraft-based processing enables vehicle position estimation autonomous of ground-based orbit determination processes, reducing the need for lengthy navigation observation passes and reducing the navigation update latency. To capture the performance of this navigation system and demonstrate its capabilities, the proposed research questions

must be addressed to support the hypotheses via a series of experiments.

CHAPTER III

NAVIGATION ANALYSIS APPROACH

This chapter presents the developed approach to designing, architecting, and modeling the navigation concept presented. A tool is needed that can capture a wide range of functionality in terms of state estimation, communication modeling, and state propagation. The proposed approach integrates elements of Model-Based Systems Engineering and Agent-Based Modeling into a unified conceptual framework. This approach enables development of a simulation environment that can capture both the performance of specific navigation systems, and model network-wide emergent behavior. This modeling and simulation environment is driven by the conceptual framework development. Additionally this will enable evaluation of state estimation performance as the individual agents increase in autonomy and the network evolves over time. The motivation for this and an overview of the analytical approach is described in this chapter.

3.1 Need for Navigation Analysis

To answer the research questions required to test the hypotheses, a series of experiments must be performed to address the performance and capability of the proposed packet-based navigation system compared to current state-of-the-art methods. Several methods are available, these include analysis by comparison to already existing systems, extrapolation from existing systems, and implementation and observation of performance. Due to the unique aspects of this navigation method, its performance cannot be extrapolated from current analysis methods. Additionally, state estimation performance data is not readily available to enable for comparison or extrapolation for comparison.

A standard implementation of state estimation or measurement processing for deep space navigation does not exist. Though some basic algorithms and techniques carry over from mission to mission, the algorithms are customized to suit the specific scenario under analysis. This includes varying the navigation requirements as well as the level of fidelity

used in gravitational modeling. For example, the New Horizons spacecraft takes advantage of Earth-based measurements for observation [77]. Due to the great distances involved in performing two-way ranging, advanced tonal signals are being used to enable signal detection in such a low level of received power detection. Additionally, due to the uncertainties in mass distribution and orbit between Pluto and Charon, the ground-based measurements are being used to determine both the position of New Horizons as well as the orbits and gravitational properties of Pluto and Charon. This requires a very complex state estimation filter in which both the vehicle position, the precise planetary positions, and gravitation fields must be modeled.

There is no standard spacecraft navigation performance analysis suite for deep space navigation from which to infer. Therefore, additional experiments must be performed to gather data on the proposed navigation system's performance to enable design and analysis. The greatest fidelity data comes from real-world experiments. But due to the extreme cost and time required to develop a deep space probe, this is not possible. Because of the large amount of investment in such systems and lengthy development cycles, it is not possible to integrate this test into existing spacecraft (though it could be via means of a software update). A large amount of forward planning and integration is required to perform any tests on an orbital platform.

As such, the data required for analysis must take its basis in analytic and computational methods. A modeling and simulation environment will serve as the experimental tool to perform analysis on system capability. This will need to address a range of use cases and measurement types in order to track the system performance over time for a variety of navigation architectures. The environment will act as a surrogate to physical experimentation in order to allow for system analysis. This approach does have some slight limitations. It is impossible to develop exact gravitational and force models. An analytical tool will provide evaluation to levels of accuracy that are considered reasonable for this type of early conceptual development. Additionally, even though ever-increasing fidelity models can be included, it is important to focus on initially capturing high level performance parameters, such as state estimation errors. The implementation of this framework will demonstrate a

proof of concept mission, allowing it to prove the feasibility of NNAV for further development and analysis. As initial analysis results are captured and processed, it is possible to identify the key system performance drivers and focus further modeling efforts on improving the fidelity of the environment. The specific requirements of the analysis environment are given in the following section.

In addition to developing an modeling and simulation environment to conduct analysis for a specific mission and navigation system, there are also higher-level needs that the conceptual framework must meet. As discussed previously, there is no standard deep space navigation analysis suite. Due to the wide range of mission scenarios being planned and large investments in these missions, it is important to develop a validated, proven, and open conceptual framework to centralize mission planning and performance verification. This seeks to integrate deep space navigation functionality into a unified environment to allow for improved collaboration and mission planning across the design cycle. This requires thorough analysis of a generic deep space navigation system in term of its requirements, use cases, operational scenarios, and structure. Capturing this data will allow for shared verification, validation, and documentation of the underlying analysis approach and serves as a method to spread knowledge of the tool among its users. In order to be applicable to all deep space navigation methods, this conceptual framework can be used to develop generic models that can be implemented across missions and scenarios.

3.2 Required Functionality of Framework Implementation

The defined hypothesis and research questions form the basis of the requirements for the implemented framework's modeling and simulation functionality. These needs will inform the development environment selection that will then be used to perform the analysis. The defined navigation functional requirements follow from the research questions as follows:

To address (RQ1), defined in the previous chapter, a series of functions can be identified to define the operation and functionality of the software-based simulation environment. These describe the main capabilities required to answer the research questions, enabling analysis on the identified hypothesis. This research question focuses on the need to capture

system-level navigation performance. First, the performance parameters must be identified. Typically, a navigation system's capability in measured in terms of its state estimation errors. For an inertial frame I, the navigation position and velocity error can be identified as follows:

$$\epsilon_{Position} = \sqrt{(x_{true} - x_{est})^2 + (y_{true} - y_{est})^2 + (z_{true} - z_{est})^2} \tag{1}$$

$$\epsilon_{Velocity} = \sqrt{(v_{x,true} - v_{x,est})^2 + (v_{y,true} - v_{y,est})^2 + (v_{z,true} - v_{z,est})^2} \tag{2}$$

$$\epsilon_{clock} = t_{sc,estimated} - t_{true} \tag{3}$$

This measure captures the difference between the estimated and true state. Another parameter that must be included is the clock error, due to the timestamping of packet arrivals being used in processing. For performance analysis and comparison, typically the normalized value of each error will be used. An additional performance measure is the number of measurements and the data of the required packets. Also, to capture the time-based nature of the navigation problem, integrated error terms have also been identified to capture the total integrated position and velocity error over time. These are given in Equations 4, 5, and 6. Due to the stochastic nature of the analysis, these values are typically numerically integrated from step to step, or they are calculated via post-processing analysis.

$$\epsilon_{position,integrated} = \int_{t0}^{tf} |\epsilon_{position}(t)|dt \tag{4}$$

$$\epsilon_{velocity,integrated} = \int_{t0}^{tf} |\epsilon_{velocity}(t)|dt \tag{5}$$

$$\epsilon_{clock,integrated} = \int_{t0}^{tf} |\epsilon_{clock}(t)|dt \tag{6}$$

These terms track cumulative error over time. Due to the infrequent and varying time intervals between measurements, this measure captures the integrated performance of the

state updates and propagation effects. This is preferred to average error, which cannot capture the time between updates. These identified parameters focus solely on the state estimation performance of the navigation method.

With the desired measures identified, it is possible to break down each requirement into the functions required to calculate each. At the highest level, the simulation must track the true state of the spacecraft and the estimated state of the vehicle. To capture the state of a vehicle, whether true or estimated, the implementation must be able to load reference initialization data as well as include propagation models to integrate the vehicles estimated or true position. Additionally, for reference to true missions, the toolset must be capable of loading true reference vehicle data. In order to propagate the vehicle state, the simulation must estimate the inertial and non-inertial states on the vehicles. To capture the time-based nature of the simulation, the tool must also be capable of modeling clock error, as well as any onboard corrections that may be performed. Additionally, to integrate the state method, the implemented framework must include the capability to solve Ordinary Differential Equations in order to propagate the modeled state.

In order to capture the navigation performance of the proposed method, it is necessary to capture both the measurement modeling and the state estimation capability. To trade and capture the widest variety of measurements possible, the implementation of the framework must have a well-defined modular interface to additional analytic models to capture a range of spacecraft-based observations. The ability to integrate modular state estimation models is also necessary in order to trade various algorithms. A well-defined interface is required in order to allow for a wide range of analysis cases and vehicle scenarios. A summary of these requirements is given in Table 6.

The second research questions focuses on the capability to capture the performance of the navigation packets. The simulation environment must be capable of modeling space- and ground-based communication assets in terms of their transmission and reception capability. This is needed to monitor when two spacecraft can close a communication link and transfer data. Modeling the deep space telecommunication link is crucial to identifying when the

Table 6: Requirements to Address (RQ1)

Capture Navigation Performance
1.1 Initialize body to reference/input data
1.2 Compare estimated to reference/truth data
1.3 Calculate non-inertial forces on body
1.4 Capture inertial forces on body
1.5 Integrate body's state
1.6 Modular measurement interface
1.7 Modular state estimation interface
1.8 Calculate State Errors as a function of time

Table 7: Requirements to Address (RQ2)

Capture Packet Performance
2.1 Perform Deep Space Link Analysis
2.2 Autonomous Packet Generation
2.3 Capture Transmission Delays
2.4 Autonomous Reception and Processing of Packet
2.5 Integration of Packet with State Estimator
2.6 Onboard estimation of other SC states

packets can be processed. Additionally, the implemented simulation framework must be capable of the generation of the navigation header content based on an asset's best known state information. The simulator must also process the packets forward in time to determine the true time of arrival to capture transmission delays, and forward the packet to the correct agent at the true arrival time. This autonomous generation of outgoing packet and their reception is a key component in the simulation infrastructure. Additionally, the state estimation models must be capable of processing the packet content into usable measurements which can be integrated into a state update. In order to process the received packet, the spacecraft can make no assumptions on the packet content, and therefore must propagate the other body's state in addition to its own. This is used both for packet processing as well as determining when to initiate communication links and when it may be in contact range with other assets. These requirements are all summarized in Table 7.

To analyze the trade-offs between navigation packets and state measurements, it is important to utilize a state estimation algorithm that can integrate both types of information

Table 8: Requirements to Address (RQ3)

Packet and Measurement Integration
3.1 Interface to external measurement models
3.2 Model Autonomous or Scheduled Measurements
3.3 Process measurement into state estimator

into its state estimation techniques. Additionally, the simulation environment must be capable of generating measurements at predefined intervals (or autonomously generated) to model an onboard navigation sensors. To enable comparison against a wide range of observation methods, the estimator must have a well-defined interface for external measurement generation and integration to the state estimation filter. An overview of these is given in Table 8.

The fourth research question (RQ4) addresses the need to perform design space exploration and analysis. In order to trade packet, measurement, and spacecraft parameters, the simulation environment must provide links to standard design tools. These include a Monte Carlo analysis package to facilitate statistical design space exploration and identification of trends in performance due to variable inputs. Additionally, the tool must support an external file input interface to allow for loading of predefined analysis cases to enable design space exploration and sensitivity studies. An interface to optimization routines is required to allow for the determination of optimal packet content, as well as optimal measurement updates. These design tools are required to support analysis trades and optimization. As part of the analysis, the specific frequency of measurements and packets, specific packet content, and state estimation parameters will require the use of Monte Carlo analysis and optimization in order to provide optimal parameters for a given vehicle and mission scenario. Additionally, other design tools may be needed to support identification of system performance trends. This can be implemented by means of a well-defined interface to the framework implementation. Table 9 provides a summary of these requirements.

Lastly, the implemented simulation framework must be capable of running a variety of cases to capture specific analysis scenarios. (RQ5) focuses on the development of these

Table 9: Requirements to Address (RQ4)

Design Trades and Analysis
4.1 Model and vary packet content and measurements
4.2 Integration with Monte Carlo tools
4.3 Capability to perform optimization on packet properties
4.4 Modular interface to external design tools

Table 10: Requirements to Address (RQ5)

Use Case Definition
5.1 Support a range of analysis scenarios
5.2 Modular input to allow for saving and loading of use cases
5.3 Robust framework to variety of studies

usage scenarios. The software tool must have the capabilities to analyze a wide range of user-defined scenarios to capture a range of performance. The requirements are identified in Table 10. These include verification cases as well as specific design cases that focus on specific functionality and integration of packets and measurement types. Thus the framework implementation requires a well-defined set of interfaces that allow the designer to select and build use cases, with an analysis backend to support a range of missions and analysis scenarios.

3.3 Current Methods of Navigation System Analysis

There are several approaches to the development of simulation software to address deep space navigation design. Each has a unique implementation approach and typically focuses on one aspect of the design. This section characterizes a range of currently available approaches. Each will be characterized in terms of the identified functional requirements to enable analysis of a packet-based navigation architecture.

Analysis of the underlying physics and geometry of navigation form the basis of any simulation architecture capturing the performance of a navigation system. These methods focus on the general physical principals involved in communication link analysis and observation methods to derive performance equations. Identification and modeling of noise parameters allows for predicting measurement accuracy. By capturing the distance and orientation between assets, it is possible to model and analyze individual observations between

assets.

This method of analysis also includes modeling and prediction of the forces acting on the spacecraft, including gravitational and other stochastic effects. By capturing the applied forces on a body and the use of numerical integration techniques, it is possible to simulate the trajectory of a spacecraft. In addition to modeling the dynamics of the vehicle and the actual measurements, it is also important to develop an implementation of the state estimation algorithms under analysis. The combination of these assets allows for the development of an integrated toolset to capture both the true dynamics of the spacecraft as well as the onboard estimate of the vehicle's state parameters through integration of modeled measurements and use of implemented state estimation filters. Through direct analysis of the nonlinear dynamics and filtering process, it is possible to capture the performance of the algorithm. Additionally, the calculation of state update parameters allows for direct assessment of the navigator's performance. Due to the random nature of the measurement errors, a Monte Carlo Analysis is typically used to capture mean performance over a series of observations.

An additional approach to navigation system performance is the use of linear covariance (or LINCOV) modeling [49] [115] [20] [72]. This method involves the use of a linearized parameter estimation model to estimate the errors in the estimated state. It does not directly model the navigation updates, but rather focuses on capturing the predicted error over the course of the trajectory. This is performed by linearizing the errors about the design trajectory. This allows for analysis of the estimated capability of the approach. By using the linearized models, it is possible to capture the performance of the estimators and measurement with only one analysis case.

The analysis methods mentioned are all computational-driven, requiring implementation in an analysis tool. Several additional approaches to simulation design are described by [109] [133] [134]. These references present methods to approach the analysis, though typically in the scope of a particular analysis scenario.

3.4 Generic Framework Approaches

Due to the continued increases in computational processing capability, both in terms of computational capacity and available memory, computer-based experiments have become a staple of engineering systems design and analysis. The increases in processing capability enables both increased fidelity of the analytical models developed and reductions in processing time. This allows for either faster analysis turnaround or improved capture of stochastic effects through the use of Monte Carlo simulation, increasing the number of analysis performed. The use of these modeling tools also allows for a great deal in flexibility and application to a wide range of scenarios. As these computer-based modeling and simulation methods continue to gain traction in engineering conceptual design and architecture analysis, there is a growing push to address this development using a formal approach. This is particularly true in terms of software systems to enable design and code re-use[95]. This is needed to enable understanding and provide development paths for large intricate simulation architectures.

From roots in software engineering and development, the concept of a framework has emerged. A thorough definition of framework is given by Riehle[95]:

> They (frameworks) represent the domain as an abstract design, consisting of abstract classes (or interfaces). The abstract design is more than a set of classes, because it defines how instances of the classes are allowed to collaborate with each other at runtime. Effectively, it acts as a skeleton, or a scaffolding, that determines how framework objects relate to each other.

This notion is used to capture the system under analysis to provide a foundation for analysis and design, providing definitions, prototypes, and interface definitions for objects within a system. This has been applied to design of simulation of a variety of systems such as missiles[76], satellites[83], or tactical simulations[1][22]. This is then used to feed into system or software design. By developing a validated generic framework, the design is able to use it as a foundation for further analysis studies and design of similar systems within the specific domain An example of this is using a spacecraft design framework's elements

across a variety of mission scenarios. The investment in understanding and documenting the system provides a common definition of the system and the analysis functions required.

This need has developed into research and development of generic analysis frameworks, such as described in [88] which applies interface design and system decomposition to allow the generation of integratable software blocks. These architectures generalize the analysis needs and functionality in order to provide a computational capability independent of the specific scenario. Formal building blocks of analysis needs are defined at a low level and are combined to form the analysis environment. Functional blocks, specific to a defined analysis case, are then built upon these foundations to form an architecture or scenario under study. The development of a generic conceptual framework allows the decoupling of specific performance or operational modeling from the underlying operation of the simulation. This provides for a standard conceptual framework that supports a wide variety of missions and scenarios. These can be analyzed by defining operations, behaviors, calculations, and implemented models. By utilizing a common simulation framework, it is possible to focus effort on the system performance and analysis models.

Primary research in the field of framework design deals with capturing increasingly complex systems. This complexity of the framework and the objects required to build it up are one of the disadvantages of framework design as described by [95]. Riehle approached the framework design by defining roles to each aspect of the architecture and using that to drive high level analysis and modeling. It is also difficult to capture the independent behavior of complex objects in the architecture while still maintaining their collaborative nature. This has been addressed by Siegfried[107] in the development of a standard architectures for defining behavior in complex modeling systems. The main complexity with framework development remains with the definition of the system and the usage of modeling approaches to capture these complex relationships. One such modern integrated approach is the integration of SysML and Model-Based Systems Engineering, which use formal graphical modeling to capture complex system behavior, requirements, and structure. This approach is described later in this chapter, after an overview of the current relevant space navigation disciplinary tools available.

3.5 Current Tools and Implementations

There are several tools available for the analysis and design of space missions. Each has a primary focus and objective, and is particularly geared towards a specific analysis problem. The current field of deep space navigation and communication design was surveyed to provide an overview of some of the primary tools that could be applied to the proposed navigation architecture. These are from a variety of institutions, government to academic to commercial, and together provide analysis towards many assets of mission planning and analysis.

3.5.1 Orbit Determination Toolbox (ODTBX)

The Orbit Determination Toolbox[1] is a project based out of the Goddard Space Flight Center. The product consists of an analysis package for the MATLAB programming environment to allow for orbit determination analysis and design. A range of measurement models and several state estimation methods, such as sequential state estimation, batch filtering, and linear least squares, are included. This tool models, in detail, a range of spacecraft navigation observations.

The analysis package has a strong focus on low earth orbit analysis cases, including detailed gravity models of the earth by default, and providing ready-made dynamics models for these types of missions. The tool is driven by input dynamics models both for true and modeled trajectories, and does not support the loading of external data to drive a true trajectory. The modeling environment does support a wide range of measurement and analysis capability.

3.5.2 STK

AGI's Satellite ToolKit[2] is a top-of-the-line package for orbit propagation and mission analysis. It includes extensive packages for designing missions and supports highly detailed visualization capabilities to aid in analysis and documentation. This tool has wide usage for determining link design, coverage, and mission planning.

[1] http://sourceforge.net/projects/odtbx/
[2] http://www.agi.com

There is not much capability though in terms of navigation design. While the software can model the communication links required for a GPS-like navigation constellation and can propagate a vehicle's trajectory, the underlying navigation filters are not implemented. The primary interface to navigation studies occurs through the use of external software that can link with the primary STK executable.

3.5.3 Open-SESSAME

Open-SESSAME is the Open Source Extensible Spacecraft Simulation and Modeling Environment Framework [123][3]. This is a software package developed in C++ at the Space Systems Simulation Laboratory at Virginia Polytechnic Institute and State University and released in 2003. The intent of this development library is to develop a standard functional basis for simulation analysis for a range of spacecraft with a focus on individual vehicle control analysis and design and wide intended applicability. The software itself provides the functional backend of a simulation. Though the library has a wide range of functionality, its last developed release was in 2003, and has not seen much recent activity.

3.5.4 Space Network Protocol Emulators

To model the communication networks, protocol emulators similar to the Interplanetary Overlay Network simulation[4] or other network emulators [31] are typically used. This tool provides very detailed modeling of the communication processes and data transfer rates, focusing on the implementation and capabilities of the protocol. This allows for a range of studies of network distribution and infrastructure. The models are not linked with spacecraft estimation routines or autonomy models, focusing on the communication aspects of the mission analysis.

3.6 Gaps of Current Tools to Required Functionality

Each of the available tools has it own strength and unique approach to modeling spacecraft capabilities. All of the packages follow a simulation-based dynamic modeling approach

[3] http://spacecraft.sourceforge.net/
[4] https://ion.ocp.ohiou.edu/

to the analysis. ODTBX additionally includes the capability for linear covariance-based analysis for state estimation performance evaluation. These tools individually allow for a very thorough examination of state propagation routines, orbit determination for low earth orbit, and communication protocol modeling.

In order to capture the capability of the emerging navigation systems a framework is required that includes all of these aspects of analysis. All of these must be included due to the highly integrated nature of communication, navigation, and timing in modern deep space navigation systems. Research into new autonomous navigation system design is done individually for each mission, with specialized software and analysis tools typically tied to a specific mission or technology. A framework is required that allows capture of all aspects of the navigation problem, allowing for an integrated simulation capability, from requirements analysis and conceptual to model implementation.

The above-mentioned tools each address a particular function within the needed framework's requirements. A summary of each tool's capabilities are given in Table 11. But the integration and availability of the tools is a problem. STK is a very capable toolset in terms of mission planning and trajectory design, but does not have much capability in terms of deep space navigation. ODTBX is also very powerful, but is tied to the MATLAB computational environment, which is closed source. ION and Open-SESSAME are both open source packages and allow for analysis across a range of platforms. The only down side is their limited capability in space navigation, and need for large integration efforts. While ION is currently under active development, Open-SESSAME's last release was in 2003, and has limited documentation and user base available. This is in contrast to the other packages which are all in active use.

3.7 Framework for Navigation System Simulation and Analysis

As described above, there are gaps in the capability and approach of current tools to enable the analysis of a packet-based navigation system, such as NNAV. Several tools are available that do a strong job of spacecraft state propagation. There also exists a very capable set of tools that have been developed to perform ground-based orbit determination with accurate

Table 11: Software Package Capabilities

Analysis Requirement	ODTBX	STK	Open-SESSAME	ION
1.1 Initialize body to input data	√	√	√	
1.2 Compare estimated to reference/truth data	√	√	√	
1.3 Calculate non-inertial forces on body	√	√	√	
1.4 Capture inertial forces on body	√	√	√	
1.5 Integrate body's onboard state	√	√	√	
1.6 Modular measurement interface	√		√	
1.7 Modular state estimation interface	√			
1.8 Calculate State Errors as a function of time	√			
2.1 Perform Deep Space Link Analysis	√			√
2.2 Autonomous Packet Generation				
2.3 Capture Transmission Delays	√			√
2.4 Autonomous Reception and Processing of Packet				
2.5 Integration of Packet with State Estimator				
2.6 Onboard estimation of other SC states		√		
3.1 Interface to external measurement models				
3.2 Model Autonomous or Scheduled Measurements	√			
3.3 Process measurement into state estimator	√			
4.1 Model and vary packet content and measurements				
4.2 Integration with Monte Carlo tools	√	√		
4.3 Capability to perform packet optimization				
4.4 Modular interface to external design tools	√	√	√	√
5.1 Support a range of analysis scenarios	√	√	√	√
5.2 Modular input and interface to use cases	√	√	√	√
5.3 Robust framework to variety of studies	√	√	√	√

capture of a wide range of error sources. For in-flight navigation, in-house algorithms and processes are slightly tweaked and adapted to new problems. This allows for a large flight legacy as well as the continual improvement of the navigation functionality. But for entirely new navigation systems or measurement approaches [105] [74], typically the analyst must develop an analytical tool from the ground up to capture the underlying physics of the problem. These developed tools usually become in-house mission-focused efforts.

In order to speed analysis of advanced navigation systems and to be able to quickly trade competing measurement approaches and sensors, a standard library is needed to both aid in initial design and to allow for architecture development. A standard package of analysis tools also allows for the user to increase the functionality of the environment over time by including advanced force models, timing models, measurement techniques, and advanced state estimation approaches. These sub-models are then integrated into one standard simulation and modeling environment which is linked to systems analysis tools, such as optimizers to maximize state estimation parameters.

As mentioned above, ODTBX takes initial steps in this direction for developing a standard space navigation package, though it is more focused on orbit determination than autonomous state estimation. It provides a capable analysis environment for navigation systems analysis. The tool does not however include the capability for multiple spacecraft propagating at the same time or for onboard estimation of other spacecraft, which is required in NNAV. Additionally, the arrangement of the tool with its layers of functionality can provide a steep learning curve to additional functional implementation, both for state estimators and measurement models. The tool is also limited to computers with MATLAB(R) functionality.

To enable a range of navigation experimentation and analysis, a new architecture can be shown to both ease software development while still allowing for a complex system to be modeled. This proposal recommends a framework which integrates well-developed Model-Based Systems Engineering techniques and tools to capture the functionality of the simulation modules to provide verification of the development of the environment. These models will additionally serve as a knowledge repository for future module design and

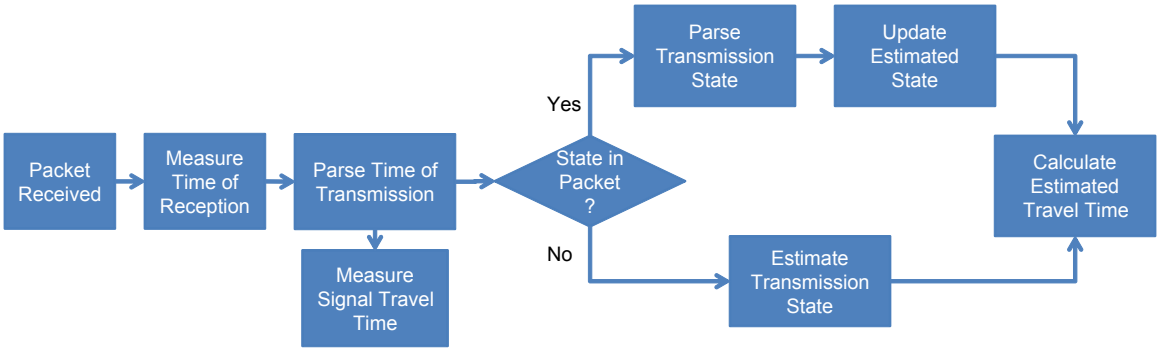


Figure 14: Typical Algorithm Flow Diagram

implementation, providing strong documentation of the interfaces and analysis approach to aid in future development efforts. This framework can also be used to capture and define potential behaviors to be implemented in the simulation environment.

The other factor in allowing the simulation of a complex network of interacting independent spacecraft is formulated from Agent-Based Modeling approaches. This method of analysis focuses on the development and study of complex behaviors of systems of assets. It is specifically focused on analyzing independent agents, which have some level of autonomy and decision-making. These techniques will inform the development of the environment in order to allow for advanced studies and further analysis of spacecraft behavior, enabling research into areas of adaptive navigation. These two driving approaches are briefly described below.

3.7.1 Model-Based Systems Engineering and SysML

With the advancement of computer modeling tools and standardization of computational languages to capture system relationships, the method of Model-Based Systems Engineering (MBSE) has emerged. This approach to systems engineering provides an alternate approach to the current complex requirements traceability methods currently in use. The main driver for a model-based approach is the deficiencies in traditional document-based approaches. Some main issues with this process are that the specifications are often defined after the fact, purely as a documentation measure, the written words tended to be ambiguous, and the requirements generated are seen as nothing more than a paper effort[37].

Current applications of requirements tracking and functional decomposition are developed using tools intended to be used as presentation aids (such as PowerPoint), diagramming tools (such as VISIO), and spreadsheet utilities (such as Excel). An example of this type of diagram is the generation of algorithm flows diagrams in PowerPoint such as that shown in Figure 14. This model is static and created within a presentation tool, which limits the representation richness and the capability for computer-aided processing and integration. The current methods capture requirements and system definition in complex diagrams, that are typically tailored to a specific group or user and have limited re-usability. These approaches serve mainly to present information to a human, with limited uses beyond documentation. As detailed requirements tracking techniques are applied to increasingly complex engineering applications, the tools becomes increasingly problematic to generate and read. This is due to the size of the data and the complex internal relationships.

The disconnect between requirements and design engineers increases the risk of miscommunication. Issues such as un-addressed requirements and differences between the system as designed and implemented, become increasingly probable as an effect of using non-standard disconnected design and analysis tools. As demonstrated above, the current models of the system are typically static, and once created can be difficult to change or update. As these systems analyses become more complex and are linked to the analysis and implementation, there is a growing need for an executable, dynamic architecture that can easily change over time, be easily editable, and be tied directly to the actual implementation and physical properties of the system under study.

As opposed to these traditional documentation techniques, MBSE is the formal application of models to support systems engineering throughout the project lifecycle[63]. This is similar to the application of 3D feature-based CAD models to support mechanical design in comparison to traditional 2D drafting approaches. These 3D CAD models provide greater insight into the system as a whole, and are directly geared towards implementing clear traceability from systems requirements to the systems operational modes to its designed structure and specific implementations. These methods link the standard systems engineering practices through these formal model constructs to both capture knowledge and

develop a system under analysis[46]. This combination allows the designer to easily track the assumptions and functional and structural decomposition of the system for traceability and for knowledge transfer. Additionally, this can act as a bridge between the systems engineer and designer, allowing for allocation of requirements to specific systems[46]. There is a wide range of implementations of MBSE techniques[32]. These techniques have been used to develop and optimize in applications from spacecraft control[70] to subsystem design[71] to communications systems[120].

One current embodiment of these techniques is an adaptation of the Object Management Group's (OMG) Unified Modeling Language (UML)[5], which is widespread in supporting software engineering. This new approach is named the Systems Modeling Language, SysML[46][6]. SysML pulls in standard systems engineering model views and nomenclature into a formal standard, building on a subset of UML functionality[127][46]. The formal definition of SysML[86] is given as:

> The OMG Systems Modeling Language (OMG SysML) is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities. In particular, the language provides graphical representations with a semantic foundation for modeling system requirements, behavior, structure, and parametrics, which is used to integrate with other engineering analysis models.

SysML has four self-defined pillars that capture multiple views of a system. These are: Structure, Behavior, Requirements, and Parametrics[86]. These four pillars allow comprehensive definition of a system, from initial design to implementation. SysML has broad capability to support system engineering in architecting new systems via these pillars. It is a broad standard supported by groups such as the OMG and the International Council on Systems Engineering (INCOSE) [86] [127].

These methods have been implemented to address a variety of systems engineering

[5]http://www.uml.org/

[6]SysML is a trademark of Object Management Group, Inc. in the United States and/or other countries.

problems. For example, the INCOSE MBSE Space Systems Challenge Team[26] is currently researching and documenting the application of MBSE through SysML to a series of spacecraft systems design problems[112] to both expand current methods and demonstrate capability to foster increased usage. Recent progress in this work[111] documents the integration of modeling tools and the morphing of SysML into an executable architecture. This was achieved by using software interfaces, such as ParaMagic[7], to develop embedded executable functions within the SysML views as well as links to external software analysis tools such as STK and Matlab. The team was able to demonstrate the analytical functionality within the SysML environment to allow for calculation of communication parameters and simulate time-dependent mission parameters. The implementation was limited by the difficulties in integrating the broad array of external software environments into one modeling and simulation environment. This was noted to cause difficulty in software integration and debugging and demonstrates the difficulties in integrating complex simulation software with the SysML framework.

In addition to space systems, MBSE and SysML have been applied to a large variety of systems engineering problems. In order to provide for documentation of the developed SysML models and requirements tracing and automated change tracking, Delp et al. have developed an extension of the framework to allow for automated document generation and extraction of key model information[16]. Bajaj et al. have integrated the SysML architecture with a discrete event simulation, Orchestra, for the design of embedded electronic systems[4]. This implementation focused on defining the structure of the simulation within SysML and using domain specific language to allow for a direct simulation of the elements. Additional work by Bajaj focuses on the development of a full-scale System Lifecycle Management (SLIM) software platform that links a wide variety of analytical capabilities to allow for comprehensive system design and analysis built around a central SysML system definition[5]. This work also documents the application of these principles to satellite design, military operations, and financial planning[6]. Vanderperren applied the requirements analysis aspects of SysML to capture system-on-a-chip design[126], demonstrating how this

[7] http://www.intercax.com/products/paramagic/

application of SysML allowed for improve stakeholder communication. This has also been applied to submarine weapon systems[90] and space telescope design[69].

These applications show the wide usage and growth of MBSE and SysML. A focus of current research in this field is the integration of complex simulation tools with the modeling environment. Although the current tools offer software packages to integrate the systems modeling with detailed analysis, these are very tool-, vendor-, and application-specific, due to the complexities of software integration. With continued usage and development of the SysML standard, these integration libraries will become increasingly important to demonstrate a centralized executable environment that allows for system design, modeling, and evaluation.

3.7.2 Agent-Based Dynamic Simulation

Agent-Based Modeling (ABM)[128] [82] [21] is an approach to dynamic systems simulation that focuses on the analysis of systems which are highly interactive and often stochastic in nature. This field of simulation focuses on observing high level systems trends, which are a result of the bulk behavior of the interacting agents. ABM builds on the development of Discrete Event Simulation, which focuses on analyzing a sequence of operations and their interactions, but expands the approach to a continuous systems allowing for changing operations. The analysis focuses on the identification and emergence of complex behaviors of the group of systems. The systems under analysis are geared towards capturing the dynamics trends in the population. There are many simulation approaches for implementing this analysis, but most involve independently operating agents that can interact and act on their own.

For the analysis of a deep space navigation and communication network, each spacecraft can be considered an independent agent, with onboard measurement processes and unique errors. The incorporation of these simulation methods allows for the capture of a wide range of spacecraft behaviors, including spacecraft pointing, inter-spacecraft link scheduling, and data transmission. More importantly, this type of modeling allows for capture of the performance of individual spacecraft assets as well as the performance of the entire network.

These methods provide a robust approach to implementation and simulation design to allow for a variety of studies allowing in-depth analysis of a system of spacecraft and their intrinsic individual and group behavioral trends.

3.7.3 Integrated Approach to Navigation System Development

This proposal develops an integrated approach to deep space navigation simulation and analysis. This methodology builds on the capabilities of both techniques described to take advantage of their unique capabilities. Model-Based Systems Engineering techniques will be used to identify the functionality of the tool, from the high-level requirements to the functional allocation of specific implemented software modules. The use of these tool is similar to other applications of MBSE techniques to capture deep space communication networks [9]. These methods are used to enable a thorough understanding and knowledge capture model of deep space navigation. The application of these techniques at multiple levels of composition allows for both analysis of spacecraft subsystems interactions, and captures the implementation of specific measurement models. The robust capability of MBSE allows it to capture a wide array of data. It can be used to capture spacecraft composition, interactions with both internal and external agents, specific activities and behaviors, and instantiations of the modeling agents. These developed models also act as class definition and provide pseudocode for the implementation of the simulation environment and the specific agents.

These models will then be captured using simulation methods inspired by ABM. Unique agents types are defined by the use case and defined capabilities. To support analysis development, each agent's behaviors and attributes are described and developed via MBSE-approaches within the conceptual framework. This allows the designer to capture specific spacecraft behavior relative to specific use cases that capture the desired operation of the navigation system. Similarly, utilizing an Agent-Based approach to the system analysis allows for capturing a range of independently acting agents, and supports the framework capability for advanced spacecraft autonomous algorithm development, implementation, and analysis. This provides a very strong analogue to the actual system under study.

Utilization of this analysis approach allows for capture of individual vehicle navigation

performance and capability trends at the entire network level. By simultaneously modeling all aspects of the network, it is possible to capture the global capability of the entire network. This allows for identification of performance and navigation capabilities to be observed as a function of growing network size and identification of individual spacecraft behaviors. By allowing the spacecraft to have variable behavior, trends in terms of navigation update rates and capability can be observed.

The proposed method presents a model-based approach to Agent-Based systems design and implementation. The systems engineering techniques are used to capture the system and interactions between objects in the simulation. Block Definition Diagrams are used to show the interactions between defined objects as well as identify attributes and define a common definition structure. Activity Diagrams and State Machine Diagrams are used to capture the behavior of the various simulation subsystems (both spacecraft and simulator). This serves as the pseudocode and algorithm definition of the integrated analysis framework implementation. This level of systems modeling is then used to re-inform the definition and define common functionality of subsystems and objects. The Systems Engineering models are used to thus define the architecture of the simulation as well as inform the analytical implementation. This technique allows for full definition, documentation, and exploration of the analytical requirements of the modeling and simulation environment within the conceptual framework and leads to an object-oriented approach, preparing the system for the use of additional libraries and future analyses.

The Block Definition Diagrams can also be exported to machine-readable formats and used as stereotypes to define templates for the data input/output structure for the implementation of the framework. This allows the user to define the navigation system at a conceptual level and links these models to the physical code. This provides traceability to the analysis and centralized object definition and documentation. The Activity and State Machine Diagrams form the basis of the implemented simulation coordinator task, as well as declarations of subsystem functionality and properties.

The analysis approach can be summarized as a sequence of high-level activities that allow for the complete simulation and analysis of deep space navigation systems. The

approach contains three main steps: System Requirements Analysis, Analysis Framework Design and Implementation, and Simulation Verification. These steps flow into each other with each step feeding products into the next. The analysis flows from the top-level capture of the navigation concept under design down to the individual software modules that analyze functionality and back up through the validated framework implementation to provide analytical results of the overall navigation architecture. Together these all form an integrated research approach which can be summarized by the acronym **CRAIVE** (Concept of Operations, Requirements, Analysis of Framework, Implementation of Analysis, Verification of Models, Evaluation of Concept) and mirrors the overall structure of this thesis.

The first part of the analysis method focuses on the high level generic deep space navigation system. To begin the analysis, a **C**oncept of Operations must be developed to capture the high level functionality and use of the navigation system. This represents system characterization and serves to visually collect the systems and its operations to feed into the analysis. This conceptual analysis for NNAV is given in the previous chapter.

The second step is to analyze the **R**equirements of the navigation system. This is performed at the highest level, capturing the functional needs and how it must interact with external assets and the uses it must perform. With this understanding, the systems designer can continue with the **A**nalysis of the conceptual navigation framework. This is performed by further developing the requirements and use cases using Model-Based Systems Engineering tools. The goal of this modeling is to capture in detail the navigation system's interactions and functional interfaces among its subsystems and external assets. This step includes the development of diagrams capturing the sequential operations, providing a high level view of its dynamic functionality. The modeling concludes with the decomposition of the navigation into logical blocks with defined attributes and operations that work together to fulfill the defined requirements.

With the conceptual system information captured, the next phase begins by focusing on the **I**mplementation of the framework. The inputs to this stage are the logical design of the navigation design and its sequence of operations. The requirements of the conceptual framework are developed to capture the functional needs required to analyze the navigation

system's performance. This is done by mirroring the system design at the software level, with the framework design closely following the algorithms and functional breakdowns laid out previously. Upon completion, these models form the definition and outline of the software implementation. Using the developed models and blocks as a reference design, the objects are implemented in simulation. Similarly the system's operations are also modeled within the framework.

Once the simulation implementation is complete, the functional blocks require **V**erification to ensure proper operation. This step compares the results of the implemented analytical models for specific use cases to compare with results from standard software packages. This step is necessary to validate the results of the analysis and to provide comparison of the implemented framework capability to other packages.

With these steps concluded, the now-verified simulation environment can be used to **E**valuate the performance of the navigation system under consideration, specifically NNAV for this thesis. At this point, the implemented analysis tools are applied to the original scenarios of interest to determine high level performance and capability measures. With this step, the method's loop closes, with the verified developed environment feeding back into the analysis and design of the high level navigation system and providing design iteration.

3.8 Proposed Capabilities of Navigation Framework

With this process forming the basis of the development of a space navigation framework, several higher level hypotheses can be formed that describe the expected benefits. These are derived from the implementation of the framework and capture its expected benefits and improvements to existing methods.

> **(H4)**Integration of MBSE and ABM analysis approaches into a unified navigation framework will capture analysis of multiple independent measurements, packets, and spacecraft.

This relates directly to the capability of the framework to enable the modeling of a range of state estimators and measurement sources. Utilizing a modular architecture and robust interface design, it is expected to enable the evaluation of a wide range of navigation

problems, without making any assumptions of the type of data being receive. The only constraint will be on the interface definition.

> **(H5)** The navigation analysis framework will require implementation in an object-oriented simulation environment, to allow for expansions and inclusion of a range of external measurement, state estimation, and analysis libraries.

The Model-Based Systems Engineering approach, in addition to allowing for multiple measurements, enables the inclusion of external libraries to capture any functionality of the analysis. This is also enabled by the clearly defined interfaces and simulation linkages defined through the application of model-based engineering methods.

> **(H6)** Navigation framework modeling will enable definition of input and output interfaces, to provide a common data definition for the implemented simulation environment.

An additional benefit of the model-centric approach is the direct application of the system definition models to the implementation of the interfaces defined within the framework. Block Definition Diagrams, used in defining the composition of the navigation system and simulation package enable this capability. The modular approach to input variable definition leads to a robust interface that can integrate with other specific implementation needs to form a well-defined data definition format for the definition and storage of analysis cases and system parameters.

Through the use of modern flexible programming languages, the implementation of the framework will be allow for user input and definition to the model. This can be achieved both by architectural definitions in the Model Diagrams as well as directly into the files loaded by the simulation environment. This allows for a robust interface layer between the user and the software to allow for several levels of simulation definition.

The models also capture the internal behaviors of the spacecraft navigation system. These models are used to give insight into the processes that must occur both onboard on the spacecraft and within the framework implementation to allow for state estimation

procedures and navigation analysis. With these tools, additional understanding of the processes involved is made clearer to the systems designer and allows for a strong tie-in between implementation and definition of spacecraft behaviors.

> **(H7)** The execution of navigation framework implementation will captures the performance of various measurement types, enabling design space exploration and analysis.

With the use of Agent-Based simulation design, the simulation backend will be able to analyze the dynamic behavior and linkages between multiple spacecraft assets. Additionally, this acts as a verifiable analog to the actual system under study, allowing for evaluation of performance and capturing of system attributes. A modular object-oriented front-end, coupled with a user interface allows a range of design space explorations, and trade studies to be performed.

> **(H8)** The incorporation of ABM techniques in the simulation implementation will enable optimization of state estimation processes and algorithms through variation of onboard spacecraft behaviors.

With the capability of the framework implementation to analyze a variety of usage cases and scenarios, the analytical modules can be utilized in additional ways. Another primary usage of the integrated software package will be to optimize the various aspects of the navigation system under study. In order to address the highly stochastic nature of the problem, which is due to the modeled random noise in measurements and communications and the estimation process, an off-the-shelf genetic algorithm will be used to perform this optimization. The modular implementation and robust interface will allow for the integration of external optimization tools in order to ascertain the best possible performance of the navigation architecture and gain additional insight into the optimal system design.

3.9 Research Focus

In order to address the hypotheses discussed above, the design approach will be executed and the resulting simulation architecture tested in order to verify these statements. Several research questions to address the capability of the framework are described in detail. These will serve as the foundations of the implementation and method development for the framework used in order to be able to address the required functionality.

> **(RQ6)** How can MBSE methods be used to capture multiple measurements, spacecraft, and state estimators algorithms for a navigation system?

To show the capability of the framework to capture various spacecraft, measurements, and estimators, the model based approaches will be utilized. This research question addresses specifically how the framework will be used to capture the requirements, behaviors, and properties of a series of agents and their unique characteristics.

> **(RQ7)** How do the MBSE outputs capture the navigation system architecture, inform the simulation interfaces, and how can these be implemented in software to enable a versatile modular interface?

This research focus captures the ability of the system to operate with independent modules. To answer this question, the framework implementation must exhibit a robust interface between different modules and its implementation be able to draw upon multiple potential libraries and functionality cores. Additionally, it must address how the solution can be implemented in software to allow for these capabilities, and can drive the programming language specification.

> **(RQ8)** How can the specific conceptual models be integrated into a common conceptual framework and used to define the data and input/ouput software implementation interfaces?

In order to provide insight and define the system interfaces, the framework implementation needs a defined interface to data capture. Additionally, the research must address

how the defined models transfer to input decks that can be used by the simulation interface in analysis. This capability will provide a link between the modeling tools and the software implementation to give insight into the data definitions and input parameter properties.

> **(RQ9)** What methods can be used to capture the internal behaviors and algorithms that form the analytical core of the framework and how do these be integrated with the simulation implementation?

This part of the research addresses how the proposed methodology can be used to describe and model the internal behaviors of the simulation elements and their relation to the higher level system activities. This analysis must determine what analysis tools can be used to capture these models. Additionally, this research must address how these models can be used as a baseline and documentation of the implemented algorithms, linking the simulation implementation to the modeling framework.

> **(RQ10)** How is the simulation framework executed in order to capture verifiable performance of a navigation system of interest and provide design space exploration capabilities?

This focus addresses two main aspects of the implemented simulation package. First, the system itself must be verifiable to provide confidence in its analytical result as well as tie the individual modules to external analysis packages. This can also be performed by demonstrating the physics-based implementation of the performance and simulation modules. The second part of this research question focuses on how the model is used to capture system performance of a given scenario. In order to provide data on the design space to flow into trade studies, the simulation package must be able to perform sensitivity analysis and use data simulation techniques to generate system performance data to enable design space explorations and trades.

> **(RQ11)** What data analysis techniques and optimization tools can be linked with the framework to allow for design analysis and parameter optimization?

Another key capability of the framework implementation is the integration and use of optimization packages. These are needed in order to provide for additional trade studies and analysis to feed into parameter estimation methods and to provide for determining optimal measurement frequencies and structures to minimize the need for state updates. The framework and its implementation must be robust with well-defined interfaces to allow for integration of these tools. This is also needed in order to provide for analysis of multiple observation types with optimal update frequencies.

These research questions, which are derived from the hypotheses given above, establish the requirements and capabilities of the navigation analysis framework. The results of the derived analysis mentioned above will provide for verification of the functionality of the implementation. The development of this framework is described in detail in Chapter 5 after an introduction to standard analytical methods in orbit propagation, timing analysis, and state estimation in the next chapter.

CHAPTER IV

SPACE NAVIGATION ANALYTICAL BACKGROUND

Accurate representation of the dynamics of the spacecraft allows for realistic analysis and simulation of the spacecraft along its trajectory. Additionally, this serves as the basis upon which a navigation technique's performance can be observed. For many cases, improvements in an estimator's implemented equations of motion will cause an increase in accuracy of the propagated parameters, allowing for improved tracking of error values and a reduced requirement on state updates. Several key systems anchor the framework's algorithms and tie in the problem's underlying physics to navigation system. These areas are presented to provide the reader with an understanding of the analytical assumptions and provide documentation of the standard underlying physical models. The standard literature in orbit determination and propagation, and state estimation serves as the primary references of this material. Unique to this work is the development of navigation packet-specific measurement models which are presented in Section 4.7. The following sections will discuss the standard state propagation methods, state estimation approach, measurement methods, communication link design, and packet processing algorithms utilized in space mission design with references to the standard references.

4.1 Analysis Frame

In order to develop a physics-driven framework to simulate the motion of multiple spacecraft, one must first define the primary coordinate frame for the navigation system of interest. For the design, the frame is assumed to be J2000 with the sun acting as the central body, due to the large focus on performance during interplanetary flight cruise segments. To match the formatting used by the available data sets, the base time is set to be the Barycentric Dynamical Time (TDB) relative to the J2000 epoch, 12:00:00 GMT on January 1, 2000. This timescale includes relativistic effects, and captures what a highly accurate clock at the Sun's center would measure. The units of time is thus seconds past this epoch. The reference

coordinate frame is defined by the orientation of Earth's mean equator and equinox at the specified epoch [79] [114]. The combination of these two form the basis of the analytical frame of reference.

4.2 State Propagation

In order to predict the future position of a vehicle in space, state propagation methods are used to integrate the equations of motion. These integrate the applied forces on the body to capture the body's dynamics. The primary inertial force on the spacecraft is due to the gravitational effect of the considered planetary bodies. An analytic model is required to accurately capture the dynamics of the trajectory. This is used both in onboard propagation and in the state estimation to form a description of the equations of motion. The basic form of the gravity equation is given in Equation 7. The dynamic models developed in the application follow standard orbital dynamics perturbation techniques such as found in [124] and [7].

The total modeled gravitational force on the spacecraft is the sum of the central body's effect plus that of additional third bodies. In the derived models, the mass of the spacecraft is assumed to be negligible (compared to the other gravitational body) allowing for the use of the predefined gravitational constant, μ, of each external body. The integrated equation for total gravitational force, F, is given in Equation 8. CB represents the properties of the central body and the summation term iterates over i third bodies.

$$\mathbf{F}_{body} = -\frac{\mu_{body}\mathbf{r}_{body\rightarrow sat}}{r^3_{body\rightarrow sat}} \tag{7}$$

$$\mathbf{F}_{total} = -\frac{\mu_{CB}\mathbf{r}_{CB\rightarrow sat}}{r^3_{CB\rightarrow sat}} + \sum_{i=1}^{N}\mu_i\{\frac{\mathbf{r}_{sat\rightarrow i}}{r^3_{sat\rightarrow i}} - \frac{\mathbf{r}_{CB\rightarrow i}}{r^3_{CB\rightarrow i}}\} \tag{8}$$

Additional terms that will play an important part of the analysis are captured by the Jacobian, which is the derivative of the forces with respect to the individual states. These definitions form the basis of the State Transition Matrix, which is used to propagate errors from one time to the next. These are derived from the dynamic equations of motion. These terms are primarily used in the state estimation techniques to propagate the covariance

matrix. The two main terms considered are the derivative of the acceleration with respect to the current position, and the derivative of the acceleration with respect to the vehicle's velocity. These are given in Equations 9 and 10.

$$\frac{\partial \ddot{\mathbf{r}}_{sat}}{\partial \mathbf{r}_{sat}} = \{\frac{-\mu_{CB}}{r^3_{CB\rightarrow sat}} - \sum_{i=1}^{k} \mu_i \frac{1}{r^3_{sat\rightarrow i}}\}\mathbf{I} + 3\{\frac{\mu_{CB}\mathbf{r}_{CB\rightarrow sat}\mathbf{r}^T_{CB\rightarrow sat}}{r^T_{CB\rightarrow sat}} + \sum_{i=1}^{k} \mu_i \frac{\mathbf{r}_{sat\rightarrow i}\mathbf{r}^T_{sat\rightarrow i}}{r^5_{sat\rightarrow i}}\} \quad (9)$$

$$\frac{\partial \dot{\mathbf{r}}_{sat}}{\partial \mathbf{r}_{sat}} = 0 \quad (10)$$

4.3 Dynamic Clock Modeling

Due to the time-based observation process, capturing the dynamic behavior of the onboard oscillator over the course of a simulation is required to fully characterize the navigation performance. The noise in a clock, S is given by Equation 11, which models the spectral density of a clock's frequency fluctuation [17]. In this model, f is the reference frequency and the h represents (in decreasing subscript) white phase noise (h_2), flicker phase noise (h_1), white frequency noise (h_0), flicker frequency noise(h_{-1}), and random walk frequency noise (h_{-2}).

$$S(f) = h_2 f^2 + h_1 f + h_0 + h_{-1} f^{-1} + h_{-2} f^{-2} \quad (11)$$

The primary measure of a clock's dynamic performance is given by the Allan Variance [3], σ^2_y, given in Equation 12. This performance measure captures the fractional frequency performance, y, which is the normalized difference between the observed and the reference signal. This is also referred to as a two-factor variance. This captures a measure of the frequency stability averaged over two subsequent time periods (n and $n+1$) of the same time interval, τ. The term τ represents the length of the time over which the fractional frequency is averaged. This term is used to capture the usage of the clock at a desired operational interval. For example, an oscillator will have different noise characteristics on very small time intervals when compared with averaging over longer periods of time.

$$\sigma_y^2(\tau) = \frac{1}{2}\langle(\bar{y}_{n+1} - \bar{y}_n)^2\rangle \tag{12}$$

The oscillator noise is captured as a series of terms modeling the power spectrum of the noise in the frequency domain. Several parameters are used in characterizing timing sources. The two primary terms used are the white phase noise and the random walk frequency noise, which represent the random shift in the oscillation value and random white noise in the phase of an oscillator. These can be combined in the form given in [125] [87] [13] to develop a dynamic model of the clock error. The dynamic model of the clock states captured in terms of the bias (b), and drift (d), is given in Equation 13. This model shows the considered error terms as applied by normally distributed random variables. The dt term represents the time between subsequent propagations. Additionally, for integration into state estimation algorithms, a model is needed to capture the process noise of the state propagation, Q. The developed form for this is given in Equation 14 [87]. This will be used in augmenting the state estimation routine to allow for clock corrections based on the prediction noise.

$$\begin{pmatrix} b \\ d \end{pmatrix}_{k+1} = \begin{pmatrix} 1 & dt \\ 0 & 1 \end{pmatrix} \begin{pmatrix} b \\ d \end{pmatrix}_k + \begin{pmatrix} N(0,1) * \sqrt{\frac{h_0 8\pi}{dt}} \\ N(0,1) * \sqrt{\frac{h_{-2}2}{dt}} \end{pmatrix} \tag{13}$$

$$\mathbf{Q_{clock}} = \begin{pmatrix} 8\pi h_{-2} dt + \frac{2h_0 dt^3}{3} & h_0 dt^2 \\ h_0 dt^2 & 2h_0 dt \end{pmatrix} \tag{14}$$

4.4 State Estimation

There are a variety of estimation techniques that exist for determining a vehicle's state, given external measurements and accounting for the noise in the observation. Least squares estimation is a common example of this technique. This method selects the variables that minimizes the sum of the squares of the errors between the predicted and measured values for a series of measurements. This method typically requires a linear problem definition, but can be applied to nonlinear systems by using iteration.

When there is knowledge of the errors involved in the measurements and how the errors propagate over time, additional variance-based techniques can be used. The most prolific method for this is the Kalman Filter [48] [23] [50] [108]. This technique utilizes statistical inference of the noise in the measurement with the covariance of the estimated state to compute an optimal state update given the state uncertainty. This is obtained by modeling how the measurement changes with respect to the estimated states, as well as the use of a dynamic model to measure the estimator's error state.

Several variations of this algorithm exist. The particular method of interest for this analysis is the Extended Kalman Filter. The dynamic model is given by Equation 15 This implementation is intended for use in nonlinear systems, where the measurement matrix (the sensitivity of the measurement with respect to the measured states) and the dynamics require nonlinear models for accurate propagation and estimate. For the Extended Kalman Filter, the dynamic model for the state $\mathbf{x}$ is given by a nonlinear function $\mathbf{f}$ which calculates the derivative of the state at an instant in time given knowledge of the state and any external forces. $\mathbf{w}$ is the random noise in the dynamics and is assumed to be Gaussian with a covariance defined by matrix $\mathbf{Q}$, which captures the process noise.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) + \mathbf{G}(t)\mathbf{w}(t) \tag{15}$$

In addition to modeling the dynamics of the vehicle, it is also required to build a model for the individual measurements, $\mathbf{y}$, which will be processed by the estimator. The function $\mathbf{h}$ captures the true measurement in terms of the spacecraft's states and observation error. This noise is captured by $\mathbf{v}$ which is assumed to be a Gaussian distribution with covariance matrix $\mathbf{R}$. The model given represents the system state in the discrete time domain at the $\mathbf{k}$-*ith* measurement.

$$\tilde{\mathbf{y}}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k \tag{16}$$

These models form the basis of the analysis and are used to build up the estimator. Upon initialization, the initial state estimate and initial covariance, $\mathbf{P}$, are set. These capture the

initial state and errors in the current state estimate. The formulas for these are given in Equations 17 and 18.

$$\hat{\mathbf{x}}(t_0) = \hat{\mathbf{x}}_0 \tag{17}$$

$$\mathbf{P}_0 = \mathbf{E}\{\bar{x}(t_0)\bar{x}^T(t_0)\} \tag{18}$$

Between measurements, the system states and covariances are propagated forward in time. This allows tracking of the estimated location over time and provides an initial estimate upon reception of a measurement. The vehicle state is propagated using the user's choice of algorithm to determine the state at the current time. The dynamic model is also used in the propagation of the covariance state. The dynamics of the error is given in Equations 19 and 20, where $\mathbf{F}$ represents the derivative of the equations of motion with respect to the states and $\mathbf{Q}$ is the process noise.

$$\dot{\mathbf{P}}(t) = \mathbf{F}(t)\mathbf{P}(t) + \mathbf{P}(t)\mathbf{F}^T(t) + \mathbf{G}(t)\mathbf{Q}(t)\mathbf{G}^T(t) \tag{19}$$

$$\mathbf{F}(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}|_{\hat{\mathbf{x}}(t),\mathbf{u}(t)} \tag{20}$$

These models are used to capture the state estimate prior to an update, and is denoted as $\hat{\mathbf{x}}_k^-$ and $\mathbf{P}_k^-$. The Kalman gain $\mathbf{K}$ is the core of the state estimation algorithm. This utilizes the estimated covariance, noise, and the measurement model derivatives, $\mathbf{H}$, to calculate the weighting to apply to the state update. The formula for the measurement model derivatives and the Kalman gain are given in Equations 21 and 22.

$$\mathbf{H}_k(\hat{\mathbf{x}}_k^-) \equiv \frac{\partial \mathbf{h}}{\partial \mathbf{x}}|_{\hat{\mathbf{x}}_k^-} \tag{21}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\hat{\mathbf{x}}_k^-)\{\mathbf{H}_k(\hat{\mathbf{x}}_k^-(\mathbf{P}_k^-\mathbf{H}_k^T(\hat{\mathbf{x}}_k^-) + \mathbf{R}_k\}^{-1} \tag{22}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\{\bar{\mathbf{y}}_k - \mathbf{h}(\hat{\mathbf{x}}_k^-)\} \tag{23}$$

$$\mathbf{P}_k^+ = \{\mathbf{I} - \mathbf{K}_k \mathbf{H}_k(\hat{\mathbf{x}}_k^-)\}\mathbf{P}_k^- \tag{24}$$

The state estimator uses the measurement model to calculate the expected value of the observation given the current state estimate. This is performed using the defined **h** function. The state update is thus calculated by applying the Kalman Gain (**K**) to the difference between the observed and predicted measurement. This update gain is also used to update the estimator's tracked covariance states. The formula for this sequence is given in Equations 23 and 24. This functional sequence works to provide a very capable nonlinear state estimation approach that has been implemented for a wide variety of scenarios. This algorithm will form the basis of the navigation performance analysis.

4.5 Measurement Models

Observation-based state measurements form the basis of any navigation technique. Any method used to update the estimated state can be derived down to be one of a few key measurement methods. The standard base observations are state, position, and range. Each of these will be described to capture the models used in the simulation and state estimators.

The state measurement is a direct observation of the current position and velocity of the spacecraft. This would typically be obtained by processing a state update provided by the ground, and would serve as a reset or update to the state estimation procedures. The measurement vector in terms of the current true state, including **r** and **v**, is given by Equation 25. This is shown as measured from the input state estimate. It also includes the effect of measurement error, which is captured as a random variable, **N(x,y)**, from a Normal distribution of mean zero and given standard deviation. When using this equation to generate the onboard estimate of a measurement state, this noise is set to zero. The matrix **H** which captures the derivative of the measurement with respect to the observed states is given in Equation 26. The **R** matrix from Equation 27 captures the noise of the observation captured in σ_r and σ_v. The state matrix includes position, velocity, and clock bias and drift for eight total states.

$$\mathbf{y}_{StateMeasurement} = \begin{pmatrix} r_x \\ r_y \\ r_z \\ v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} N(0,\sigma_r) \\ N(0,\sigma_r) \\ N(0,\sigma_r) \\ N(0,\sigma_v) \\ N(0,\sigma_v) \\ N(0,\sigma_v) \end{pmatrix} \tag{25}$$

$$\mathbf{H}_{StateMeasurement} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{26}$$

$$\mathbf{R}_{StateMeasurement} = \begin{pmatrix} \sigma_r^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_r^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_r^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_v^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_v^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_v^2 \end{pmatrix} \tag{27}$$

The equations for a measurement of position are very similar, relative to current or broadcast current position of the spacecraft. Again the models capture the effect of measurement error individually in each state, assuming normally distributed error with zero mean and given standard deviation. The measurement equation is given in Equation 28 with the applicable **H** and **R** given in Equations 29 and 30. The equations shown also allow for the position measurement to another body.

$$\mathbf{y}_{PositionMeasurement} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix}_{Spacecraft} - \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix}_{RelativeBody} + \begin{pmatrix} N(0,\sigma_r) \\ N(0,\sigma_r) \\ N(0,\sigma_r) \end{pmatrix} \tag{28}$$

$$\mathbf{H}_{PositionMeasurement} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{29}$$

$$\mathbf{R}_{PositionMeasurement} = \begin{pmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_r^2 & 0 \\ 0 & 0 & \sigma_r^2 \end{pmatrix} \tag{30}$$

Multiple observations methods can produce a range measurement. Some examples include measurement of apparent diameter of a planetary body and phase based ranging techniques such as those used in global satellite system navigation. These allow for estimation of a scalar range between two assets. The measurement model for range between the spacecraft (sc) and a relative body (Rel. Body) is given in Equation 31. Due to range being a bulk property of all measurements, it can affect multiple estimated states as seen in the **H** matrix in Equation 32. The noise matrix **R**, Equation 33, is simple due to its single dimensional nature.

$$\mathbf{y}_{RangeMeasurement} = \sqrt{(r_{sc,x} - r_{Rel.Body,x})^2 + (r_{sc,y} - r_{Rel.Body,y})^2 + (r_{sc,z} - r_{Rel.Body,z})^2} + N(0, \sigma_{Range}) \tag{31}$$

$$\mathbf{H}^T_{RangeMeasurement} = \begin{pmatrix} \frac{(r_{sc,x} - r_{Rel.Body,x})}{\sqrt{(r_{sc,x} - r_{Rel.Body,x})^2 + (r_{sc,y} - r_{Rel.Body,y})^2 + (r_{sc,z} - r_{Rel.Body,z})^2}} \\ \frac{(r_{sc,y} - r_{Rel.Body,y})}{\sqrt{(r_{sc,x} - r_{Rel.Body,x})^2 + (r_{sc,y} - r_{Rel.Body,y})^2 + (r_{sc,z} - r_{Rel.Body,z})^2}} \\ \frac{(r_{sc,z} - r_{Rel.Body,z})}{\sqrt{(r_{sc,x} - r_{Rel.Body,x})^2 + (r_{sc,y} - r_{Rel.Body,y})^2 + (r_{sc,z} - r_{Rel.Body,z})^2}} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{32}$$

$$\mathbf{R}_{RangeMeasurement} = \left(\sigma^2_{Range}\right) \tag{33}$$

4.6 Link Analysis

In order to capture the schedule and capability of when two spacecraft can be in communication, a link analysis is performed. This type of analysis considers the amount of energy radiated, as well as the directionality of the transmission and any hardware losses to determine the amount of power at the receiving body. The greatest signal loss is due to the vast distances the transmission travels between assets. The standard equation for these space losses, $\mathbf{L}_{space}$, is given by Equation 34 [73] [74] where λ is the wavelength of the signal, and s is the transmission distance.

$$L_{Space} = 10\log_{10}\{(\frac{\lambda}{4\pi s})^2\} \tag{34}$$

With the defined space loss and including other terms, a formula for the received power, $\mathbf{P}_{Received}$, can be determined. This equation traces the efficiency of the signal through the transmission and reception hardware. The integrated form is given in Equation 35 with $\mathbf{L}_r$ and $\mathbf{L}_t$ representing a line loss, $\mathbf{G}_r$ and $\mathbf{G}_t$ standing for gains, and $\mathbf{P}_t$ the transmitted power. Other losses are typically due to the inefficiency in the signal lines and signal conversion. The units of the final formula are in dBm, with the 30 providing the conversion from dB. t represents transmission parameters while r references reception properties.

$$P_{Received}(dBm) = 10\log_{10}(P_t + G_t + G_r + L_t + L_r + L_S + 30) \tag{35}$$

With an estimate of the received power at the spacecraft's communication interface, additional characteristics of the signal environment can be estimated. These focus on capturing the capability of the spacecraft to decode the received signal. This is largely driven by comparing the received level to the noise floor. This is a factor of the spacecraft's receiver hardware. For a given design, this value is captured in the form of $\mathbf{T}_n$ referred to as the noise temperature. It relates the underlying system noise to the radiation spectrum produced by a black body with a given temperature. Multiplying by this value, as shown

in Equation 36 by $\mathbf{k}_{Boltzman}$, one is able to estimate the noise levels **N**. Combined with the received power levels, the Signal-to-Noise, **SNR**, ratio can be calculated by means of Equation 37 which gives a comparison of the two measures. The capability of a communication system to decode a given transmission into operable data relies upon this value. This value limits the capability of the receiver and drives selection of data rates and coding and modulation schemes to enable data transfer at an expected relative signal level.

$$N = 10 \log_{10}(k_{Boltzman} T_N) \tag{36}$$

$$SNR = P_r - N \tag{37}$$

4.7 Packet Analysis Models

In order for the spacecraft to be able to process the received packets, the algorithms must be developed and presented for the onboard approaches to data processing and generation of state updates from the navigation data packets. For this analysis, the packets consist of two main types of information, transmission time of the packet and transmission location. Two scenarios are presented, one with only time of transmission and one with transmission time and location.

In order to develop communication schedules and to aid in navigation, the spacecraft maintains an estimate of the state of the other known bodies in the network. This allows the agents to predict when it could receive information from other sources and enables development of advanced algorithms for autonomous communication. The range between agents can be measured by the time difference between transmission and reception at the spacecraft. The actual measured signal travel time is used as the observed measurement. Upon reception of a packet with only a transmission time, the agent must further estimate the location of the other spacecraft at the defined time using onboard propagation methods. This information is used in the calculation of the modeled transmission time. Using this location and onboard estimate of the receiving agent's location, the light travel time is calculated, which is the quotient of the speed of light c and the distance between the two

agents. The measurement matrix $\mathbf{H}$ is thus defined by Equation 38. For cases where the spacecraft state is included in the packet, the transmitted state is used instead of the propagated state, and the onboard estimated state of the other spacecraft is updated to match the new value. This allows for improved estimation of the other body's state by allowing for updates of its state from the spacecraft which has a more accurate estimate.

$$\mathbf{H}^T_{RangeMeasurement,Packet} = \begin{pmatrix} \frac{1}{c}\frac{(r_{sc,x}-r_{Rel.Body,x})}{\sqrt{(r_{sc,x}-r_{Rel.Body,x})^2+(r_{sc,y}-r_{Rel.Body,y})^2+(r_{sc,z}-r_{Rel.Body,z})^2}} \\ \frac{1}{c}\frac{(r_{sc,y}-r_{Rel.Body,y})}{\sqrt{(r_{sc,x}-r_{Rel.Body,x})^2+(r_{sc,y}-r_{Rel.Body,y})^2+(r_{sc,z}-r_{Rel.Body,z})^2}} \\ \frac{1}{c}\frac{(r_{sc,z}-r_{Rel.Body,z})}{\sqrt{(r_{sc,x}-r_{Rel.Body,x})^2+(r_{sc,y}-r_{Rel.Body,y})^2+(r_{sc,z}-r_{Rel.Body,z})^2}} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{38}$$

The range measurement can be observed from the packet processing. With the receipt of multiple observations, it is possible to additionally form a range-rate state update. The definition of range-rate is given in Equation 39. The derivative of this formula can be taken with respect to each individual variable to determine the $\mathbf{H}$ matrix for use in the state estimation procedures. The individual formulas for each term are given in Equations 40 through 45 in terms of r and v.

$$\dot{r} = \frac{(\mathbf{v}_{sc} - \mathbf{v}_{Rel.Body}) \cdot (\mathbf{r}_{sc} - \mathbf{r}_{Rel.Body})}{\sqrt{(r_{sc,x} - r_{Rel.Body,x})^2 + (r_{sc,y} - r_{Rel.Body,y})^2 + (r_{sc,z} - r_{Rel.Body,z})^2}} \tag{39}$$

$$\frac{\partial \dot{r}}{\partial r_x} = \frac{v_{sc,x} - v_{Rel.Body,x}}{r} - \frac{r_{sc,x} - r_{Rel.Body,x}}{r^3} *$$
$$\{(v_{sc,x} - v_{Rel.Body,x}) * (r_{sc,x} - r_{Rel.Body,x})$$
$$+ (v_{sc,y} - v_{Rel.Body,y}) * (r_{sc,y} - r_{Rel.Body,y})$$
$$+ (v_{sc,z} - v_{Rel.Body,z}) * (r_{sc,z} - r_{Rel.Body,z})\} \quad (40)$$

$$\frac{\partial \dot{r}}{\partial r_y} = \frac{v_{sc,y} - v_{Rel.Body,y}}{r} - \frac{r_{sc,y} - r_{Rel.Body,y}}{r^3} *$$
$$\{(v_{sc,x} - v_{Rel.Body,x}) * (r_{sc,x} - r_{Rel.Body,x})$$
$$+ (v_{sc,y} - v_{Rel.Body,y}) * (r_{sc,y} - r_{Rel.Body,y})$$
$$+ (v_{sc,z} - v_{Rel.Body,z}) * (r_{sc,z} - r_{Rel.Body,z})\} \quad (41)$$

$$\frac{\partial \dot{r}}{\partial r_z} = \frac{v_{sc,z} - v_{Rel.Body,z}}{r} - \frac{r_{sc,z} - r_{Rel.Body,z}}{r^3} *$$
$$\{(v_{sc,x} - v_{Rel.Body,x}) * (r_{sc,x} - r_{Rel.Body,x})$$
$$+ (v_{sc,y} - v_{Rel.Body,y}) * (r_{sc,y} - r_{Rel.Body,y})$$
$$+ (v_{sc,z} - v_{Rel.Body,z}) * (r_{sc,z} - r_{Rel.Body,z})\} \quad (42)$$

$$\frac{\partial \dot{r}}{\partial v_x} = \frac{r_{sc,x} - r_{Rel.Body,x}}{\sqrt{(r_{sc,x} - r_{Rel.Body,x})^2 + (r_{sc,y} - r_{Rel.Body,y})^2 + (r_{sc,z} - r_{Rel.Body,z})^2}} \quad (43)$$

$$\frac{\partial \dot{r}}{\partial v_y} = \frac{r_{sc,y} - r_{Rel.Body,y}}{\sqrt{(r_{sc,x} - r_{Rel.Body,x})^2 + (r_{sc,y} - r_{Rel.Body,y})^2 + (r_{sc,z} - r_{Rel.Body,z})^2}} \quad (44)$$

$$\frac{\partial \dot{r}}{\partial v_z} = \frac{r_{sc,z} - r_{Rel.Body,z}}{\sqrt{(r_{sc,x} - r_{Rel.Body,x})^2 + (r_{sc,y} - r_{Rel.Body,y})^2 + (r_{sc,z} - r_{Rel.Body,z})^2}} \quad (45)$$

These values form the basis of the **H** vector, allowing for processing of the packet observations within the state estimators. Additionally, after the initial packet is received, the range and range-rate measurements are concatenated together and sent as an integrated measure to the state estimation routine for improved state updates.

CHAPTER V

SPACE NAVIGATION ANALYSIS AND PERFORMANCE EVALUATION FRAMEWORK (SNAPE) CONCEPTS AND IMPLEMENTATION

This chapter focuses on the development and implementation of the Space Navigation Analysis and Performance Evaluation (SNAPE) framework to address the research questions presented in Chapter 3. The goal of this framework is to capture the navigation architecture under study (with an integrated communication capability) and provide an approach to simulation and analysis. First, the models capturing a generic navigation system are presented. Following this, the development of the models for the SNAPE framework are described. Lastly, a software implementation of the framework will be presented in detail, focusing on its operation as well as giving an overview of its functionality. Additionally, the derivation of the implementation from the architecture and framework models will be presented.

5.1 Usage of Model-Based Systems Engineering and SysML

The first set of steps in the analysis approach focus on the development of Model-Based Systems Engineering(MBSE) models. These are constructed to capture knowledge about the navigation system under study as well as to provide a high level analysis of navigation requirements, uses, and structure. The models are used to drive further development and support software implementation.

The initial step when approaching any new system of interest which is not currently implemented is to capture an understanding of the operations of that system. This is done to both inform the proceeding development steps, but to also provide an in-depth review of the system of interest. In this thesis, the two main systems of interest are: space navigation systems, and the conceptual framework used to design and analyze them. Several steps are involved in capturing this knowledge. The method utilized in this chapter is based upon

Model-Based Systems Engineering(MBSE) techniques, as introduced in Section 3.7.1.

In order to take advantage of burgeoning standards in this field, the SysML modeling language will be used. This approach allows for the use of interacting, complex, detailed systems models into a standardized format, or series of views, that are used to provide insight into the system. This study takes advantage of this modeling language to capture knowledge about a conceptual navigation architecture, which will form the basis of the packet-based navigation implementation. Another key driver in using SysML is the array of software packages that allow for the efficient and effective definition, storage, and visualization of these models. In particular, this study utilizes Sparx Systems Enterprise Architect[1] to capture the views of the system. This package will be used to capture system requirements, use cases, interactions, and the system definition to provide high level information views into navigation system design and architecture. The selection of SysML also provides future capability in the form of an executable model, in which it can directly interact and drive the simulation interface. Integrating the system models with the analysis software creates a complete set of tools to capture the system under study and tie the performance directly back to the requirements and definitions.

The SysML specification[86] defines multiple modeling language elements that can be used to capture the properties and development of the system of interest. These elements are computer-readable with defined semantics. They are typically stored in a database format and managed by a SysML tool (such as Enterprise Architect mentioned previously) that can both edit and create entries in this database. These can also be exchanged between editors in a standardized XML clear text format known as XMI[85]. SysML defines how these elements can be presented graphically using nine different types of standard diagrams. The main ones used in this work will be briefly described here, and more detail given for each presented model in the following sections.

The Requirements Diagram captures the requirements of the system, graphically describing their hierarchy and structure. Additionally, links can be defined that show the

[1]http://www.sparxsystems.com

relationship between various inputs, allowing for capture of derived, composed, and associated requirements. This diagram can also be used to link specific use cases, test cases, and system components to identify their ability to satisfy requirements. Therefore this diagram can used to both define and validate the requirements, providing traceability in their definition and linkage to system components and structure.

Use Cases are defined by SysML to capture the actors and high level relationships between external agents and internal structure. This allows for understanding and description of how the system would be used and work with external assets. This works to capture a generic concept of operations of the system and helps to identify stakeholders and external interfaces.

The development of Sequence Diagrams define these relationships between the system of interest and external agents in more detail. These are driven by the use cases and aim to provide further insight into the actual sequences of actions between various elements over the course of system operation. This defines in detail each interaction between the defined elements. Each is placed into a separate swimlane, or vertical structure, and arrows between lanes identify functional interfaces. The interactions are ordered in sequence from top to bottom, providing an overview of the sequence of events between individual elements that define a certain action. This allows for a ABM-based approach to capturing and defining specific agent behaviors and capabilities.

By integrating several Sequence Diagrams, it is possible to identify the states of a component system in terms of its interaction with external commands. This is captured through the use of State Machine Diagrams. These capture the various states of a subsystem. External functions interacting with the system initialized various state transitions based on the specific action and the current state. This allows for tracking of a system's various ABM-driven roles and operational modes. This is increasingly important when defining complex behavior, as external or internal actions on the system can change its behavior and functional mode.

With the operations and functionality of the system identified, Block Definition Diagrams can be used to capture the components of the system. This allows for allocation of

the functional requirements and interfaces to specific subsystems, or blocks. The identified blocks form the components. Within this diagram it is possible to define part composition, aggregation, multiplicity, and generalization, allowing to define the overall structure of the system's components. Additionally, each block can have defined parameters and operations. These directly tie into the properties of the component, and reference back to the functions identified. Additionally, when developing a software model of the system, this directly defines the object structure to be implemented to allow for simulation of the system.

Internal Block Diagrams are defined to capture the interactions and connections among components within a system. By defining the interfaces and ports of the various blocks, this model captures their linkages and flow of information between the individual elements. Additionally, this model captures the input and output of the entire system and allows for identification of the interactions among system components. This diagram also serves to capture the interfaces between components, providing further documentation and insight into the simulation process as well as the internal structure of system of interest being designed.

With the system defined, and the functionality allocated to specific blocks or interfaces, it is possible to revisit the system requirements. The Requirements Diagram can again be used to capture the relationships between the system structure and its high level needs. In this model, the individual requirements are linked to the component that meets, addresses, or satisfies that need. This allows for further identification of the purpose of each system component and additionally ensures that all system needs are met. Building upon this general description of the various SysML models being used in this research, the application of each to deep space navigation and framework development will be discussed.

5.2 Modeling of Generic Space Navigation Systems

The first step in the analysis process is to define the requirements of a generic deep space navigation system that integrates both communication and traditional measurements. This is to ensure the capability to capture the NNAV concept of operations. The general high

level requirements are given in Figure 15[2]. Each block's stereotype, included within the angle brackets $\ll$ and $\gg$ defines the type of that requirement. The model identifies three high level nested requirements within the overall navigation system requirement package, where each nesting is identified by the cross-hair ($\bigoplus$) symbol. This captures a requirements view of the system. It can be seen that the main functions of the system are to estimate the spacecraft's state, propagate the state, and process any incoming measurements or packets. The performance of the system to meet these requirements determines how well the performance requirements are met, in regards to the allowed estimator error levels.

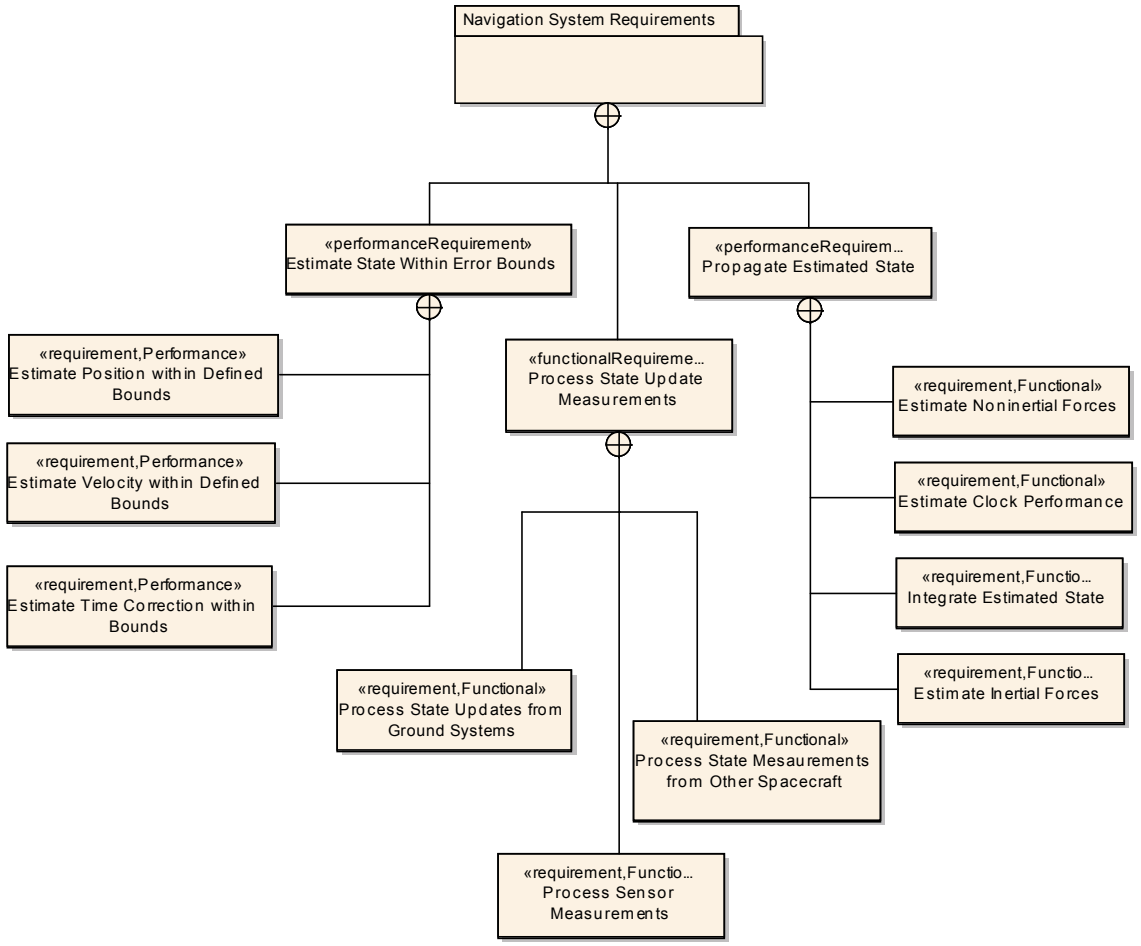


Figure 15: Navigation Systems Requirements (SysML Requirements Diagram)

[2]To meet the defined standard[86], SysML diagrams typically contain a diagram frame and title in the top left. These have been removed from the presented models to improve readability. The type of model is instead included within each diagram's associated caption to address the SysML requirements.

Each nested requirement can be broken down and considered to be the aggregation of several lower level needs. These links are identified by the dashed arrows that define a derived requirement. These capture how the requirement Propagate the Estimated State is the culmination of being able to estimate noninertial and inertial forces, estimating clock performance, and integrating the estimated state. This describes the structure and hierarchy of the system's need and allows for breaking high level performance requirements into the functional requirements that are required to enable those system capabilities. This additionally allows traceability and identification of the low level requirements at a functional and component level, aiding in system definition and structure.

The next step is to define how the system will be used and its interaction with external bodies, such as other spacecraft, ground operators, or even other onboard systems. These are all included as actors in the use case analysis, represented graphically by a stick figure as seen in Figure 16. This model is used to capture the various uses of a system to support functional allocation and external interface identification. The onboard Command and Data Handling (C&DH) Module and Fault Management Module are identified as external users of the navigation system. These are defined as being the interface layers between onboard fault detection and isolation routines as well as providing the software connection from the navigation module to sensor and communication subsystems. The main Use Cases considered in this analysis are the processing of external commands, sensor measurements, and communication packets. The uses are identified in the Navigation System block by labeled ovals.

Figure 16 describes the high level interaction of the navigation system with external modules, focused on the commanding aspect. The main users of this functionality of the navigation system are Ground Control, C&DH, and Fault Management. The specific activities captured in this diagram include the initialization and update of the spacecraft's estimated state as propagated from the ground. Additionally, it captures the capability of the fault detection system or the external commanding to trigger a filter reset condition, causing a re-initialization of the navigation state. The interfaces and triggering of actions is captured through the lines denoting association between users and uses and between

users. An example of this is the identified interface of Ground Control to C&DH, where commands are transmitted from the ground to the spacecraft which then interacts with the Navigation System to either set the state, propagate the state forward, or reset the state. Comparatively, the Fault Management Module operates independently of C&DH and Ground Control, with the capability to reset the onboard state to a pre-programmed value in case of system fault detection. This provides a high level view of the uses of the navigation system, with a focus on its state update functionality and external interfaces to other systems.

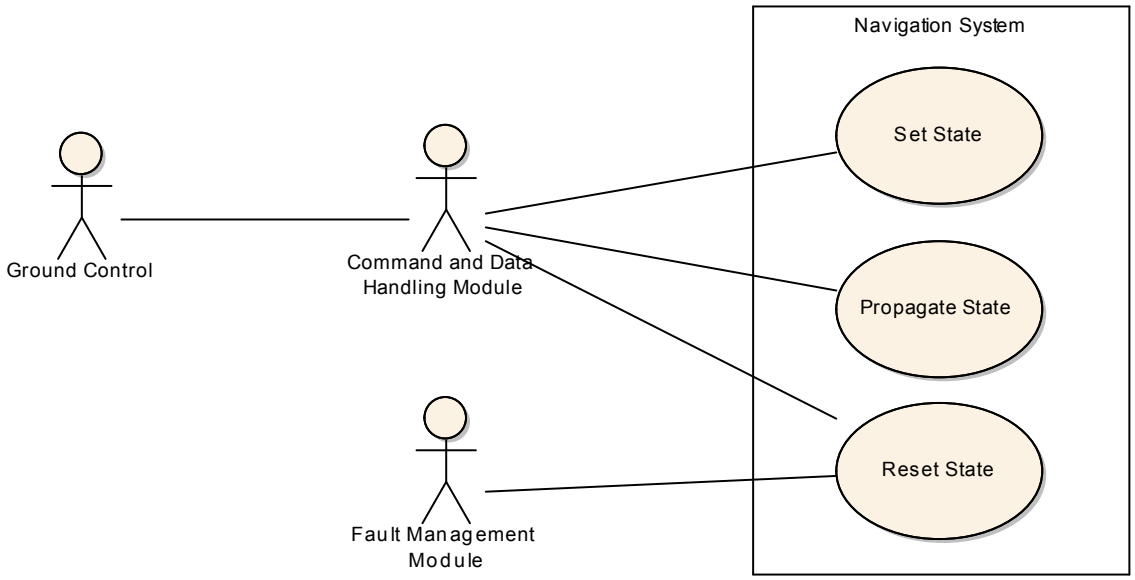


Figure 16: Navigation Measurement Processing Use Cases (SysML Use Case Diagram)

The second set of use cases, depicted in Figure 17, captures the operation of the navigation system in regards to processing various navigation update information. For this analysis, the users are identified as the Navigation Sensor directly to the Navigation System, and Other Spacecraft and Ground Control interfacing through the C&DH Module. This shows the scenarios of a sensor providing a state update to be processed by the navigation system. Additionally, it captures the reception and processing of navigation packets and state updates. These state updates are envisioned to be generated by either other spacecraft in the navigation network (such as the relays mentioned previously) or ground

assets (such as a DSN-based state update or transmitted packet). The identified uses of the navigation system capture packet and measurement processing. These use cases provide a high level view of the activities of the system and their interfaces with external agents in a graphical manner. This allows identification of associations and functional flow between individual elements and description of the operational concept of the navigation system in terms of external users and high level functional needs.

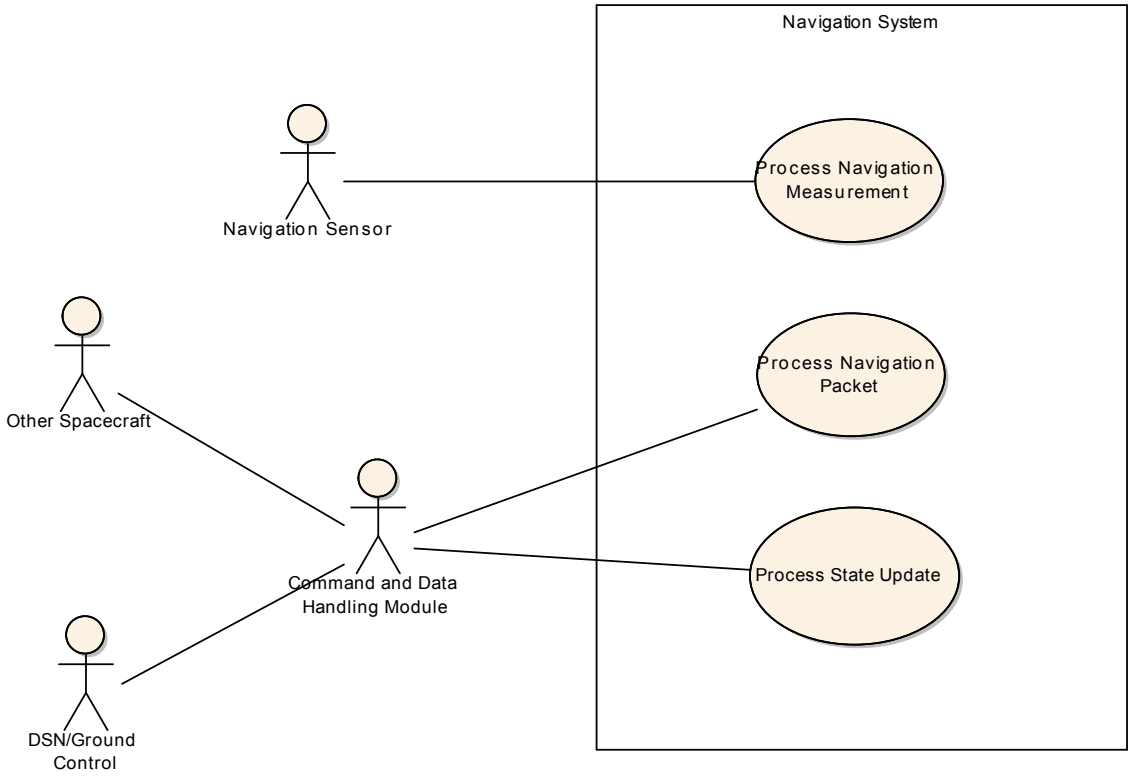


Figure 17: Navigation Packet Processing Use Cases (SysML Use Case Diagram)

Upon identification of the use cases of the system, the next step is to describe these in more detail and define in greater detail the functional interfaces between the various subsystems. This helps to identify the functional requirements of the system and lays out the initial groundwork for the interface design. This data is captured in the form of interaction diagrams. These capture each agent in a unique 'swimlane' (vertical structure) and defines a sequence of actions that occur as part of a use case. Due to the analytical focus on the navigation performance and filtering, the state update and propagation use cases are

captured via description of measurement and packet processing interaction diagrams.

The first use case, given in Figure 18, describes the sequence of activities that are used to perform the processing of the navigation packets and captures the interfaces between a Navigation Data Source, C&DH, and the Navigation System. This model captures all of the steps in the correct sequence required for successful state update from received data. The blocks at the top of the diagram each represent an individual users or system. Dashed lines represent the actions assigned to each element. The rectangles capture the active systems, and the arrows represent functional calls that point towards the element on which the function operates.

First, the packet is generated by the source agent. For this diagram the Navigation Data source can be a ground station or satellite. The packet is shown to be generated after link analysis is performed to check if the two objects are in communication range. Next, the navigation header is generated, and the data transmitted. After the light travel time delay, the packet is received by the C&DH software, which confirms receipt, and transfers the data to be processed by the navigation system. The navigation module then proceeds to process the measurement (decomposing and identifying the information contained in the header, such as source location and transmission and reception times. Following this, the navigation system propagates its estimated state to the time of packet arrival, calculates the modeled measurement (what the navigation expects the measurement to be based upon its current state estimate), and then generates the state update. The onboard state is then updated and this resulting state is returned to C&DH to confirm analysis is complete. This formulation assumes a sequential filter implementation that process each measurement individually and calculates a state update. For batch filtering methods (such as a least squares analysis), the process of generated a state update would require more detail.

The use of this interaction diagrams captures the order of actions in an operational scenario and identifies the specific functional roles of each system. Additionally, this model is used to capture the needed functions and begin to track their arguments to feed into interface design. As opposed to a psuedocode textual description or a simple list of activities, this model captures all of the interacting agents, their functions, and data interfaces in an

intuitive design allowing for complete description of the activities required to perform this function, specifically packet generations. This serves to capture in detail the system's concept of operations.

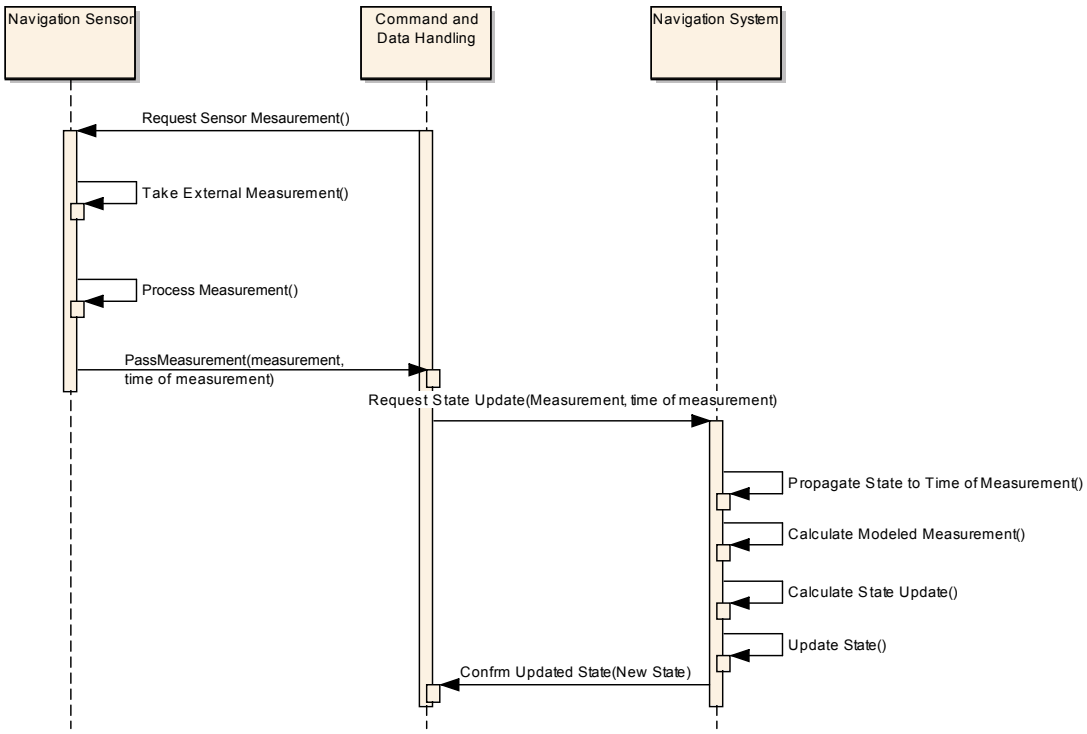


Figure 18: Navigation System External Packet Processing (SysML Sequence Diagram)

The other use case, presented in Figure 19, describes the processing of a state measurement from an onboard sensor. For this case, the C&DH module acts as the initiator, requesting the latest reading from a sensor at a specified interval or time. The sensor then takes an observation, and applies any known correction and calibration terms, and returns the value to the data handler. The data is then sent to the navigation module which propagates the onboard state to the measurement timestamp, calculates the modeled measurement at the observation time, calculates a state change, and updates the estimated state parameters. The navigation system concludes by confirming processing of the measurement back to C&DH. These models allow for capturing specific behaviors that form the foundation of ABM.

As mentioned above, this focuses on a sequential filter, and makes no assumptions about the time between measurements or the number of measurements. These conceptual models can be used to capture the behavior of a range of filters operating at a range of time scales. Regardless of the implemented state estimator, these same processes will be followed. This allows for a generalized view of the system's functionality and operation, providing a stereotype for developing modeling and simulation tools to capture this sequence of events. This model is very similar to the use case presented above for external packet processing, and allows the designer to identify common as well as unique functionality of each system.

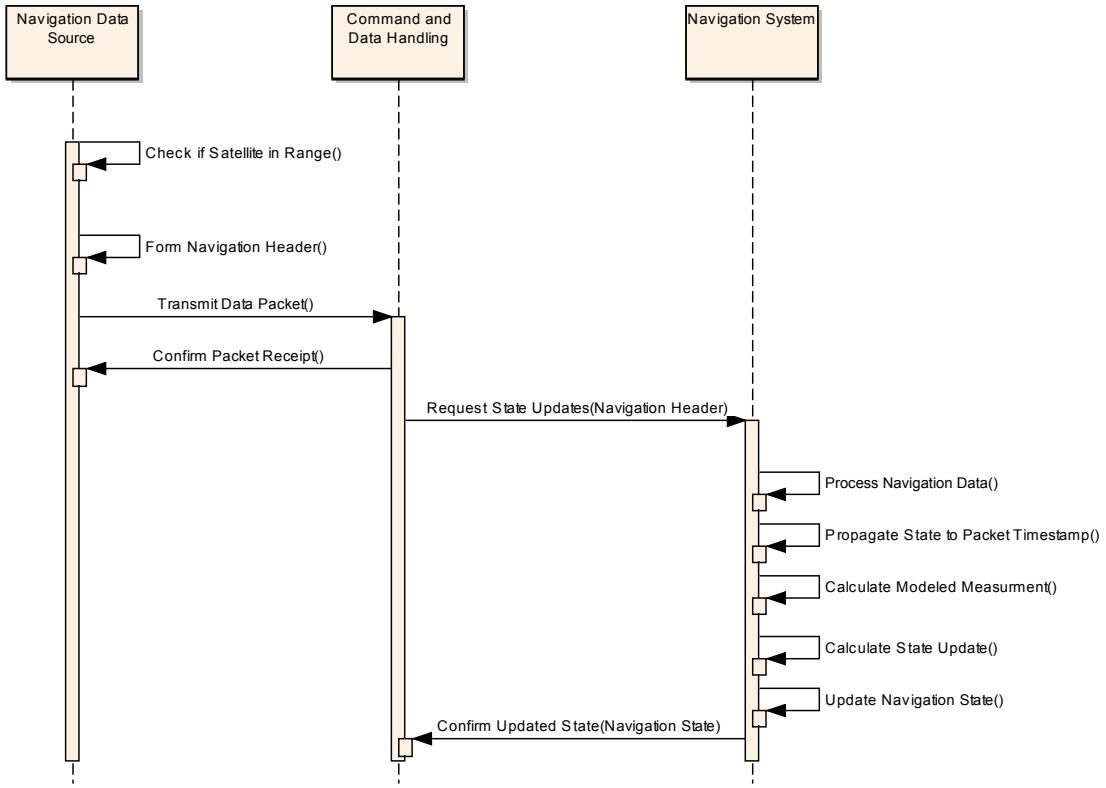


Figure 19: Navigation Measurement Processing (SysML Sequence Diagram)

The modeling of the system's interactions and functional interfaces with external agents forms the basis of understanding of the system's operational states. These are captured in more detail through the use of a State Machine Diagram. This model captures the flow of the navigation system behavior and modes of operation. The implemented model is

shown in Figure 20. The various modes are given in the diagram, along with identification of transition events and their state. Each mode of operation is identified by a defined blocks. The arrows identify state transitions from the source to resulting state. Each state transition is linked to a specific function or action being performed on or within the system, each identified within square braces. The initial state is identified by the solid circle with the initial action of [Power On] transitioning the system to an initialization mode. Additionally upon the [Power Off] action, the navigation systems transitions to the final exit state.

This captures the system behavior from power up to power down, with included fault analysis events included. Upon power up, the estimator variables are initialized. Once these values are loaded from memory, the state initialized to its last known estimated location (or built in zero-point). Upon completion, the system proceeds to an idle state awaiting for further events.

The state machine model captures a range of events which drive the system's operation. Upon detection of a failure event, the estimator is re-initialized. At a certain interval, the C&DH may generate an interrupt, requesting a state propagation to calculate a new estimated state at some specific state. Upon reception of a packet from C&DH, the estimator proceeds to process the packet and if it is determined to be valid, adhering to the format standards, a state update is calculated. Measurement updates operate in a like fashion, checking the measurement for validity, and proceeding to calculate and apply a state update upon a successful observation. All of these events tie back to the navigator's IDLE state, upon which it is waiting for further events. This model helps to capture the event-driven behavior of the navigation estimator, processing packets and measurements as available, and propagating the internal estimate at some externally controlled interval. These define the operational states of the navigation filter, and builds a framework of the behaviors of the model, defining specific operations to perform upon certain events.

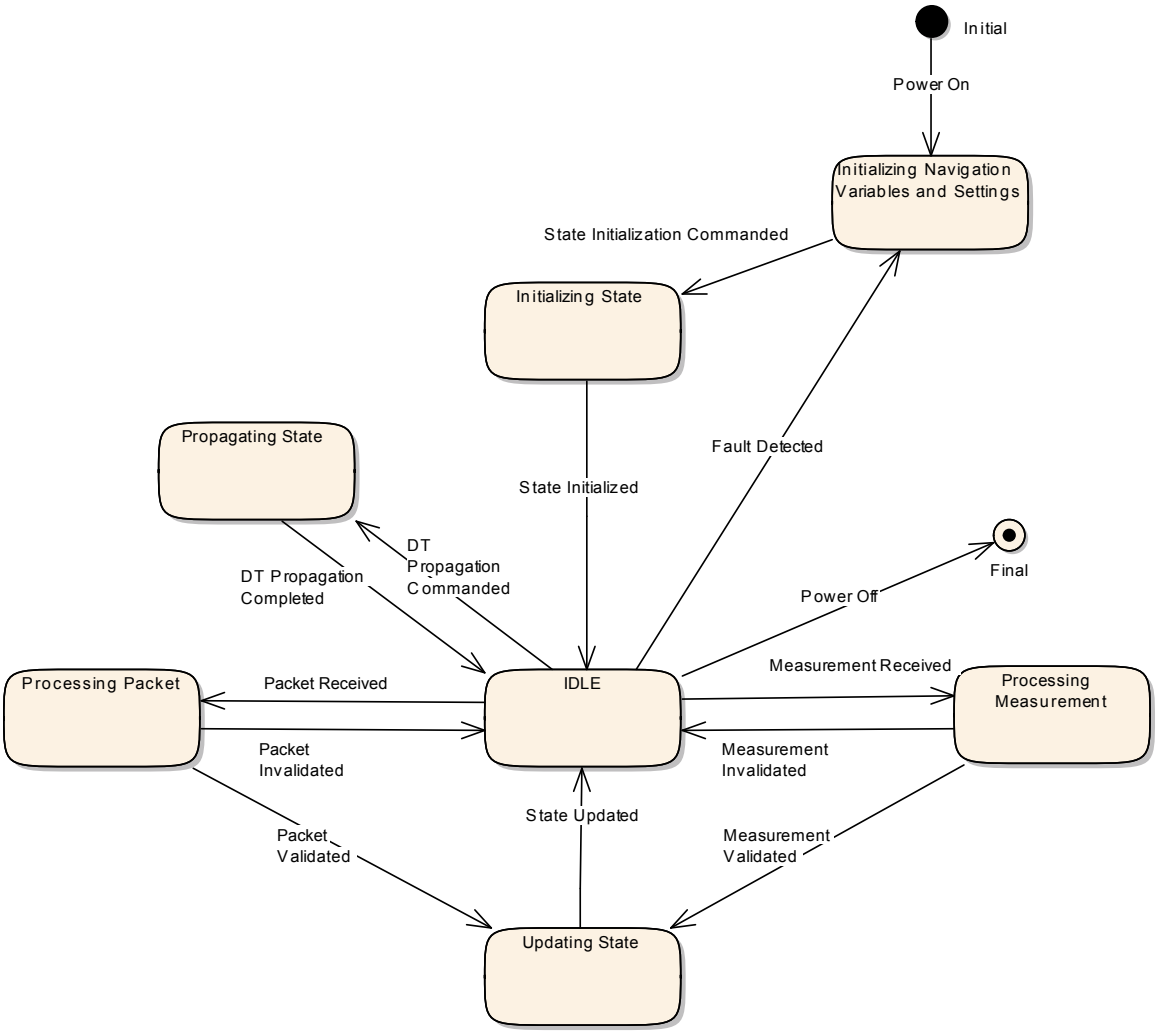


Figure 20: Navigation System Operational Modes (SysML State Machine)

With the behaviors of the system captured and the interactions described, a detailed view of the system under study and its operational modes and requirements has been captured. The next modeling step is to break the navigation system down into individual components and begin to assign functionality to each. This breakdown allows for an understanding of the subsystems that must interact and interface to build up the complete navigation system. This has two main parts, the Block Definition Diagram and the Internal Block Diagram. The first diagram is used to define the breakdown of the system, including descriptions of the parameters and operations associated with each component. This model is shown in Figure 21.

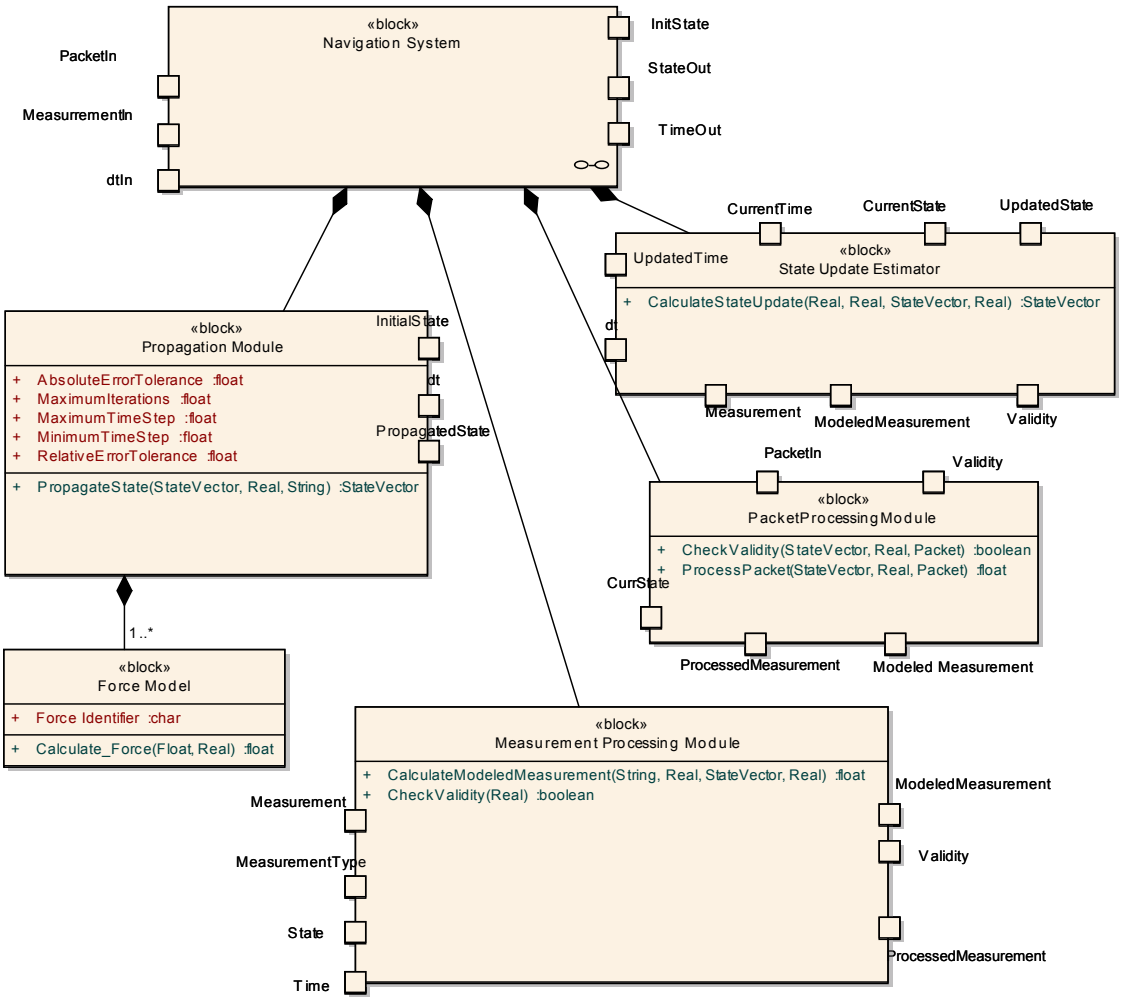


Figure 21: Navigation System Structure (SysML Block Definition Diagram)

This model represents the Block Definition Diagram of the system. The specific components are each captured by the individual blocks. Each block allows for identification of its parameters and operations, as well as their types and interfaces within the object. Flowports can also be defined for each block, allowing for definition and capture of the data flow into and out of each object. These can be identified as hollow boxes attached to a block. This model provides for insight into the structure of the system under study. The lines with ♦ represent a composition relationship of a black with the symbol attached to the higher level object. This captures how the combination of individual elements build to form an integrated system. Additionally, it is possible to identify the elements that form

the aggregates parts of a object. This is identified by lines ending with ♢ attached to the host block. The multiplicity of the item can be captured as well, identifying how multiple part of one type are integrated into the block.

The navigation system is broken down into its primary components, described as the Propagation Module, State Update Estimator, and the Packet and Measurement Processing Modules. Each of these objects perform specific roles as defined in the interaction diagrams given, performing such functions as measurement validation, propagating the estimated state forward (or backward) in time, and calculating a state estimate. Each of these operations is allocated to specific blocks and identifiable in this model. Additionally, through the use of the aggregation relationship, the model captures the inclusion of multiple Force Models within the Propagation Module. The attached label '1..N' captures this multiplicity, stating that the overarching module includes between 1 and N individual Force Models. This diagram serves to visualize the composition of the navigation system and the properties and functions of its components. The displayed interfaces on each block also allow for the visualization of inputs and outputs of each element.

The Internal Block Diagram builds upon the Block Definition and models the internal structure of the defined systems. The defined blocks and internal structure directly feed into this model by forming the base structure. A central purpose of this diagram is to capture the linkages and interfaces between elements that compose a system. As such, the individual components of a system are placed within a large central block that represents the higher level object. Ports on this outer block capture the inputs and outputs of the system. Similarly, the ports of each individual element are displayed. Associations between elements can be visualized by arrows connecting ports. Capturing these relationships allows the user to identify the flow of data within the system, from inputs through internal blocks and to outputs. This gives an internal view to the data processing sequence and allocation of data and functionality within the system.

The internal structure of the navigation is captured in Figure 22[3]. This allows for

[3]The SysML Specification defines that the flow of data into or out of a port is defined by an arrow on the port itself. To improve readability, the flow of information is instead shown as arrowheads.

further refinement and definition of the navigation system composition. The model enables visualization of the flow of data between elements in the system of interest. This information is important in the next level of development by capturing specific functional blocks to be implemented, as well as input and output contents of each's operations. By visualizing the flow of information between elements, it is also possible to capture the sequence of internal operations and build algorithmic blocks to form a basis for implementation and further analysis.

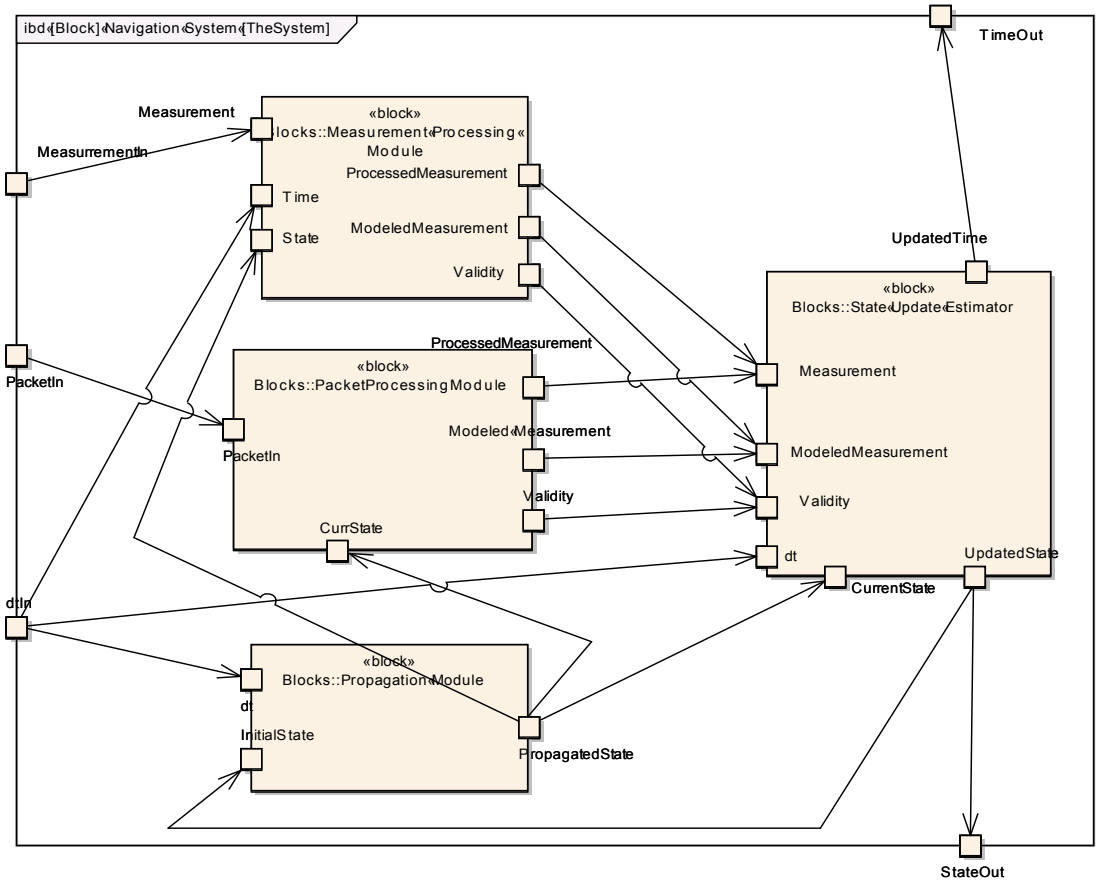


Figure 22: Navigation System Internal Structure (SysML Internal Block Diagram)

With the system defined and the internal structure developed, it is important to tie

these models back to the high level requirements. This can be done through another use of a Requirements Diagram. Mapping the system requirements to the base blocks is shown by the model given in Figure 23. This diagram includes the system's defined blocks and captures how each element relates to the system requirements. This is used to ensure that all of the required aspects of the system are implemented and aids in requirements verification and analysis. The associations in the figure represent the ability of a block to satisfy a specific requirement. Ensuring all requirements are being addressed by an element of the system allows for traceability and reduces the potential for needs left unaddressed by the system.

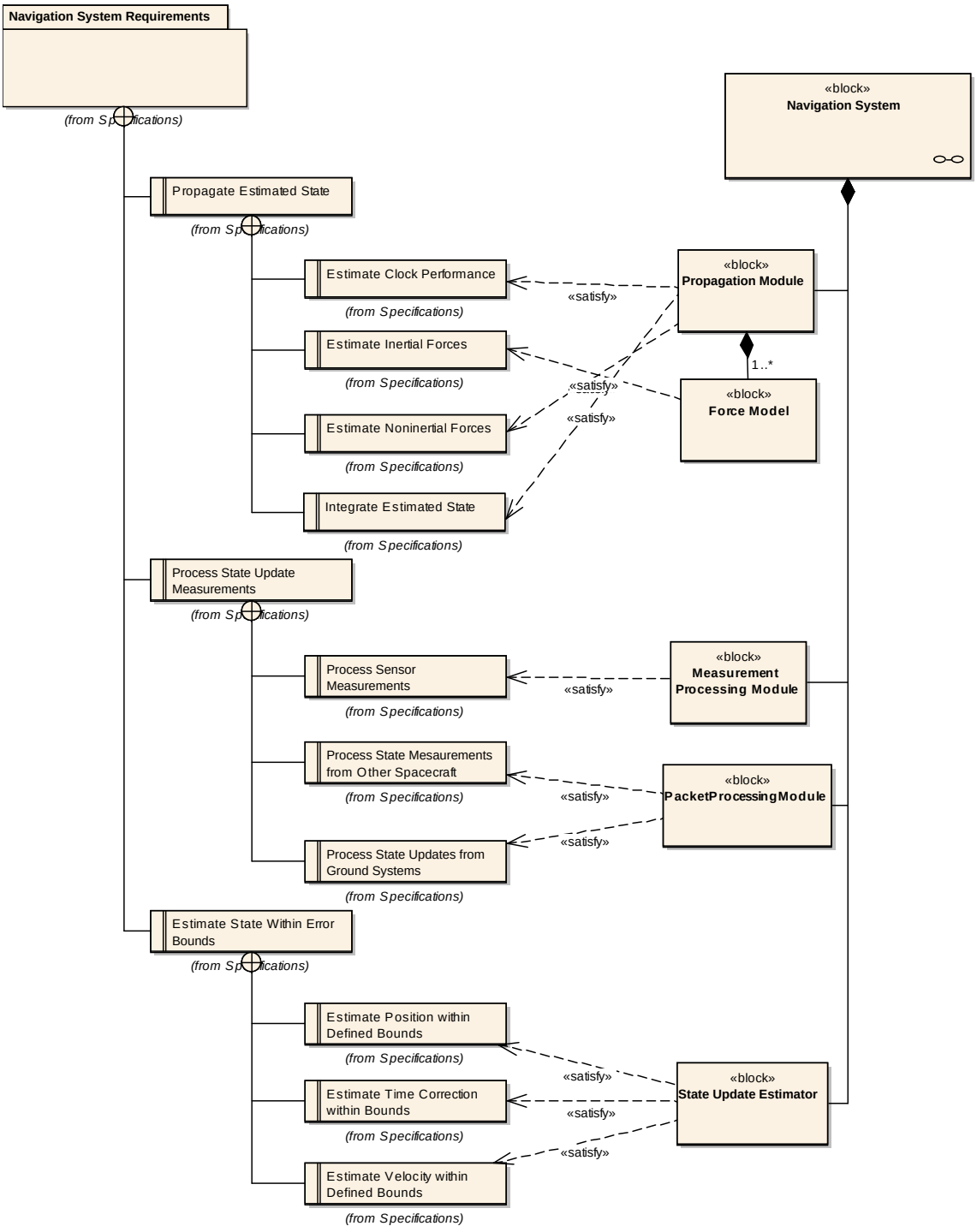


Figure 23: Navigation System Requirement Satisfaction (SysML Requirements Diagram)

With these models implemented, the designer is able to get a detailed understanding of the system under study. The requirements diagrams show the needed capabilities of

the system, and how these track to the individual vehicle subsystems. The use cases and interactions diagrams capture the operation of the system and the interfaces to external elements. Lastly, the block definition diagram and the internal block diagram show the composition of the navigation system, and the linkages between structural elements. These models all work together to give a total view of the system under study and form the basis of approach for the analysis of the system.

5.3 SNAPE Architecture Design

To capture the analytical framework used to evaluate the performance of deep space navigation systems, the designer must first define the requirements of the simulation tool and the experiments it must perform. These are derived from the models of the conceptual navigation system. The framework is implemented to mirror the real system and provide a simulation analogue for experimentation and analysis. As such, its goal is to duplicate in software the functional purposes of each block as well as the defined interfaces, providing an implementation for each operation. The use cases and the appropriate interaction diagrams serve as algorithmic outlines of the operation of the simulation.

The primary requirements are derived from the research questions which the framework must address. These in turn have a basis in the system's overall requirements. The simulation environment's requirement diagram is given in Figure 24. As shown in Figure 25, these all act as analogues to subsystems within the navigation system. This model allows for understanding and documenting the linkages between the requirements at the system and framework levels. This provides insight into the design and needs of the framework, by identification of the requirements flow between the two.

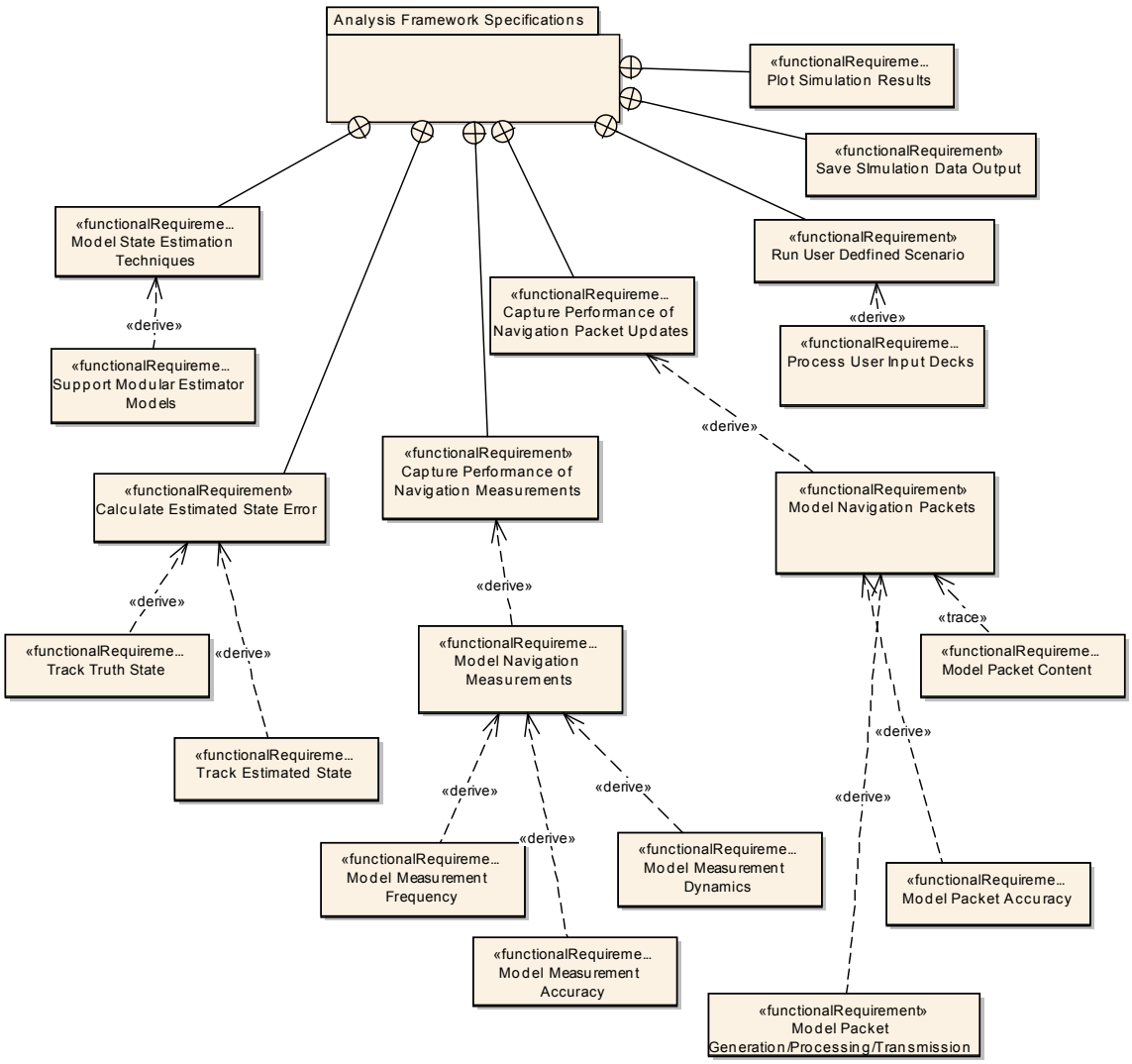


Figure 24: SNAPE Framework Requirements (SysML Requirements Diagram)

Taking the same approach as for the overall navigation system, the Navigation Framework Requirements are decomposed into lower level functional needs. The modeled relationships capture the linkages between requirements at multiple levels. In order to capture the performance of the state estimator, the framework must be able to model all aspects of state propagation, measurement estimation (both true and modeled), and estimation.

Additionally, unique requirements are placed on the system to allow for the design analysis to take place. For example, the software interface must allow for user-defined input, output of performance and state data to file, graphical depiction of simulation results, and

analysis of a wide range of measurements and state estimation algorithms. The capability for the system to operate with external optimization and design space analysis packages is inherent within the need to analyze any navigation packets and provided optimal performance. The integration of these requirements from both the operational and analytical points of view allows for complete modeling of the requirements of the framework.

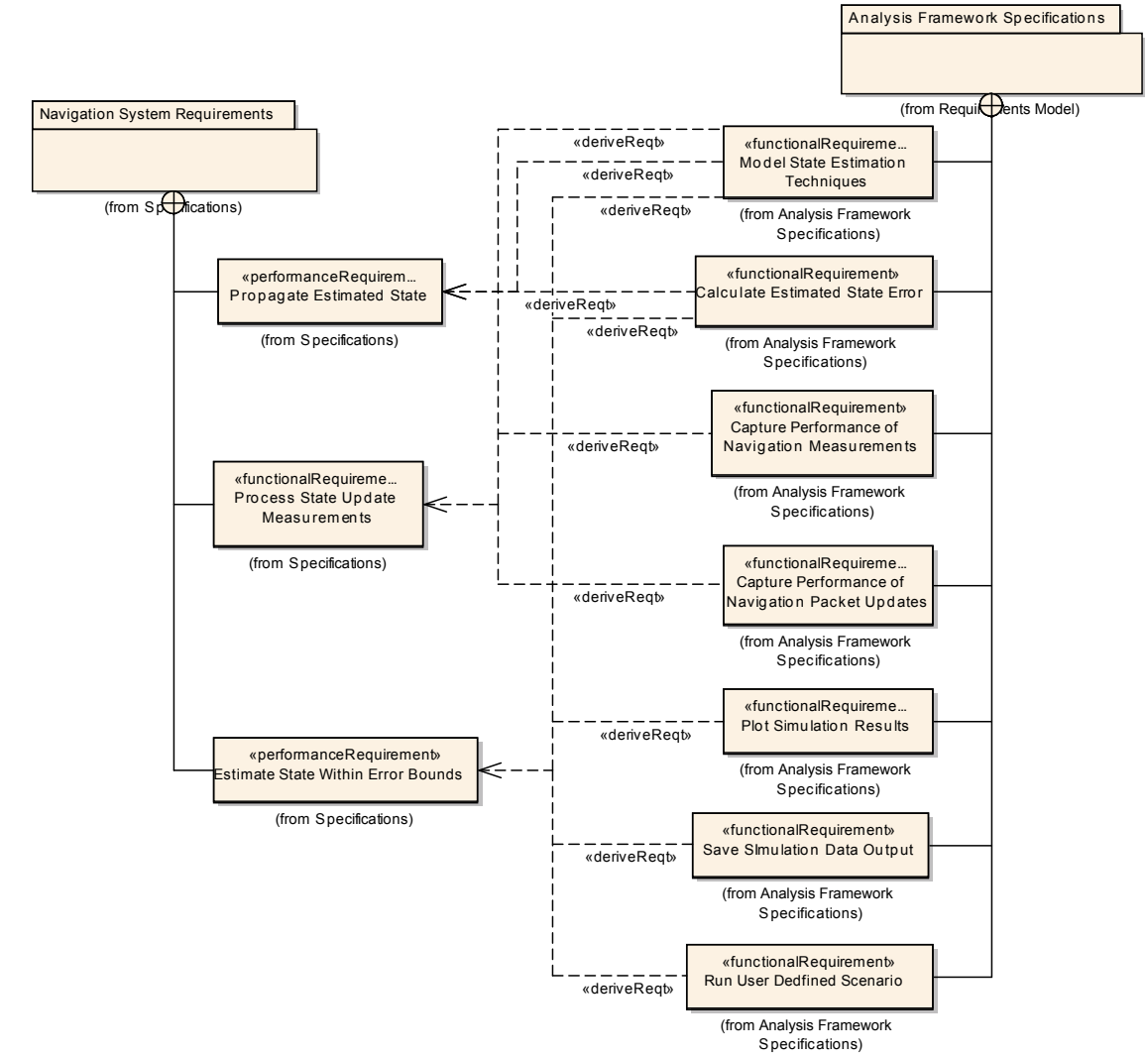


Figure 25: Relationship Between Navigation System and SNAPE Framework Requirements (SysML Requirements Diagram)

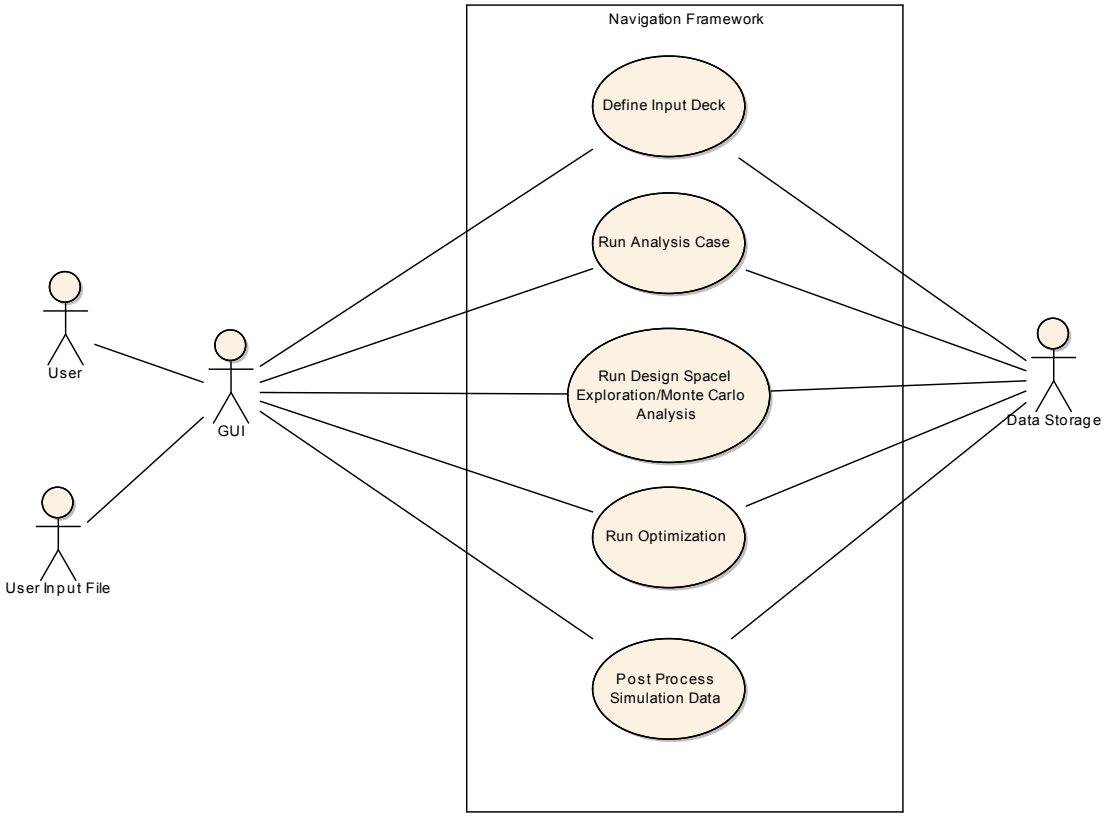


Figure 26: SNAPE Framework Use Cases (SysML Use Case Diagram)

With the simulation's requirements defined, the next step in the analysis methodology is the capture of the framework's use cases. These are similar to those defined above for the navigation system. Instead of operating with interfaces to external measurements and communication sources, these functions are modeled within the framework and the external interfaces are to the user via a graphical interface, and data storage hardware. The uses cases directly relate to enable the research questions laid out previously that define the capabilities of the framework and the ways in which it will be used. The main internal functions are: define and load an input deck, run a single analysis case, perform a design space exploration, optimize variable to maximize performance, and post process the simulation results for data analysis and visualization. Utilization of this model supports identification of the external users of the framework and its modes of operation. This aids in defining the required interfaces and ensures consideration of these needs when implementing

the analysis framework.

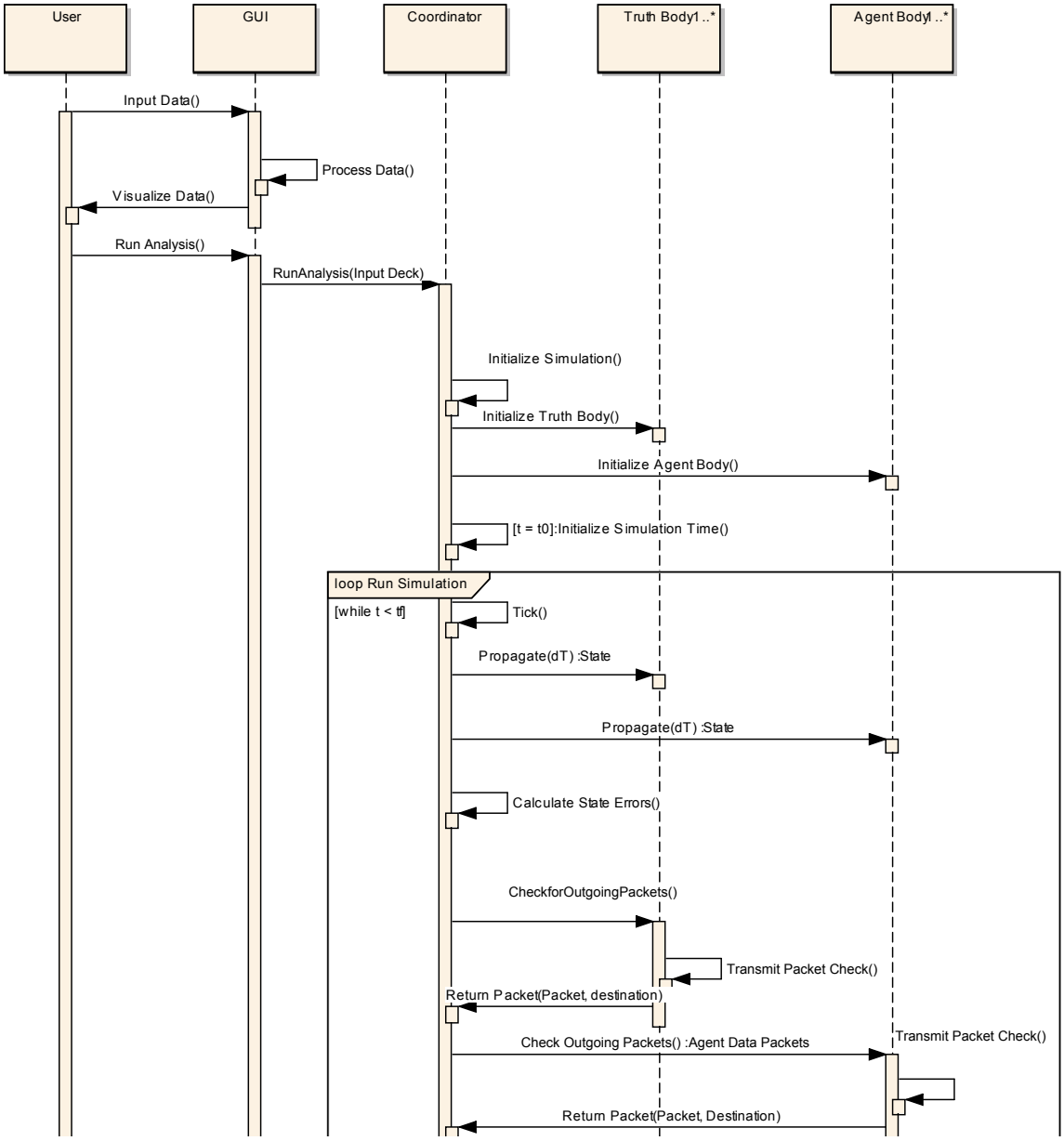


Figure 27: SNAPE Run Analysis Algorithm (SysML Sequence Diagram)

After defining the primary use cases, it is important to derive the sequence of functions and events that correspond to the analysis of the system. As described, this model captures the events that must be performed across an analysis. Part of this model is presented in Figures 27. to capture this core functionality. The ability to interact with external design analysis tools and perform design space exploration and optimization build upon this base

sequence by iterating over this sequence of functions. This model is based on that given in Figure 20, but goes into greater detail into the individual processing of each individual measurement and packet. The loop structure 'Run Simulation' captures the ability of the simulation to iterate over a period of time at a specified time tick. The other loops 'Process Packets' and 'Measurement Processing' demonstrate the need to iterate over a sequence of potential state updates occurring over the course of the simulation.

The individual swimlanes and the associated functional links also begin to describe the differentiation between truth agents and spacecraft agents. This distinction demonstrates a need for two similar objects which share many of the same properties, such as state, but are used in very different ways and serve different purposes. This also represents how the simulation framework must propagate the true state of each simulation agent as well as the agent's own estimated state. As before, these linkages between blocks form the basis of inter-object data and functional interfaces as well as serving as a high level outline of the simulation algorithms.

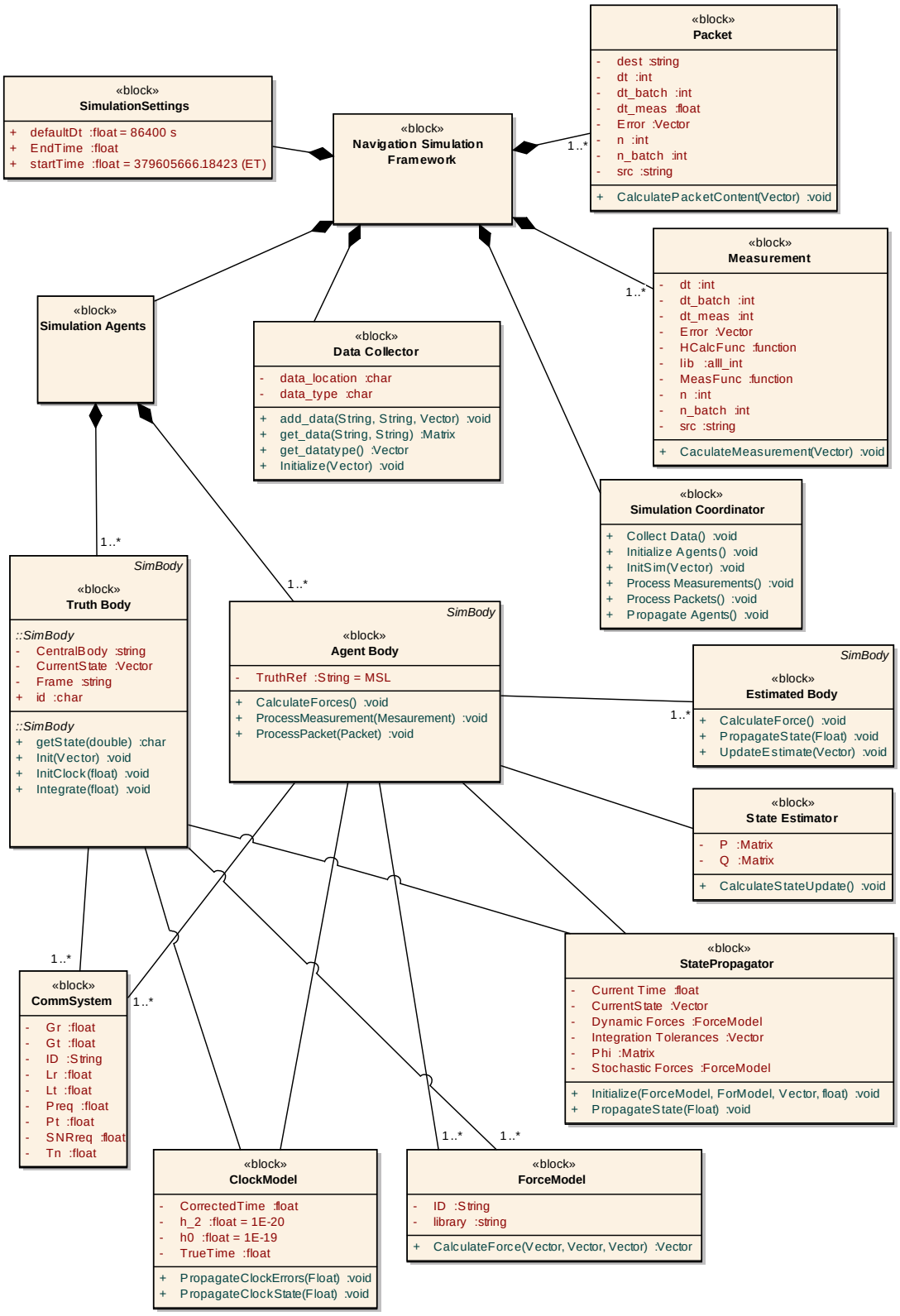


Figure 28: SNAPE Framework Design Model (SysML Block Definition Diagram)

With the individual functions defined and the primary objects identified, the framework system can be decomposed into its representative objects. This is done using a Block Definition Diagram. The model for the navigation framework is given in Figure 28. This model describes the composition of the simulation, and individual blocks forming the overall system. Additionally, each block object includes representation of its key attributes and operations, allowing for further decomposition and declaration of characteristics. This also allows for the assignment of the functional interfaces to specific model objects. This aids in the simulation implementation by providing a high level definition of the minimal set of capabilities required for implementation.

The objects within the framework are developed using ABM techniques, with distinction driven on capturing independent blocks or behaviors. The primary objects presented in the model are the simulation bodies, broken down into truth objects, agent objects, and estimated objects, individual packets and measurements, the simulation coordinator, and the collector. Additionally, the estimated agent body is composed of several other objects, such as its propagator, clock model, communication system, and state estimation algorithm. These analysis components serve as functional blocks within the agent body, allowing for implementation using a vast variety of algorithms. This decomposition also forms a basis for the data structures to be used in file input/output and object instantiation.

With the framework decomposed into its representative elements, the last step of the implementation is to add in the details for a specific analysis. For this example, the comparison of multiple measurement types is described in Figure 29. This represents an instance[4] of the framework to analyze a particular problem. The truth and spacecraft bodies have been defined as Earth, Mars, MRO, and MSL truth bodies (for simulation tracking) and the MSL Agent Body as the primary agent of interest. This object is further composed of a specific 8-state Extended Kalman Filter, and a Runge-Kutta 4(5) state propagator. Additionally, the specific simulation parameters are included that bound the analysis. These implementation diagrams can be implemented for each analysis case under study to provide

[4]This is not a standard SysML model, and does not meet the specifications. The purpose of this diagram is to provide an overview of the SNAPE implementation for the NNAV analysis scenario and to demonstrate how these types of models can be used to define input decks for test cases.

a basis for data input and data storage in a very visual manner (in comparison to machine readable text files). Additionally, this serves as a framework to collect analysis cases and their assumptions in one common platform within the Systems Engineering toolsets.

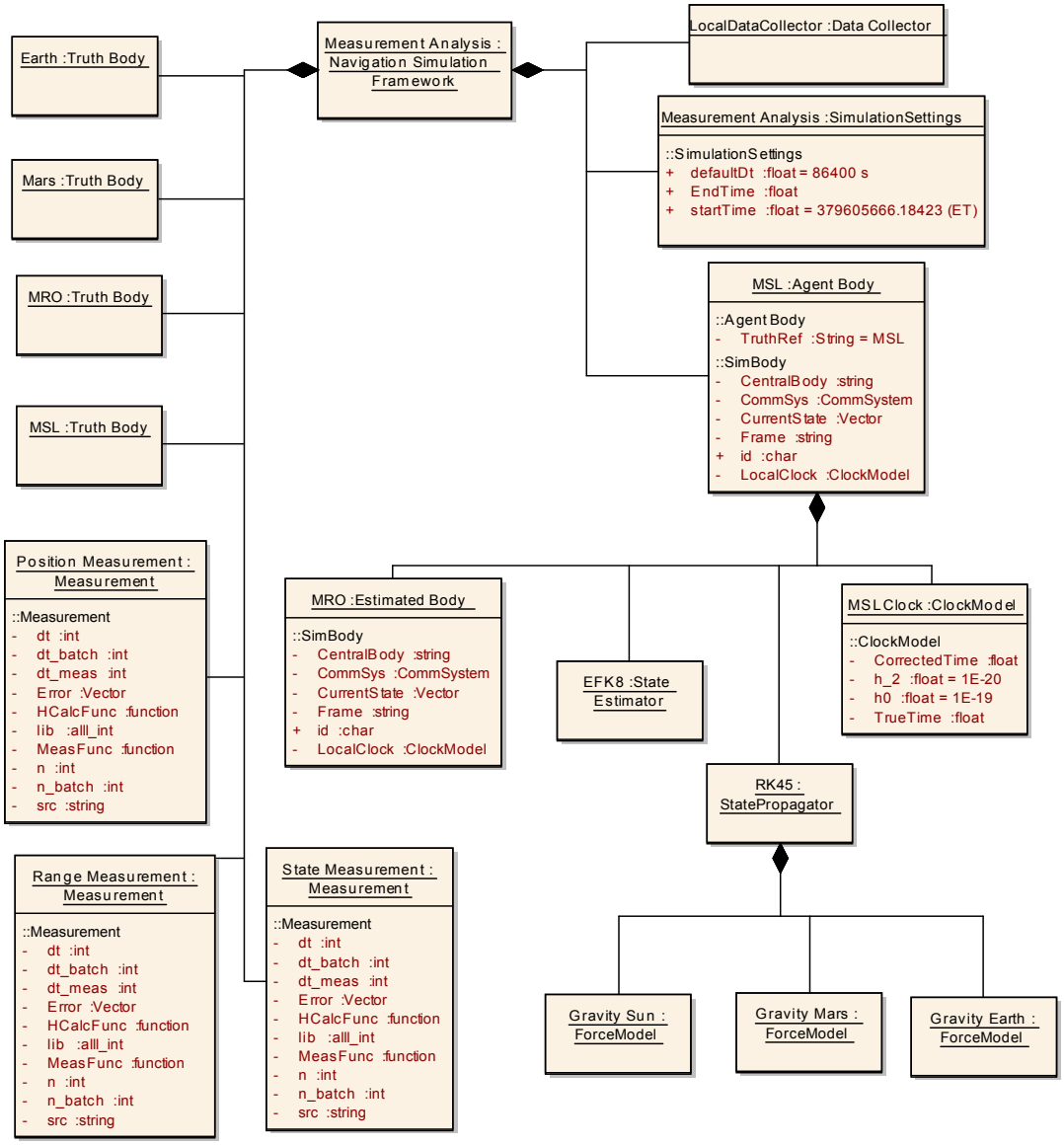


Figure 29: SNAPE Analysis Scenario (Instances shown on SysML Block Definition Diagram

The integration of these individual models serves to provide a complete view of the navigation system under study. From the top level navigation system requirements to the low level specific implemented state estimation algorithm, several layers of detail are

captured across the models. This serves to capture the designer's assumptions about the system, and the underlying logic of the navigation system. It also provides a framework to enable quick identification of common objects across analysis cases and can be used to build up a series of operational scenarios or use cases. Most importantly though, these detailed models of both functional interfaces and object decomposition feed directly into the simulation implementation to serve as a backend and definition of the tool's functional requirements.

5.4 SNAPE Implementation

The previous sections in this chapter have developed the SNAPE conceptual framework, capturing the structure, behaviors, and requirements of the navigation analysis process. One product is a collection of prototypes to capture conceptual software objects, attributes, functionality, and algorithms. A key aspect of this framework is that its development is independent of the implementation. The remainder of this chapter focuses on one implementation of the SNAPE framework into an executable simulation environment to perform analysis in support of verification and validation.

In order to allow for a modular and easily expandable development language for the framework software components, several options were compared and contrasted. The functionality of the tool posed several unique requirements on programming capability. In order to process the published trajectory data in the simulation as the truth reference, it is necessary to include access to the SPICE library of tools [2]. This is the standard package used for deep space mission planning and data analysis[5]. It includes several packages which support timing analysis, geometry analysis, science planning through its standard interface to spacecraft geometry model, trajectory input decks, and planetary ephemeris models such as DE421 [39]. The SPICE file format is also the standard for deep space trajectory data analysis and distribution. As such, it is necessary to have a software linkage to this software package. Currently the SPICE library is implemented in C/C++, JAVA, MATLAB, and IDL. Libraries additionally exist that wrap the software package in Python[6]. These

[5]http://naif.jpl.nasa.gov
[6]https://github.com/rca/PySPICE

languages therefore form the basis for the down-selection.

Important factors for selection are access to standard libraries for mathematics support, graphics output, data input/output, and modular user interface. Another important facet is the ease of functional development and debugging. Ideally this would also be implemented in a widely supported open language to aid in future expandability and development. Also important is the development and use of standard functional libraries to aid in tool capability. A stable interface is needed to ensure maximum software lifetime.

From these factors, the Python programming language was chosen for implementation. This selection was made for several reasons. Version 2.7 is a mature release of the software and is widely supported, with a wide code base from other developers. There also exists a wide range of utilities for this version of the language. The primary libraries in use are the NumPy [84] and SciPy [66] numerical analysis packages, which provide links to standard software toolsets. Additionally the recent release of a SPICE Python interface allows for integration with a wide range of ephemeris data and functionality. This language also provides libraries for quickly developing a user interface using WXPython [94] with MatPlotLib for generating data visualizations [61] and integration with a range of other developed libraries through C code integration and linking.

The other languages also offer similar support, but have not been chosen due to several factors. MATLAB, though powerful with a wide range of functionality, is a closed interface and the implementation can change year to year. Additionally, the programs can tend to be memory intensive, which caused problems during initial implementation testing. C/C++ was not chosen due to its platform specific nature, and requirement of local compilers to develop executable code. User interface development is also limited and has similar limitations. These characteristics led the work to be developed in Python, due to its open nature, wide range of analytical packages, fast development, and high level of execution performance, as well as prior experience.

The software implementation of SNAPE is broken down into four main packages: the graphical user interface, the linking libraries to connect external functions with the framework, the simulation backend, and the data collection object. The separation of these

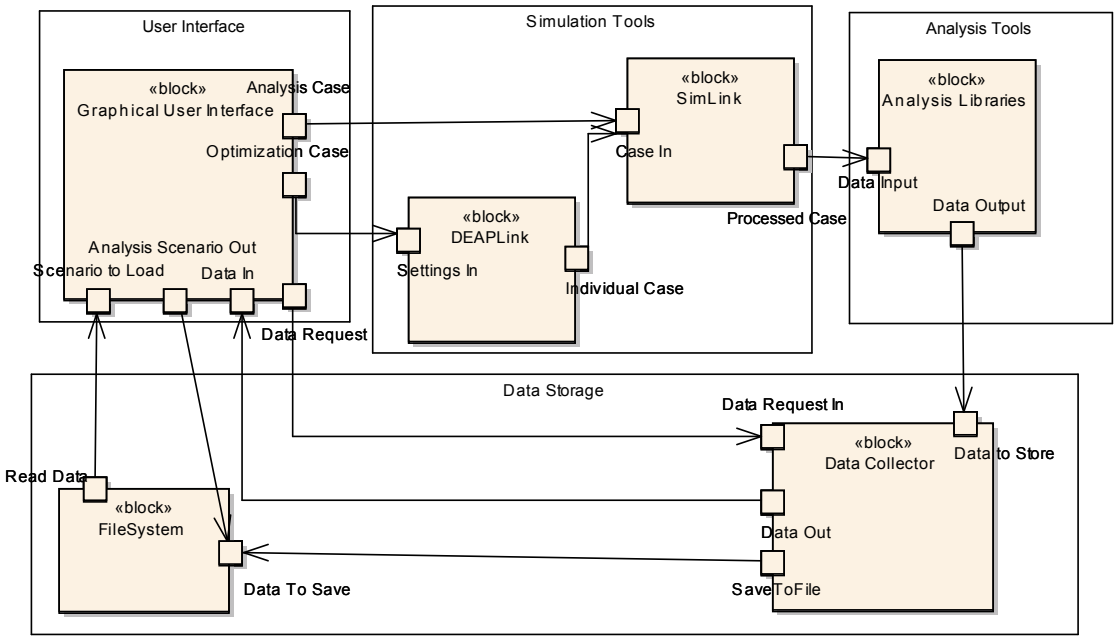


Figure 30: SNAPE Software Design - Flows between Subsystems (SysML Internal Block Diagram)

elements and their interfaces are given in Figure 30. These sub-modules are developed in response to both the requirements definition and the identified functional requirements of the simulation. Visualization of these relationships is given in Figure 31. As can be seen in the diagram, each module ties directly into the higher level requirements and implements a particular set of functions. This shows the continued flow of needs and prototypes from the initial analysis. Each of these modules and their implementation will be discussed in the following sections.

5.4.1 Overview of Simulator Execution

With the system defined and the blocks identified, the software was implemented using the Python programming language using object-oriented techniques to develop a modular robust simulation framework. The next series of subsections focuses on the analytical backend of the computational environment. Other functionality of the simulation is enabled by the proper design and implementation of these software modules. These algorithms and the modeling approach are captured to both serve as documentation and to provide a rational

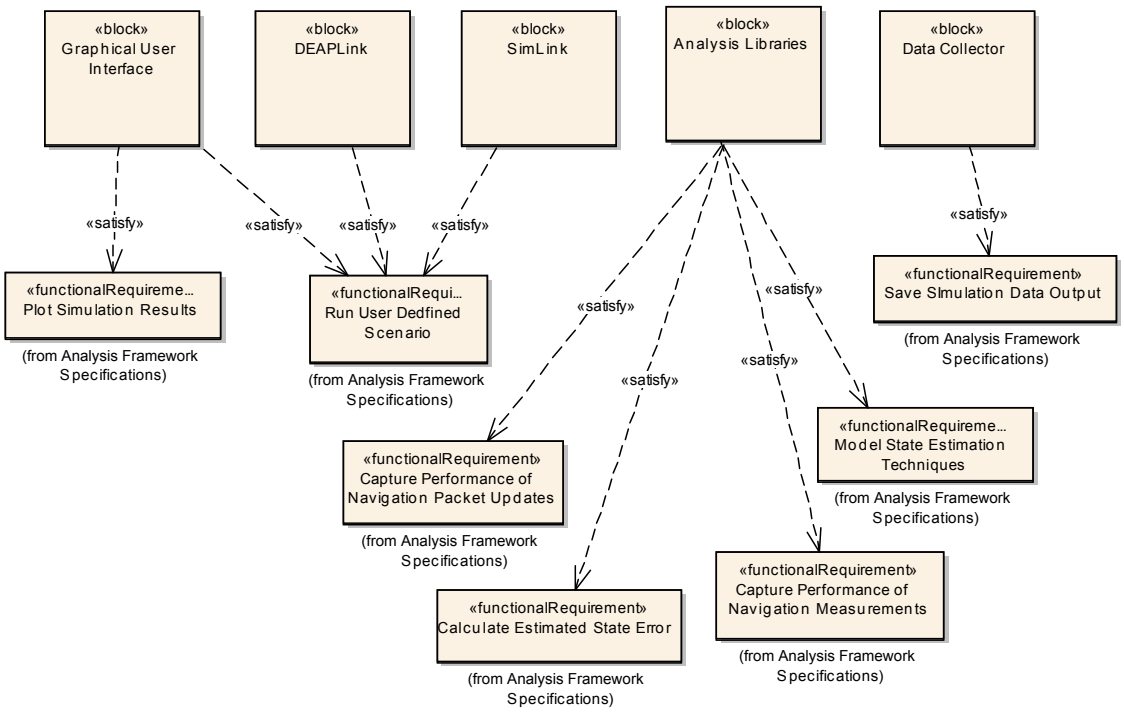


Figure 31: SNAPE Software Requirements Satisfaction (SysML Requirements Diagram)

basis for the method of analysis.

5.4.2 Simulation Coordinator

In order to integrate all of the individual software blocks and perform general simulation housekeeping functions, a simulation coordinator is implemented as a Python class. The functional requirements of this system are given in Figure 32. The coordinator integrates the individual analysis modules and runs the analysis. Its sequence diagram is based on that developed for the framework, as given in Figure 27. The inputs are defined as an input deck, consisting of the defined truth bodies, estimated bodies, measurements, packets, and simulation parameters, and a link to any data handlers to store any required performance or status data. The coordinator is built upon ABM simulation approaches, allowing for definition and operation of independent simulation agents.

The initialization function of the class builds up the simulation model, initializing the individual agents in the simulation, and preparing the analysis variables. The run_sim function performs the analysis of a given defined scenario and operates in the sequence laid

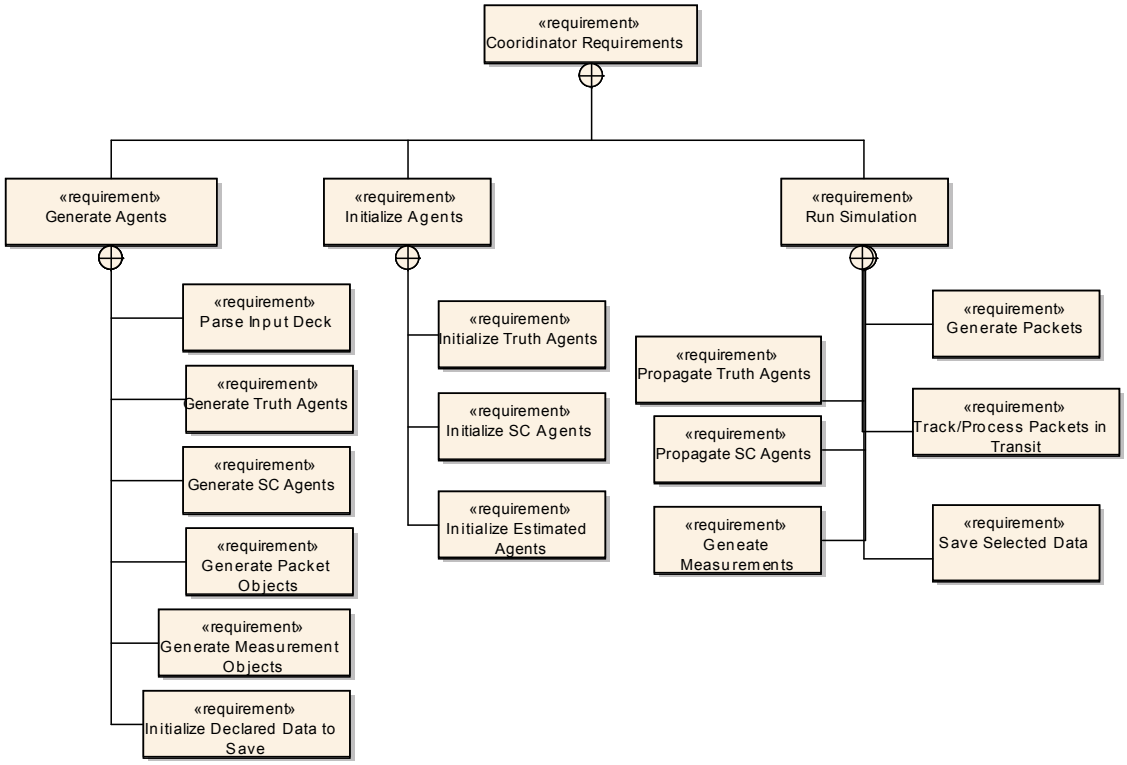


Figure 32: Simulation Coordinator Functional Requirements (SysML Requirements Diagram)

out in Figure 27. In order to collect data, the coordinator parses the data output definitions and sets flags in each class to identify individual variables that are to be sent to data storage. Following this, the coordinator parses each truth and agent body, initializing the state of each, depending on its definition (and reference to other objects).

Each agent body also has the capability to predict and estimate the location of other spacecraft. This is performed through the use of an Estimated Agent object. This object is contained within the agent body, and is initialized to match the initial estimate a spacecraft would have of the other bodies, at its launch. It also includes functionality to allow for updates of the estimated state and definitions of force models to use in propagating the object's state. The coordinator performs this initialization for each estimated agent within each spacecraft agent in order to link the known truth states with the onboard values.

Once all of the objects are prepared, the coordinator proceeds to iterate through time to capture the performance of the navigation system. Using a default input time step, it first updates the state of each truth body, bringing the truth dynamics up to time. Next, it proceeds to command each spacecraft agent to propagate to the current simulation time, as well as commanding each of its internal estimated agents to propagate forward as well. Following this, it iterates through the measurement objects and checks internal timestamps to determine if a time for one to be dispatched has been reached. If the measurement is due to be released, the truth measurement is calculated and passed to agent body, which uses its onboard state to model the measurement and then process it through its onboard filter. If the measurement is not due yet, the remaining time is estimated. This is compared to the default timestep to enable direct iteration to the event or the next propagation, whichever is first.

Once all of the measurements have been tested for generation, the coordinator performs a link analysis between each defined communication for the truth bodies following the algorithms described in the previous chapter. This is done as a diagnostic measure to track reception power levels and link capability. The software then proceeds to iterate through the list of predefined packets and check if any have reached time for transmission. If so, the link is analyzed to determine if the packet can be received by the other spacecraft using the

truth states of the objects. If the data can be received, the packet data is built, and the time of arrival is calculated. At this point, the packet is placed into a queue awaiting the correct simulation time to be released to the receiving agent for processing. The coordinator then iterates over the active packets, and checks for any ready to be received. If a reception time has been reached, the individual packet is removed from the queue and transferred to the spacecraft agent for processing. Similar to the measurement checking, the coordinator also checks for the next reception event and compares this time interval to the default time step. If the calculated timestep is smaller, it is used in the next propagation event of the simulation.

The simulation coordinator also includes a large amount of data housekeeping capability. By defining a debugging level within the simulation parameters, varying amounts of detail can be output to the user describing the current state of the simulation. Additionally, at multiple points in the analysis, the simulator performs a check for flagged output variables at the system level. These include state propagation errors, state estimation errors, and other error terms. This allows for capture and post-processing of these values to determine the performance of the system and feeds into data visualization.

5.4.3 Truth Body

The truth body class is an implementation of a navigation agent within the simulation framework. These objects are specifically used to track the true position and state of an object over the course of the simulation, serving as a truth reference. The initialization parameters include declarations of a truth reference, any available communication systems, and clock parameters. This allows for tracking of the truth onboard clock's state as well as its state. The class definition allows for the capability of integrating any truth source, though its integration function. Currently, this is implemented as a SPICE state lockup for the given time, frame, and central body, but can be expanded into future iterations to include loading ephemeris from file, or propagation given an initial state.

5.4.4 Estimated Body

The estimated body class implements a limited set of these features. It is intended to be used with an agent body to capture the estimated state of an external body. This allows for analysis of the spacecraft's propagator and the effects of errors in the state of other sources, which is a key driver for processing a navigation packet (determining the transmission location). This object includes a fully implemented propagator, allowing for definition of other bodies to include in gravitational analysis and other forces. The types of bodies are also defined and used to determine their estimated location and parameters, such as gravity models. Currently these models are also implemented as SPICE lookups, using the capture planetary properties data and ephemeris in the DE421 ephemeris library. This is used to represent the spacecraft launching with an up-to-date model for planetary bodies. The propagation function calculates the total acceleration on the vehicle as a function of the current time and passes this, along with its estimated state to utilize SciPY's integration library. The specific integration scheme and parameters are defined via the input deck to determine the propagation method and acceptable error levels. This estimated state is used to inform the parent agent of its estimated position.

5.4.5 Agent Body

The agent body expands the functionality of the estimated body, including the capability to define force models to include in its propagation as well as the specific integrator to use. The software is currently set up to pass in an integration class of the form implemented in the SciPY Ordinary Differential Equation integration library. As such, it is compatible with all of those potential options as well as external libraries that match the predefined class structure.

In addition to providing state propagation functionality, this agent also includes the capability to process measurements and packets. By the identification of modeling functions for the measurement type, the agent is able to estimate its expected value of the observation. Additionally, it can calculate the table of partial derivatives linking the observed state to the state variables. These are typically used in state estimation procedures. The modeled

values as well as the measured are then passed to the onboard state estimation object, which calculates a state update. The agent's estimated state is thus corrected.

The object also includes similar functionality to process and interpret any received packets. In order to process the received data into a format usable by the state estimator, the agent includes processing to analyze the contents of the packet, and process it accordingly. This analysis process includes both estimating the observed value calculated from the packet contents, and constructing an observation using this information. Typically, this involves propagating an estimated agent's state backwards in time to the known time of transmission. The partial derivatives must also be calculated for the navigation filter iteration. Through this processing, the state of any estimated bodies comes into play and is used in the observation which the filter uses to calculate a state update. The individual components of the agent body are discussed below, going into detail on the propagator, measurement processing, packet processing, and state estimation implementations.

5.4.6 Propagator

The agent deck allows for specification of the specific propagator and force models to use in the onboard agent propagation. This is applicable to both agent and estimated bodies. The standard interface to the integration libraries is by means of a gradient function which calculates the derivative of each state. These can be calculated by nonlinear equations. The functions to calculate the forces are specified by the inputs variables containing other gravitational bodies and other forces. These are specified in terms of an id and a function linking to the force model. For state propagation a link is passed in agent initialization to the SciPy.Integrate library's dopri5 function [56]. Additional integration terms passed into the object include absolute tolerance, relative tolerance, and max number of steps to use in the integration. The force models also link to their respective functional identifiers. For the implemented models, the affect of gravity of external bodies is calculated based on the spacecraft's estimated, onboard ephemeris libraries, and SPICE planetary parameters. In addition to the dynamic states, the model also propagates the state transition matrix, which is used in the state estimator. The dynamics of these systems are defined as presented in

the previous chapter. With this functional interface, initial state definition, and final time the propagator is able to integrate the spacecraft's estimated state forward or backward in time.

5.4.7 Measurement Processing

The processing of navigation observations is also handled by the Agent Body class. The true measurement is calculated by the coordinator in reference to the true dynamic state. Most of this data captures observations of external bodies. As such, the first step in the processing is to calculate the observed body's state at the time of observation. This step is currently performed by the onboard ephemeris libraries. Once calculated, the reference measurement function is called using the best estimated state parameters. This input is a direct reference to an external function that follows the defined input prototype (including the state estimate, central body location, time, and predefined errors), allowing for a modular interface to a variety of output functions. This has been implemented and testing with independent functions for time, range and position (both absolute and relative), and state measurements. Additionally the gradient matrix is calculated for the measurement type using the onboard state estimates. This information is transferred to the state estimation algorithm for update calculation. Upon reception of a state update, this is then applied to the spacecraft's estimated position along with a reset of the state estimation matrix (due to the observation and assumption of the new state being corrected to true).

5.4.8 Packet Processing

The navigation packets are generated in the simulation coordinator's run loop. The data in these packets are embedded with the estimated state at the time of transmission. For this analysis, it is assumed that the sources of packets, MRO and DSN, have highly accurate, frequently updated ephemeris and timing information, providing essentially true observations of state. Upon reception of a packet, the agent first checks to see if it is tuned in to the corresponding transmitter. This behavior is captured in a state machine diagram given in Figure 33. This attempts to capture the communication search pattern of the spacecraft and begins to define the autonomous behavior. For example, the spacecraft does not assume

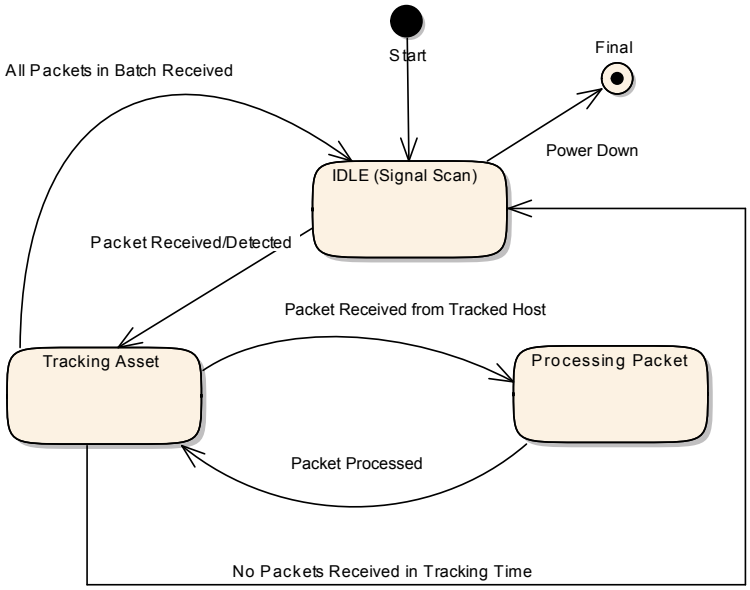


Figure 33: Packet Processing Modes (SysML State Machine Diagram)

any a priori scheduled passes, but operates in an idle listening mode. After initial reception, the agent focuses on the transmitting host until either the batch is complete or no signal has been received in a predefined time period, triggering a timeout event. This also represents the ability of the spacecraft to only communicate with one spacecraft at once. Due to its modular nature, this behavior can be modified and updated for a specialized agent model which could include more advanced communication, scheduling, and analysis capabilities.

Once an agent is tuned in to a specific navigation host, it then proceeds to analyze the received data and form a navigation measurement. There are two primary classes of measurement packets currently captured in the analysis: time of transmission and time of transmission with known transmitting agent state. Additional data packets can be analyzed by including additional measurement and processing modules. The architecture itself is robust to a variety of potential transmitted state information between agents. The agent calculates an observation, typically a range and timing observation from the packet reception values. If the transmitted state is not included, the onboard estimate is used. With the known time of transmission, and current estimated state, the range observation and its gradient are calculated. This information is then processed by the state estimator and the state corrected.

Additionally, the agent packet processing is capable of forming additional indirect state measurements. Specifically, the spacecraft estimates the range-rate of the distance between it and the transmitting host. This allows for a velocity observation (similar to a Doppler measurement of radial velocity) but is computed using the newly corrected state, and allows for further refinement and improvement of the velocity by its indirect observation. The implemented algorithms mirror those developed in the previous chapter.

5.4.9 State Estimation

The state estimator is implemented as a class inheriting the prototype NavFilter class definition. This object contains the estimation algorithms used in the state estimation and correction processes. Six and eight state (includes clock bias and drift terms) Extended Kalman Filters are currently implemented in the framework. This class captures all of the attributes and operations which the filter will perform during the simulation. In addition to calculating a state update, this object is also responsible for tracking and propagating the states covariance matrix, which describes the current noise in the state estimate. This matrix is included in the filter due to its dependence on the state estimation algorithm in use.

The primary interface to this object is through the state update function. The inputs to this include the modeled measurement, the observation, and the estimated gradient. The filter proceeds to propagate the covariance forward in time using the state transition matrix (which is continually tracked from the time of the last measurement update). With this updated value, the filter is able to calculate the Kalman gain and generate a state update, which is returned to the requesting agent. With this modular and agent-independent implementation, it is possible to test and analyze a wide range of analysis cases by following the class prototypes.

5.4.10 Data Warehouse

Storage and retention of data is the goal of the data warehouse. Prior to running an analysis case, the defined data is declared in the input deck. Individual data variables are defined by the object id, object type, and specific data array. As defined events occur in

the simulation, such as propagation or processing a state update, the coordinator checks if that data is flagged to be stored. If so, the data identifiers and the relative data are forwarded to the data warehouse. The modular interface allows for the warehouse to be implemented across a wide range of storage medium. For the initial implementation, this module simply stores the data locally within the simulation to a variable internal to the Sim_DataCenter class. This allows for the fastest storage and retrieval of data within the analysis framework.

For long term storage and later analysis, these data files are outputted using the Python Pickle module[7]. This built-in Python module allows the writing of internal data objects and classes directly to the file system in a way that allows for later direct operation of the individual variables once reloaded into Python. These pickled data files form the basis of the data storage warehouse. Additional interfaces could be implemented to save the data directly to file or to an internal database in real-time. These could be developed by following the data center functional prototype.

In addition to the storage of data, this object also includes functions for data export to a user. This enables any Python script that can load a pickled file to have direct access to the data warehouse and all of its functional interfaces. For the current implementation, each data element is appended to a master list. To return a specific piece of data, an id and object type are specified. The data warehouse then parses through its data elements and returns any data that meet the requested inputs. A final function of the module is to return a data summary, providing all ids accounted for and any data types available. This simplifies later analysis into an unknown database, by providing an overview of its data contents. External functions of the user interface and other analysis tools interact with these data objects to retrieve information for post-processing or visualization.

5.5 Simulation Integration and Operation of SNAPE

The previous sections described the internal behaviors of the analysis libraries of the simulation framework. The individual modules presented define the minimum capability needed

[7] http://docs.python.org/2/library/pickle.html

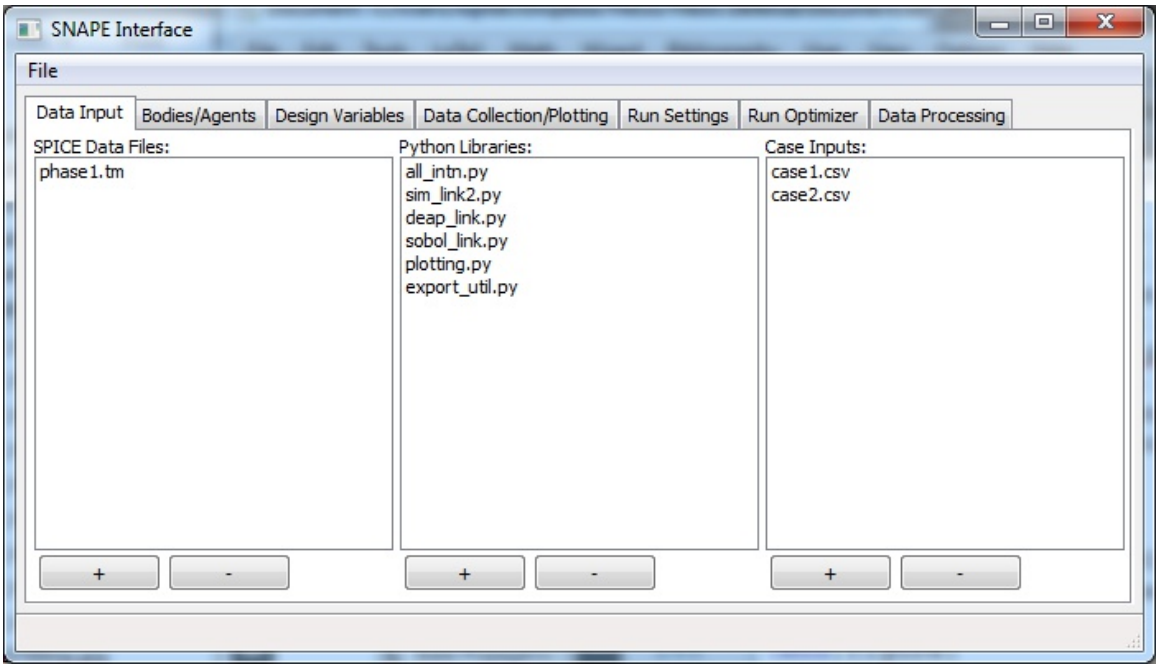


Figure 34: Data Input Interface

to analyze a specific navigation scenario. There are several key requirements of SNAPE that exist independently of the analysis backend, supporting users of these modules. These include the graphical user interface, the data definition modules, the parser between these and the simulation environment, the integration of optimization libraries, and the visualization interface. All of these aspects are part of the SNAPE framework's implementation.

5.5.1 Modeling Interface and Data Input/Output Definition

In order to provide the user a graphical method of interacting and defining an analysis case, a graphical interface was developed. This user interface allows for an interface with the simulation backend to define, load, save, or modify an input deck, run analysis cases, and visually explore the results. The initial data input screen gives a view of the overall user interface layout. This initial view is presented in Figure 34. This window allows for the definition and tracking of the software libraries and data files in use within the simulation. Future iterations of the interface plan for these file input windows to integrate with the analytical backend and probably library access for agent definition and linkage to predefined variable inputs.

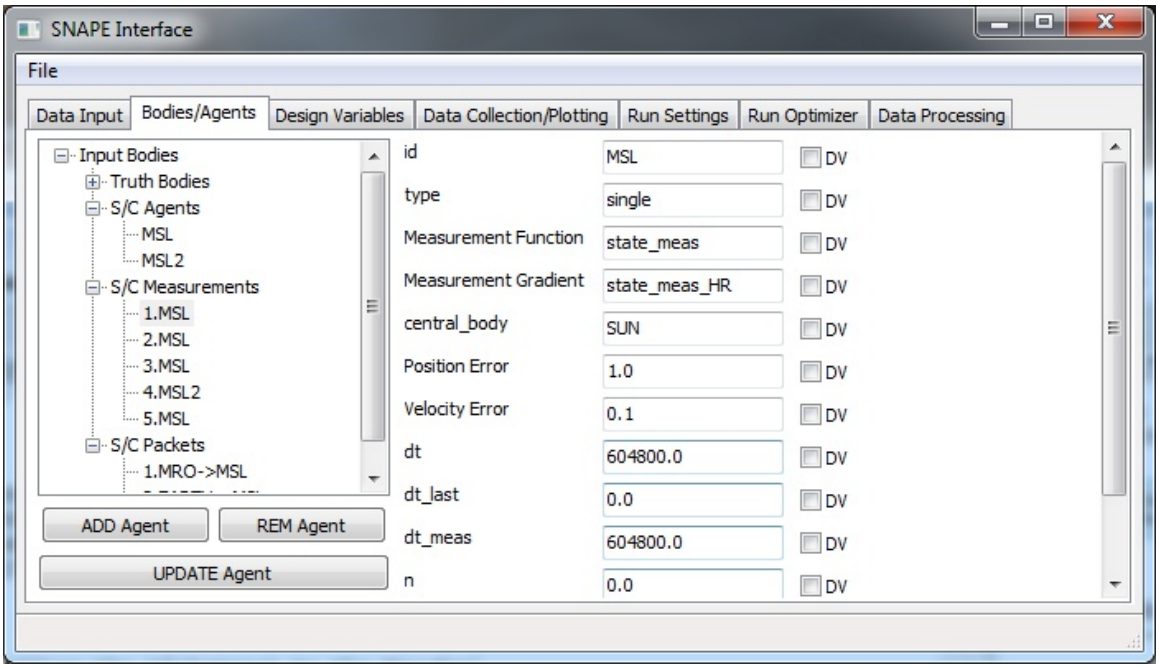


Figure 35: Agent Definition Interface

The next tab allows for the user to define the agents that are included as part of the simulation. This view can be seen in Figure 35. This interface utilizes a tree structure to organize the individually defined simulation objects. These are grouped by truth bodies, spacecraft bodies, measurements, and packets. Through this interface the user has access to all simulation parameters, and is able to define a wide range of scenarios. By double-clicking on a specific object, the user is able to pull up the properties of an individual object. Through this interface, the user can also identify parameter of interest to act as a design variable. This allows for inherent collection of the variables values during runtime and enables the variable in Monte Carlo and optimization analyses. After input, these can be updated, and saved back into the input deck for analysis.

Another aspect that needs to be defined prior to running the simulation is to capture the data variables of interest. Each type of object has a default potential set of measurements that can be collected (for example an objects state, position error, velocity error, etc). These are specialized to each specific kind of agent and are programmed into the simulation framework. In order to tailor the data collection needs to only those required by the user, each data source of interest must be added to this data collection list. This list is passed

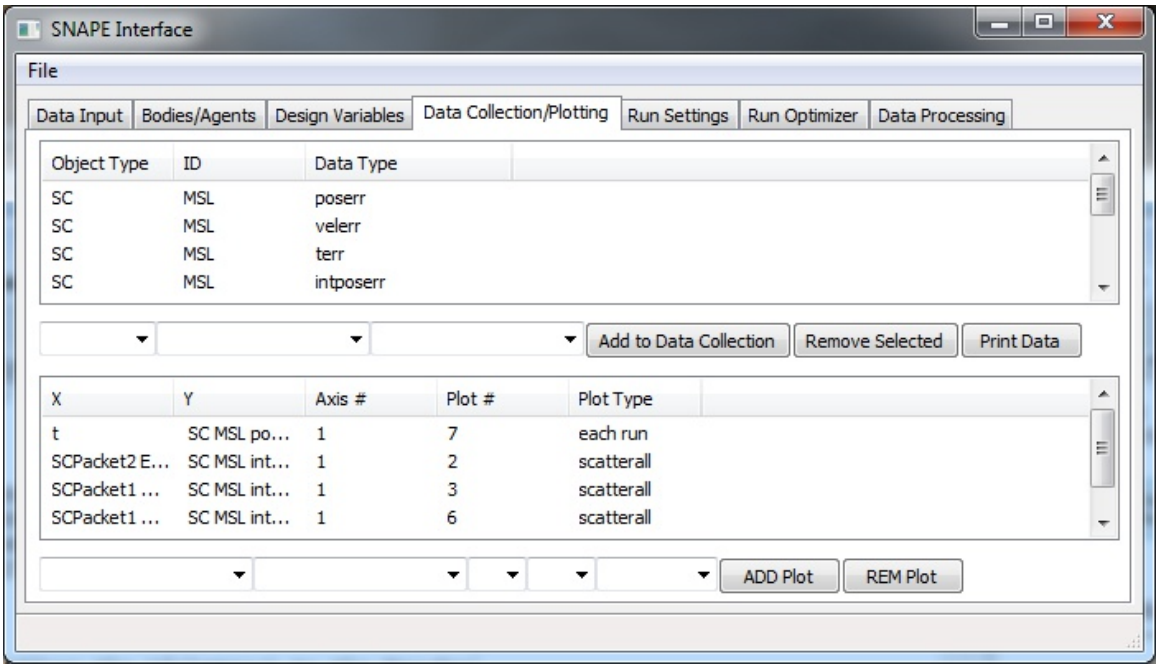


Figure 36: Data Collection Interface

to the analysis backend which then saves these data sets to the data collector. Through this interface, the user can also define a series of plots to be generated post analysis. The options allow for plotting of each analysis case for dynamic measurements as well as scatter plots of defined variables. This data input interface is shown in Figure 36.

The user can also define input ranges for any variables identified as design variables on the next tab. This interface can be seen in Figure 37. The view captures all of the characteristics of these variables, including the associated object and specific variable. Most importantly, this allows the user to define a range of options for each design variable. Uniform, normal, integer, and logarithmic integer distributions are currently implemented. The input parameters change definition for each analysis case. For the normal distribution, the inputs are the mean and standard deviation of the variable. For the other options, the inputs are the maximum and minimum values for the variable. The log-integer case utilizes a sample from a distribution of integers. The selected value is then interpreted as a power of ten for use in the analysis engine. This allows for uniform studies over a wide range of orders of magnitude. This is particularly useful when looking at navigation performance across orders of magnitude error values as well as allows efficient input of variables such

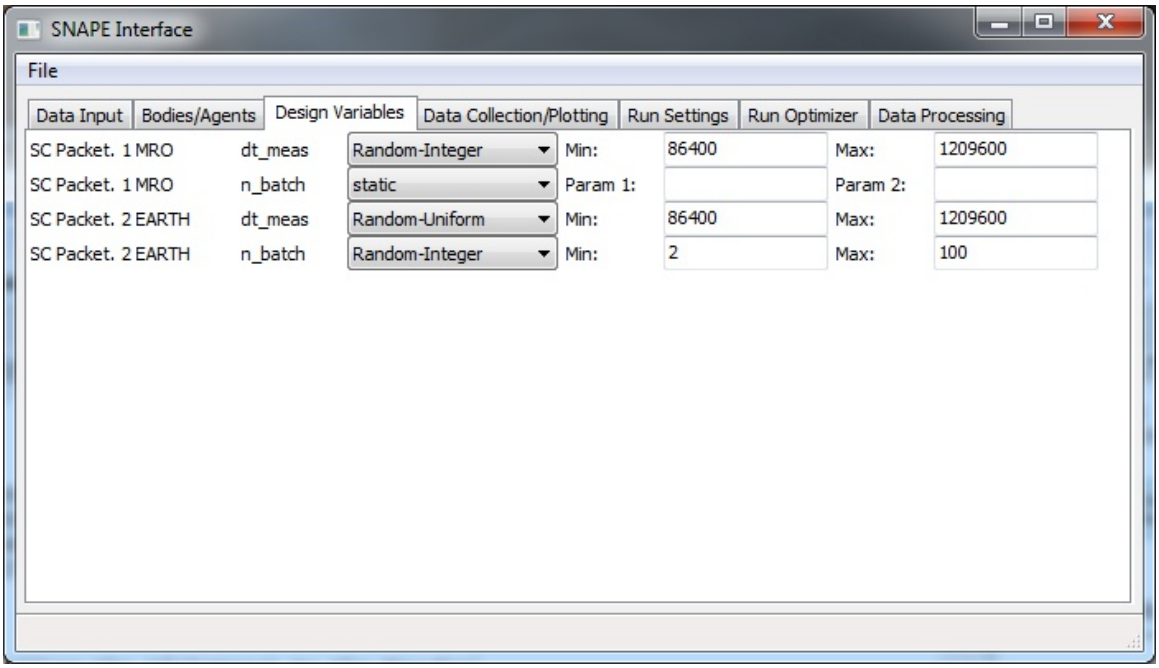


Figure 37: Design Variable Interface

as clock noise which are defined as powers of ten. Also, this enables initial studies into numerical optimization of filter parameters, such as the process noise, in which the order of magnitude is key.

5.5.2 Simulation External Interface Definition

Before proceeding onto how the user defines and performs an analysis or a design trade study, the link between the front end and the analytical backend must be described. As shown above, the SimLink library provides the interface between these two tools. This is due to the changes in data format between them. One key difference is due to the way the analytical framework loads and interprets external libraries. From the user interface, the designer is able to specify external functions through their library using text strings, identifying both the source file and the specific function handle. The simulation initialization functions are designed to take in a link to the actual function. As such, a link must exist between the two to make this transition. As such, the SimLink library instantiates the objects and functions that are passed to the simulation.

Another function captured by this interface is the generation of the random numbers

for the design variables of interest. By parsing the options set to a variable, this link makes the actual random draw, providing an independent selection external to the backend analysis. This functional separation also allows for a more user friendly input definition in the front end, by allowing everything to be defined by string inputs. This also improves the readability of the user input decks written to a file. The SimLink library is therefore shown to act as a wrapper around the simulation framework allowing for a robust input deck format to be used and providing an interface to a wide array of input cases.

5.5.3 Design Space Exploration Capability

With the link between the graphical interface and the simulation backend defined, it is now possible to resume the discussion of the user functions. The simplest form of study is to run a single simulation case for an input deck. Inputs such as the simulation start time, default timestep, ending time, and output file are some of the key options that the user can set. Additionally, the user can select the debug level to be used for printing status data to the console. This interface is given in Figure 38, which gives an overview of this interface.

In order to perform a Monte Carlo Analysis of the design space with the user-defined input variables, the number of cases can be specified. In addition, the simulation end time of each individual case is provided. This is due to the highly stochastic nature of the analysis of the problem, and the same scenario with different specific initial error states can produce a range of results. This is due to the scalar definition of input initial position and velocity errors. In the initialization functions, these are used as standard deviations and applied to the initial true state. With the previously described tabs, the user can very quickly define an input scenario and perform a statistical analysis of the trade space.

5.5.4 Sensitivity Analysis Approach

Another important study used to capture the properties of the design space is to perform a sensitivity analysis on the defined design variables to the selected output parameters. There are multiple methods of analysis that can be used. One method is to calculate system sensitivities through the decomposition of the system variance [100] [101] [102]. Analysis methods exists for utilizing a directed Monte Carlo simulation to estimate the first order

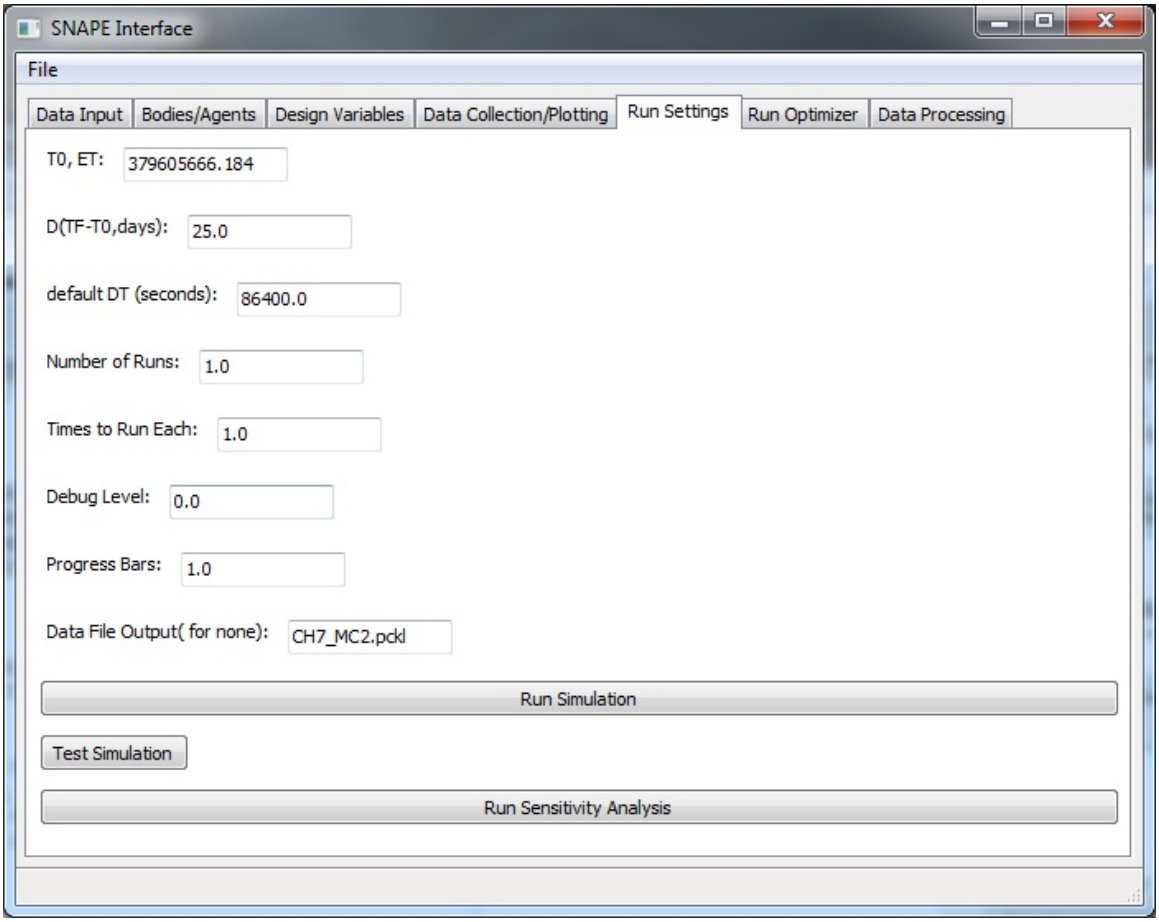


Figure 38: Simulation Interface

effects (S_i) and total variable effects (ST_i). The algorithm implemented is that presented by Sobol [110]. For this analysis, a specific sequence of design points is selected that allow for a uniform sampling of the space that maximizes the amount of design information available for a given number of cases by equally partitioning the space.

The Python library Sobol_Lib written by Corrado Chisari[8] was used as the sequence generator for this analysis. These defined cases were run through the analysis backend and processed using the Sobol and Saltelli identified algorithms to calculate the total effect. For this analysis, the number of runs is specified by this front end. It is important to note that the actual number of runs performed by the algorithm is N(2+d), where N is the specified number of runs and d is the number of input variables. The variance estimators [100] are used to generated the main effects and total effects of each design variable for each specified output variable. These values are then written to a file for further analysis.

5.5.5 Optimization Library Integration

The next level of analysis that can be performed with the simulation framework is to perform an optimization on the defined design variables. Due to the highly stochastic nature of the problem and state estimation, a genetic optimization algorithm was chosen for integration to the framework. This is additionally useful due to the combination of discrete and continuous variables in the design space (such as the number of packets in a batch for example). Another benefit of using Python is the large selection of compatible libraries available. Several exist for the implementation of the optimization libraries from nonlinear optimizers (such as NLOPT[9] and SciPY Optimization libraries) to generational algorithms, such as the Distributed Evolutionary Algorithms in Python library (DEAP) [43][10] and inspyred[11]. For this implementation the DEAP is utilized. This library was chosen due to its wide support and implementation of functions used in genetic algorithm optimization routines. It provides developed building blocks for each optimizer function and allows the user to design a specialized algorithm. The optimizer supports a range of

[8] http://people.sc.fsu.edu/~jburkardt/py_src/sobol/sobol.html
[9] http://abinitio.mit.edu/wiki/index.php/NLopt
[10] https://code.google.com/p/deap/
[11] http://inspyred.github.com/

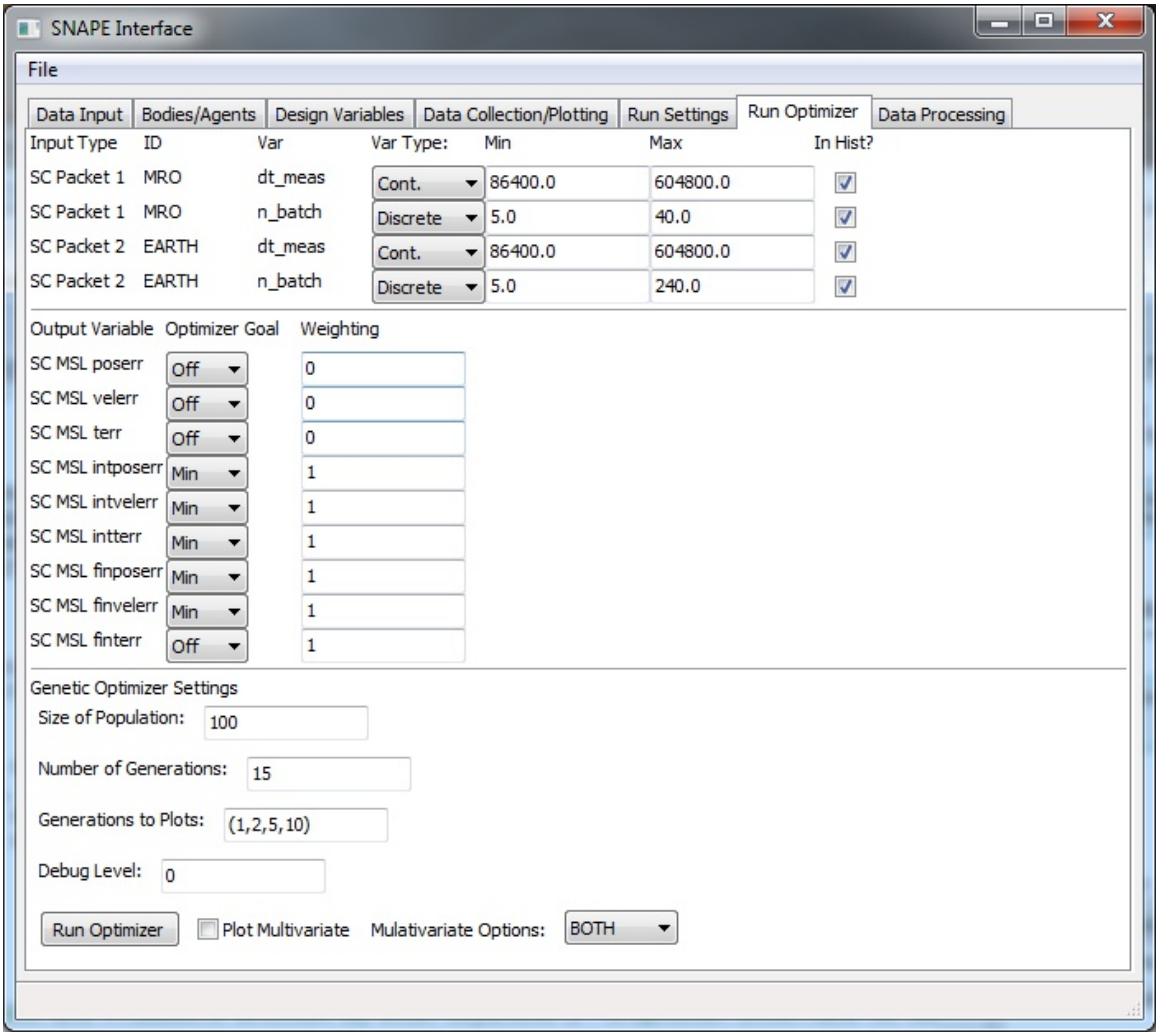


Figure 39: Optimizer Interface

optimization functions, including multi-objective genetic algorithm performance functions. This broad range of optimizer capabilities provides a robust basis for a large amount of design cases.

The interface, shown in Figure 39, allows for description of the input variable, specifying their range and type of variable (discrete or continuous). There is also the option to include each design variable in a series of histograms that capture the design space at a particular generation. The interface additionally allows for the specification of output parameters to use in the integrated evaluation function. Each variable can either be ignored, maximized, or minimized. The algorithm is set up by default to maximize all variables. In order to provide for a minimization, the weight is changed to negative. The user can also define

individual weightings for each design variable. Lastly, there are inputs for the population size, number of generations, and generations to include output for. By default, the optimizer stores to a file the user-identified parameter for each population member of each generation. This allows for further analysis of the optimizer's performance and tracking of the system performance.

An additional software link exists between the user interface and the SimLink library described above. This is referred to as the DEAPLink. This piece of software integrates the chosen genetic algorithm functions and generates the populations to be processed through the analytical framework. This tool manages the analysis process, tracking each case, saving the data to files, and performing the member selection, crossover, and mutation functions using the provided DEAP functionality. The current framework provides a straightforward optimization capability. With the integration of the DEAP library, there is a vast potential for additional optimizer development and studies of their performance for the navigation system. The current DEAP implementation can also act as a template for future optimizer developments.

5.5.6 Data Visualization

After completing the analysis, it is important to be able to visualize the results of the simulation. This is enabled through the use of the data visualization tab as shown in Figure 40. Upon selection of a data collector output file, the interface presents the user with a selection of the available data that can be displayed. The five drop down boxes across the top allow for selection of the primary id, secondary id, object type, data identifier, and specific variable. These selections are auto-generated as part of loading the data collector objects from file. An example of a primary and secondary id are generation and member numbers. The user can then select the data to be plotted on the given x-axis, y-axis, z-axis, or added to a list of variables to generate a multivariate plot. Additionally, the tool has the capability to read in the selected variables values and export these in csv format to be integrated into a wide range of data analysis packages.

The plotting interface can also serve as a diagnostic of an optimizer's performance. For

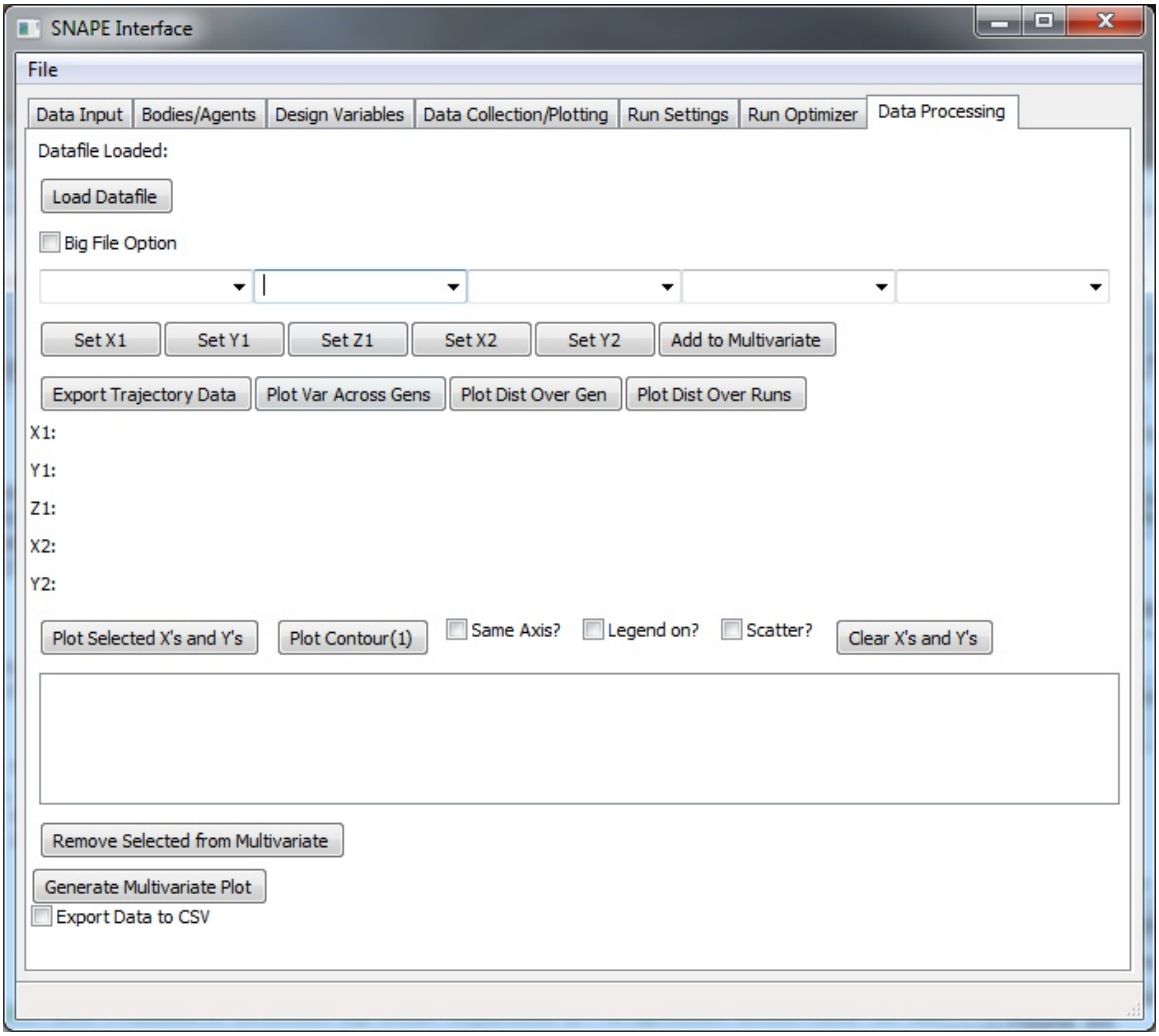


Figure 40: Data Processing Interface

an optimizer, each iteration is saved to a file. This data can then be processed to track a variable across the individual iterations. This is useful for tracking the performance of the genetic algorithm, such that the designer can easily identify trends in the optimized variables. This is observed by plotting the mean of a variable (either input or output) across a population across generations. Additionally, vertical error bars can be used to capture the standard deviation of the design variable. Also, the tool can be used to generate histograms of a parameter both across an individual population and across multiple generations.

Due to the high multi-dimensionality of the problem, it is also important to generate visual aids that can support identification of multivariable integrations. This is done through the main use of two types of plots, multivariates and contours. The graphical interface has the capability to plot a series of variables against each other in order to identify trends in the design space. Also, given a large number of analysis runs that span a design space of multiple variables, a contour plot can be generated that shows the interaction of two variables with a performance parameter. This is enabled by using the SciPy 3D interpolation function griddata. The resulting interpolated grid allows for the generation of a contour plot that can be used in comparing the effect of two interacting variables across the design space.

5.6 Discussion of Implementation

As demonstrated in the previous sections, the Systems Engineering models within SNAPE provide a thorough overview of the architecture and implementation of the navigation system of interest. These tools helped to define the underlying algorithms and provided a basis for data formatting and transfer. These were then implemented in software using the Python programming language. A robust analytical framework was developed to enable a wide range of navigation analyses, and a front-end was implemented to enable ease of user access and data definition, analysis, and visualization.

5.6.1 Implementation Challenges

Over the course of the implementation, several challenges arose that required additional development. The navigation algorithms were initially prototyped in MATLAB. Although this provided very good computational performance, it required a large memory burden,

and was unable to perform a large Monte Carlo Analysis. The design and implementation of the framework in Python alleviated this problem initially. Due to the large number of cases being run as part of the Monte Carlo Analyses and optimizations, small memory leaks within the NumPy mathematical library (due to the integration of underlying C implementations) caused early exiting of the analysis processes.

To remedy this issue and to allow for improved computational efficiency, the SimLink interface was tweaked to allow for threaded operation. This evolved into the use of the Python multiprocessing library. This allowed for the execution of each individual analysis case in its own processing thread. This provided for an increase in computational speed (due to the newly implemented parallelism of the analysis) as well as a decrease in memory usage. This improved efficiency is due to the capability to free up memory after each individual analysis case is performed, limiting total memory in use at any time. This implementation approach was also applied to the DEAPLink interface to allow for increased computational capability. This helps to alleviate the long run times encountered by the simulation software in performing long analysis runs.

5.6.2 Demonstration of Modularity

In order to develop a framework that can be used for a wide variety of analysis cases, it is important to develop the interfaces and software to allow for modular component use and integration. This is needed to provide for a singular definition of the simulation case and allow a variety of analysis to be performed. This is particularly important in terms of the various measurement models, and estimation algorithms. Developing a modular interface enables the quick analysis of a variety of analysis and supports simplified integration of new assets and modeling functions.

The clearest example of modularity can be seen in the base mission scenario template that is used to build up a specific analysis case. This example will focus on the definition of a spacecraft and its embedded properties. As seen in the vehicle definition example defined below, a Python dictionary structure is used to capture the range of variables defining an agent. These keys are defined to match with agent class definition and are used in object

generation and initialization. Each specific variable has a range of properties. As seen, each defined variable can be a function, list, object, or specified value. If the variable is a function or a class (used for passing filter implementations), an additional input called 'lib' specifies the specific library file the class will be loaded from. By utilizing this Python-specific structure, the functions, classes, and variables that form each variables are defined.

Listing 5.1: Input Deck Definition

```
temp_odict = odict()
temp_odict['id']=odict(name='id',type='static',value='MSL',dscr="
    Spacecraft Identifier",dv=0)
temp_odict['sctype']=odict(name='S/C Type',type='class',value='
    Agent_Body',lib='alll_intn',dscr="Spacecraft Model",dv=0)
temp_odict['truth_ref']=odict(name='Truth Reference',type='
    truth_body',value='MSL',dscr="Truth Data Reference",dv=0)
temp_odict['state']=odict(name='State Def',type='static',value='
    posvel',dscr="State Identifier",dv=0)
temp_odict['clock_h0']=odict(name='Clock h_0',type='static',value
    =1.0e-19,dscr="Spacecraft Clock h0",dv=0)
temp_odict['clock_h_2']=odict(name='Clock h_2',type='static',
    value=1.0e-20,dscr="Spacecraft Clock h_2",dv=0)
temp_odict['frame']=odict(name='frame',type='static',value=frame,
    dscr="Data Source Frame",dv=0)
temp_odict['central_body']=odict(name='central_body',type='static
    ',value=central_body,dscr="Truth Data Central Body",dv=0)
temp_odict['grav_forces']=odict(name='Gravitational Forces',type=
    'sel_list',value=('SUN','EARTH','MARS'),dscr="Grav. Forces
    OnBoard",dv=0)
```

```
temp_odict['grav_models']=odict(name='Gravitational Models',type=
    'sel_list',value=('SPICE','SPICE','SPICE'),dscr="Grav. 
    Modeling Source",dv=0)
temp_odict['init_error_pos']=odict(name='Init Pos Error',type='
    static',value=1.0,dscr="Initial Position Error",dv=0)
temp_odict['init_error_vel']=odict(name='Init Vel Error',type='
    static',value=0.1,dscr="Initial Velocity Error",dv=0)
temp_odict['integrator']=odict(name='Integrator',type='integrator
    ',value='dopri5',dscr="Onboard Integrator",dv=0)
temp_odict['reltol']=odict(name='Rel. Tol.',type='static',value
    =1.0e-6,dscr="Integrator Rel Tol",dv=0)
temp_odict['abstol']=odict(name='Abs. Tol.',type='static',value
    =1.0e-6,dscr="Integrator Abs Tol",dv=0)
temp_odict['maxstep']=odict(name='Max. # Steps',type='static',
    value=10000,dscr="Integrator Steps Max",dv=0)
temp_odict['n']=odict(name='n',type='static',value=0,dscr="n",dv
    =0)
temp_odict['filter']=odict(name='Filter Type',type='class',lib='
    alll_intn',value='EKF_8State',dscr='Onboard Filter Type',dv=0)
```

Upon parsing of the user defined scenario by the SimLink library to be passed to the simulation, this input deck is parsed and formatted to match an executable format. For example, Python has the capability to take as a function argument a class instance or a function. When parsing filter objects, the SimLink library generates a filter of the specified library and class, with the defined input variables. This object is then passed to the agent body definition and its functions called throughout the simulation operation. Using common interfaces and predefined class stereotype supports this modularity between components. Similarly, for modeling measurements, function references are passed to capture the objects that calculate the observed value and **H** matrix for a measurement type.

These implemented capabilities of SNAPE are further demonstrated throughout this document through the generation and definition of specific analysis through the graphical user interface, which builds on this modular data definition format. The ability to accept input deck containing links to functions capturing a range of measurements and packet definitions are clear demonstrations of this capability. This enables the integration of a variety of analysis functionality and provides an interface to allow for future simulation framework expansion and development.

5.7 Summary of Implemented SNAPE Capabilities

In order to capture the performance in an integrated environment capturing both navigation and communication models, the functional requirements of the software were implemented. This chapter presents one particular implementation of the SNAPE conceptual framework. These implemented functions form the basis of the capabilities that allow for validation and performance analysis of the navigation system. These inputs and test case definition variables are editable by the user through the implemented graphical user interface.

To provide an overall view and summary of each module's required functionality, Table 12 provides a listing of the major components of the simulation framework as defined in Figure 28. This table links each defined and software block to its purpose and function. The first set of elements are related to the actual running of the simulation and provide the analysis backend. The Simulation Coordinator acts as the high-level structure of the framework, initializing and containing all of the individual simulation agents. Additionally, it serves as the interface between each of the individual elements, tracking their individual states, transferring data between them, calculating the state errors of each, and processing the measurements and packets between and to individual elements. The Data Collector serves as the Data Warehouse, and collects and stores all of the defined data within the simulation.

The next three elements capture the three levels of agent behavior in the simulation. This behavior is inspired by and allows for an ABM simulation approach. The Truth Bodies act as the truth references for each agent, capturing the state of each over the course of

Table 12: Implementation of SNAPE Prototypes

SNAPE Implementation Element	Primary Function
Simulation Coordinator	Initializes elements in simulation, processes packets and measurements, integrates individual simulation components, runs simulation
Data Collector	captures variables of interest, repository for data collected from analysis
Truth Body	tracks true position of agent during simulation
Agent Body	operational agent within simulation with internal processes, estimation algorithms
Estimated Body	tracks Agent Body's onboard estimate of another asset's state
Measurement	defines measurements of vehicle state, assets involved, errors, frequency
Packets	defines contents of packet, source and receiver, errors, frequency
CommSystem	captures communication capability of asset
ClockModel	statistical noise model of clock, propagates clock errors over simulation
ForceModel	calculation of force on Agent, and references for calculation
StatePropagator	propagates state of assets over time, considering force models
StateEstimator	processes measurements and packets, calculates state update, captures dynamics of state uncertainty
User Interface	user definition, generation, running of analysis, processing of simulation results
SimLink	provides interface between graphical user interface and simulation framework
DEAPLink	provides interface between DEAP optimization library and simulation framework

the simulation, typically using external data sources. The Agent Bodies are the primary objects of interest in the simulation. Each has a defined state of behaviors and capabilities captures by its embedded state estimation filter, state propagation routines, and onboard uncertainties, such as timing. An Agent Body can also contain a sub-agent, called an Estimated Body. This captures the capability of the agent to use onboard data to predict and propagate the state of an external body.

Associated with the agent bodies are the Measurement and Packet structures. These contain all of the individual state update information present in the simulation. They define the capabilities of the measurement and packet systems in terms of data content, accuracy, reference, source, and destination, and frequency. The Simulation Coordinator uses this information to generate an agent body's received information, based on its true state and perturbed by the measurement errors. Similarly, the Simulation Coordinator uses the Packet structure to captures the communications possible within the analysis. The Coordinator calculates the true time of arrival of individual packets and maintains a queue throughout the simulation of packets in waiting, releasing them to individual Agents at the true time of arrival.

Each Agent also contains several components that form the base of the analysis and capability of the asset. These are captured in the State Estimation and State Propagation blocks. The state estimator captures a specific algorithm and number of states carried, such as a 6- or 8-state Extended Kalman Filter or Least Squares Estimator. It includes the underlying logic pertaining to the processing of the received measurement, the onboard estimated measurements (based on the vehicle's estimate of truth), and timing. The end result of this is to generate a state update term. At each iteration of the simulation, the spacecraft must also propagate its estimated state. This is performed using the State Propagation block. This block is linked to specific gravity models, and onboard position information to calculate the forces on the asset. The onboard state propagation algorithm (such as the dopri5 used in the current iteration of the framework) captures the numerical precision, accuracy, and intensity of the onboard procedures.

With these elements, it is possible to define an analysis case and simulate a specific

mission scenario. The remaining blocks serve as the interface to the user and linkage routines between individual blocks of the framework and external tools. The graphical user interfaces serves as a model of the analysis approach, allowing for definition of agents, defining data for collection, running simulation cases or optimization, and processing of the output data graphically. The SimLink block provides this interface between the user's definition of the scenario and that required by the underlying simulation framework. The DEAPLink expands this functionality but provides an interface to the DEAP library, allowing for system optimization by utilization of the designated Design Variables and Performance Measures.

Each of these defined elements addresses the framework functionality requirements as defined in Section 3.2. Table 13 references the linkage between each requirement and the blocks that enable and support each. Each block has a unique purpose in the simulation and either individually, or linked together, address the high level capabilities required of the analysis framework. A majority of these functions are built-in capabilities of the Simulation Coordinator. This is due to its role as both a central process and interface between the various simulation elements. The Agent, Truth, and Estimated Bodies form the population of assets within the framework, and allow for the separation of true state from vehicle onboard estimates. The State Propagation and State Estimation contain the bulk of the complex analysis, allowing for predicting the asset's future state as well as processing measurements and packets into state updates. The modularity of these functions allows the analysis and integration of a range of state estimation and propagation algorithms and approaches. These elements all work together to allow for the integrate simulation capability.

The external software elements are used to provide an interface to other tools and analyses. For example, the DEAPLink acts as translator and reference to the DEAP library allowing for optimization of the defined vehicle parameters. Similarly the SimLink library provides the lowest level direct access to the simulation routines by acting as an interpreter between user defined and framework-specified input decks. The SNAPE front-end extensively utilizes the SimLink interface and serves as the user's interface to the simulation framework. This developed tool enables the definition, editing, saving, and loading of a

Table 13: Implementation of Framework Requirements

Framework Requirement	Implementing Element(s)
1.1 Initialize body to input data	Simulation Coordinator
1.2 Compare estimated to reference/truth data	Simulation Coordinator, Data Collector
1.3 Calculate non-inertial forces on body	Force Model
1.4 Capture inertial forces on body	Force Model
1.5 Integrate body's onboard state	State Propagator
1.6 Modular measurement interface	SimLink, Coordinator, Measurements
1.7 Modular state estimation interface	SimLink, Coordinator, State Estimator
1.8 Calculate State Errors as a function of time	Simulation Coordinator, Data Collector
2.1 Perform Deep Space Link Analysis	Simulation Coordinator, Comm System, Agent Body, Truth Body
2.2 Autonomous Packet Generation	Agent Body, Packets, Coordinator
2.3 Capture Transmission Delays	Coordinator
2.4 Autonomous Reception and Processing of Packet	Agent Body, Coordinator
2.5 Integration of Packet with State Estimator	Agent Body, State Estimator
2.6 Onboard estimation of other SC states	Agent Body, Estimated Body
3.1 Interface to external measurement models	SimLink, Simulation Coordinator, Agent Body
3.2 Model Autonomous or Scheduled Measurements	Coordinator, Agent Body
3.3 Process measurement into state estimator	Agent Body, State Estimator, Simulation Coordinator
4.1 Model and vary packet content and measurements	SimLink, DEAPLink
4.2 Integration with Monte Carlo tools	SimLink, DEAPLink, Simulation Coordinator
4.3 Capability to perform packet optimization	DEAPLink
4.4 Modular interface to external design tools	DEAPLink, SimLink, Simulation Coordinator
5.1 Support a range of analysis scenarios	All Elements
5.2 Modular input and interface to test cases	User Interface, SimLink, Simulation Coordinator
5.3 Robust framework to variety of studies	All Elements

specific scenario, capturing all assets of analysis from agent generation to truth source definition to data collection and simulation parameters. Additionally, the front-end implements interfaces to the DEAPlink library and graphical plotting routines to allow for design space exploration and visualization.

Table 14 compares these capabilities of the implemented SNAPE framework to the currently available tools and their capabilities. Focusing implementation on these identified requirements ensures that the simulation tool will be able to meet the analysis needs and address the identified research questions. Additionally, this serves to provide a summary of the implemented functionality.

The table displays how the developed tool fills the analysis gaps of the other available packages. SNAPE provides a unified simulation architecture that allows for definition and analysis of a range of analysis missions, and deep space navigation and communication concepts. This allows for detailed simulation of deep space navigation filters, both at conceptual and computational level, a variety of measurement and observations types, and onboard computational capabilities. These capabilities are enabled through the use and integration of MBSE (to capture and define the system) and ABM (to capture and simulate individual object behaviors) approaches to navigation system design. This approach results in the development of a unified framework, SNAPE, that can capture the overall navigation system of interest, how it operates, and its implementation, providing linkages from operational system requirements to detailed functional simulation.

Table 14: Comparison to Framework Implementation

Framework Requirement	ODTBX	STK	Open-SESSAME	ION	SNAPE
1.1 Initialize body to input data	√	√	√		√
1.2 Compare estimated to reference/truth data	√	√	√		√
1.3 Calculate non-inertial forces on body	√	√	√		√
1.4 Capture inertial forces on body	√	√	√		√
1.5 Integrate body's onboard state	√	√	√		√
1.6 Modular measurement interface	√		√		√
1.7 Modular state estimation interface	√				√
1.8 Calculate State Errors as a function of time	√				√
2.1 Perform Deep Space Link Analysis	√			√	√
2.2 Autonomous Packet Generation					√
2.3 Capture Transmission Delays	√			√	√
2.4 Autonomous Reception and Processing of Packet					√
2.5 Integration of Packet with State Estimator					√
2.6 Onboard estimation of other SC states		√			√
3.1 Interface to external measurement models					√
3.2 Model Autonomous or Scheduled Measurements	√				√
3.3 Process measurement into state estimator	√				√
4.1 Model and vary packet content and measurements					√
4.2 Integration with Monte Carlo tools	√	√			√
4.3 Capability to perform packet optimization					√
4.4 Modular interface to external design tools	√	√	√	√	√
5.1 Support a range of analysis scenarios	√	√	√	√	√
5.2 Modular input and interface to test cases	√	√	√	√	√
5.3 Robust framework to variety of studies	√	√	√	√	√

CHAPTER VI

VERIFICATION OF SNAPE IMPLEMENTATION

This chapter utilizes the implementation of the SNAPE framework in order to demonstrate both the processes and analytical capability of its application to an example problem. The analysis case here serves as a simplification of the NNAV architecture in order to provide insight into the underlying physical processes. Additionally these example analyses can be used to compare the simulation's performance with other tools in order verify its proper operation. These verification cases will provide increased confidence in the computational results. This formulation will allow for design studies and optimization of the parameter space that will produce logically expected results.

The specific operational concept focuses on the navigation capability for a spacecraft on a transfer orbit from Earth to Mars. The example trajectory used is the publicly available as-flown trajectory of the Mars Science Laboratory Curiosity[1]. This is used to provide a relevant baseline for error calculation and mission analysis. This implementation will serve as an example to demonstrate the described methodology for a simple verifiable analysis case. This test case only captures the effect of state measurements, and seeks to capture the frequency and type of state observations that can provide optimal performance, by means of reduced number of measurements and minimized long-term navigation error. This will demonstrate the main functionality of the framework.

6.1 Overview of Test Cases

In order to provide an analysis case similar to and applicable to the design problem, this example implementation focuses on measurement updates of a vehicle during interplanetary cruise. Navigation updates will take the form of regularly scheduled state updates from Earth with defined error characteristics. As mentioned above, the published MSL trajectory

[1] http://naif.jpl.nasa.gov/pub/naif/MSL/kernels/

is used as the truth reference for the agent. Initial errors to the position and velocity state are defined in terms of the standard deviation and applied to the truth state to initialize the agent's internal state knowledge. This was done to utilize a relatively stable design case that can be modeled by a wide range of tools to allow for experiments, demonstration of the framework implementation, and comparison to the results of other tools.

The example spacecraft serves as a representation of a Mars-bound vehicle. Power requirements are neglected in this analysis, focusing instead on the performance of various measurement methods and the relationship between various aspects of the navigation system. The estimator is implemented as an Extended Kalman Filter, with multiple state options available. The implemented state estimator utilizes eight vehicle states, capturing the three position, three velocity, clock bias, and clock drift of the spacecraft. For the example cases, the planetary bodies included in the onboard state estimator are the Sun, Earth, and Mars. The gravitational data is obtained from the SPICE libraries, and consists of first order terms. This data also provides for gradient calculations which are included in the state transition matrix for error and covariance propagation. For propagation, the framework utilizes the SciPy built-in ODE integration class with the explicit Runge-Kutta 4(5) integration algorithm [56]. The measurements being traded in this analysis include measurements of full state, position, and range. Additionally, it is possible to study the capability of clock measurements and time updates as well.

6.2 General Assumptions and Analysis Approach

For this analysis, the truth data for the planetary bodies is loaded from the DE421 ephemeris library. The vehicle truth data reference is the published MSL trajectory, specifically the mission planning and operational flight kernel. Over the course of each simulation, the truth information is set to the results of a query on the loaded ephemeris libraries for each truth body. For the spacecraft agents, the position and velocity errors are the difference between each onboard state estimation solution and the reference trajectory (as defined by the loaded data). The states of the spacecraft agents are initialized to be equal to the truth at the simulation start time with stochastic errors applied. These initial errors are defined

in the simulation input deck. The timing errors are defined by the difference between the onboard agent internal clock estimate and the true simulation time.

When calculating a measurement update, such as position or state, the vehicle's truth reference data is used as the source of information. The measurement is calculated based on this information with a stochastic term assuming a normal distribution of error, with zero mean and an input standard deviation. These stochastic parameters are applied to each dimension of the measurement to form the actual processed measurement. It is assumed that this data can be generated in terms of batches at set intervals, where the time between batches, number of measurements in a batch, and time between individual measurements in a batch are defined in the input deck. The results focus on the performance of the implemented state estimation filter to various types of information and robustness to measurement errors and content. This performance is captured in terms of normalized state position errors in terms of true spacecraft state vs. the that estimated onboard through the use of the embedded filters.

Over the course of a simulation, the measurements and packets are generated and transmitted to the specific agents at the predefined intervals with the errors applied. Upon reception of a measurement, a spacecraft agent uses its internal ephemeris models (for example, an onboard DE421 library) and onboard dynamics models to propagate its state and covariance to the time of observation. The specific forces used are defined in the input deck. For these simulation cases, the SPICE planetary physical parameters are used to capture gravitational parameters. The agents each perform a Runge-Kutta 4(5) integration with absolute and relative tolerances of 1e-6 assuming gravitational effects from the Sun, Earth, and Mars. Though most onboard estimation routines utilize fixed step algorithms of lower order, the integration algorithm is chosen to provide increased efficiency and decreased runtime over long simulation time scales. Specific measurement parameters are given in the following sections for each analysis case.

One assumption of the analysis process is that the state measurements are the result of an external orbit determination measurement process. For the Mars transfer case, it is assumed that these are the result of a DSN navigation pass. The ground observations

are used in orbit determination routines to determine the spacecraft's position and velocity state. Ground sources are then able to predict the vehicle's state to high accuracy and transmit this information in packet form to the vehicle. Errors are applied to match that achievable by DSN processes, on the order of one kilometer and one tenth kilometers per seconds [121]. Upon reception, this information is parsed by the state estimation algorithm as a measurement of position and velocity [74]. This method of analysis allows for capture of the overall accuracy of the ground-based observations without requiring detailed reproduction of the orbit determination routines and observations. These assumptions capture how the update would be transmitted operationally to a vehicle with an onboard navigation filter. Additionally, this allows for capturing the autonomous aspects of the navigation filter, in that it operates independent of Earth-based orbit determination. Although the state measurements are generated by external sources and transmitted to the vehicle, this chapter focuses on the autonomous nature of the navigation filter. This is due to the onboard implementation of the filter, and independence of the state-to-state propagation and measurement application and processing independent of ground-based orbit determination.

6.3 Framework Functional Verification

In order to verify the functional performance of the implemented models, baseline analyses[2] are prepared for comparison to standard results. There are two main aspects of the simulation which require functional verification: the state propagator and the state estimation filter. These two functions form the basis of the navigation analysis simulation and their verification will provide additional confidence in their resulting capabilities. These runs are also considered to be calibrations of the software to ensure proper operation for further analysis cases, which do not have any reference data available for comparison.

6.3.1 Orbit Propagation Analysis

A key component of the analytical framework is the orbital dynamics propagation model. The equations of motion are implemented as described in the previous chapters, and utilize

[2]Test cases to support functional verification are identified by F1, F2...

a Runge-Kutta 4(5) integration algorithm. To confirm the analytical capability of the implementation it is compared against a variety of other state propagators. The software validation package used is AGI's STK. A large variety of propagators are built into this industry standard package and as such it is well suited to this verification testing. Several specific propagators are selected for comparison against the representative truth state, which is being represented by the published MSL trajectory.

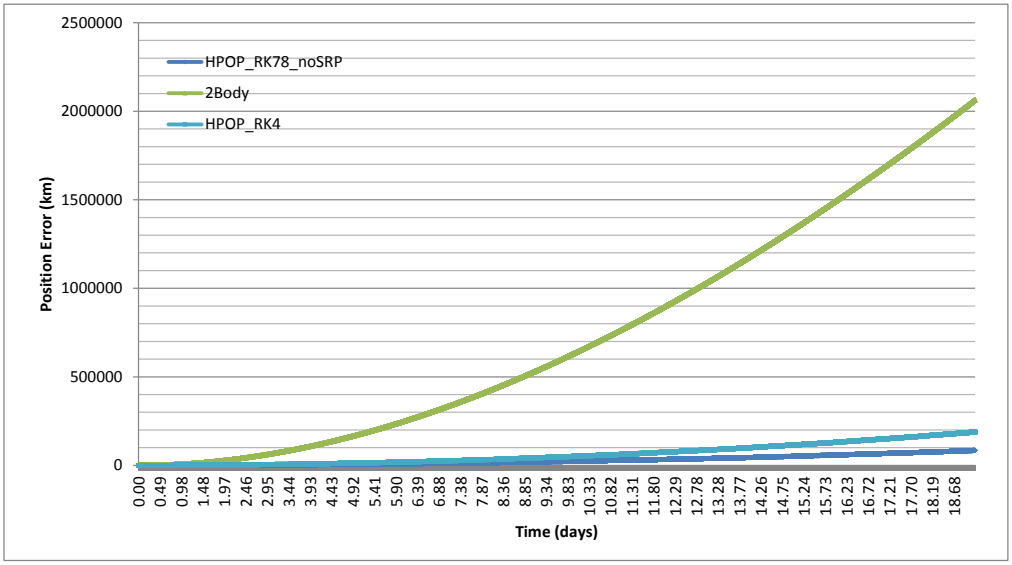


Figure 41: STK Propagator Performance

To study the propagation capability all analysis cases were initialized to the same initial value, the Sun-centered J2000 Cartesian position and velocity at ET 379605666.18423 or January 12, 2012, 2:00:00 UTC. The source of this initial data is the published MSL operational trajectory. The onboard states were propagated forward for 20 days and the results

compared against the truth state. The specific integrators chosen capture a variety of capabilities. A two body propagator was chosen as a worst case propagation scenario including only the gravity effects of the Sun (F1). Several implementations of the built-in HPOP propagator were considered for analysis. All of these cases also included the gravitational effects of Earth and Mars. These are identified as follows: using a Runge-Kutta 4-order algorithm (HPOP_RK4) with solar radiation pressure (F2) and using a Runge-Kutta 7(8)-order algorithm without solar radiation pressure (F3). Initially an RK78 with this stochastic force was included for analysis, but was shown to provide similar errors to the RK4 case. The main difference between the two is in the required number of iterations to maintain a chosen error level. The comparison of these integrations is given in Figure 41. As seen in the diagram the third body gravitational effects play a large part in improving the onboard estimator. Additionally, there is not any clear benefit to including solar radiation effects with this level of state propagation capability. This may be due to errors in the pressure model and incorrect sizing to the MSL mission. Another result may be that the operational scenario does not include solar radiation effects, or the observed effects were minimal for this mission. Further analysis can remedy this and further improve the propagation capability.

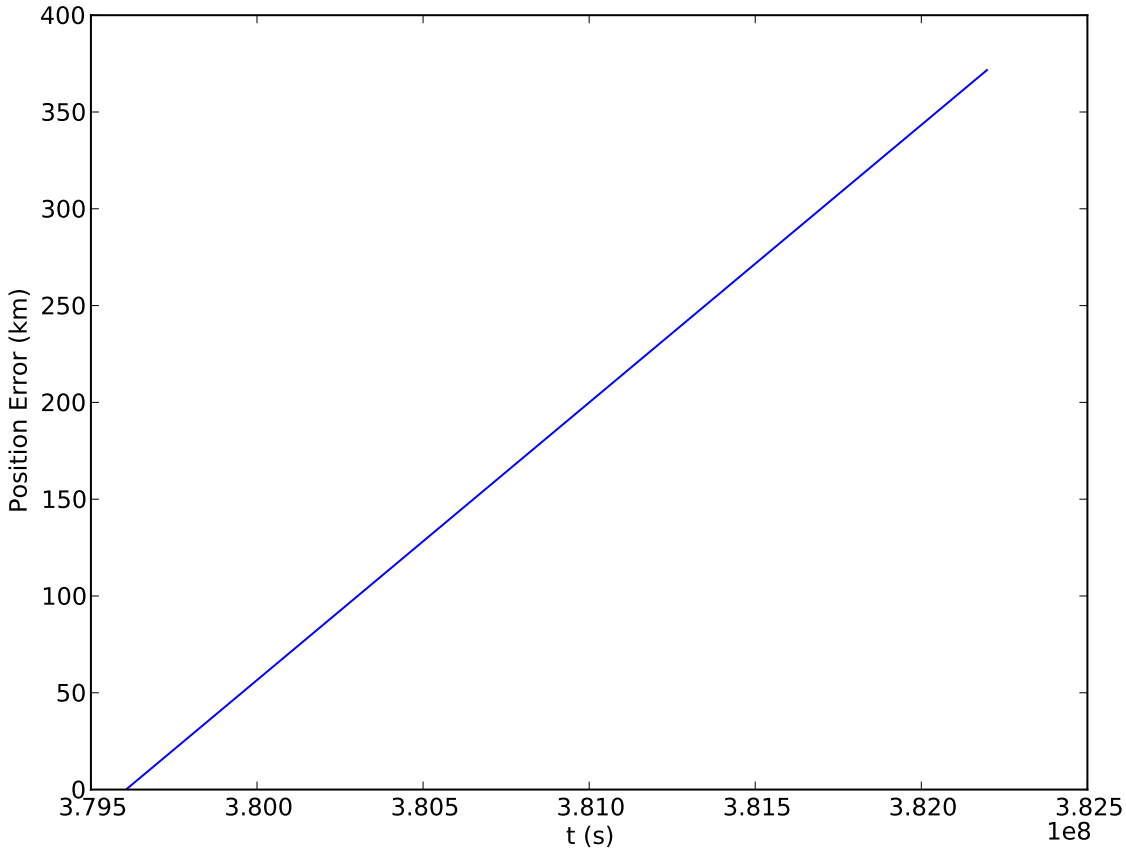


Figure 42: Framework Propagator Performance

The performance of the framework's built-in integrator (F4) is plotted in Figure 42. The simulation was performed with the same initial conditions. The difference in displayed timescales is due to the simulator's operation in ET time. This displays the capability of SciPy's built-in dopri5 algorithm. Additionally, the performance compares well with the STK analysis cases. This comparison provides a quick verification of the framework's propagation capabilities.

6.3.2 Estimator Performance

In order to validate the performance of the navigation algorithms, a similar analysis was performed using ODTBX. A simple simulation case was designed that could be compared across each software package utilizing a similar estimation algorithm. For this study, the base example case for the sequential estimator was ported to the navigation framework.

This analysis case consisted of a predefined equatorial orbit, and captured range and range-rate measurements at ten-second intervals over a five-minute ground station pass. The initial errors were synchronized between the packages. The truth data used for both the framework and ODTBX is generated utilized restricted two body dynamics of the Earth and the spacecraft. Given the initial position and velocity, the orbit was propagated 300 seconds and used as the truth reference. This data was then exported and converted into SPICE SPK format to allow for integration with the simulation framework. The initial uncertainty in each position dimension was set to 1E-4 km and for each velocity dimension to 1E-5 km/s.

Additionally, the initial filtering parameters from ODTBX were slightly modified to allow for exact comparison to the navigation framework input. An example of this is to assume equal initial covariance in each of the position axes. The measurements were modeled with the same error standard deviations and process noise filter parameters of 1E-3 km in range and 1E-6 km/s in range rate.

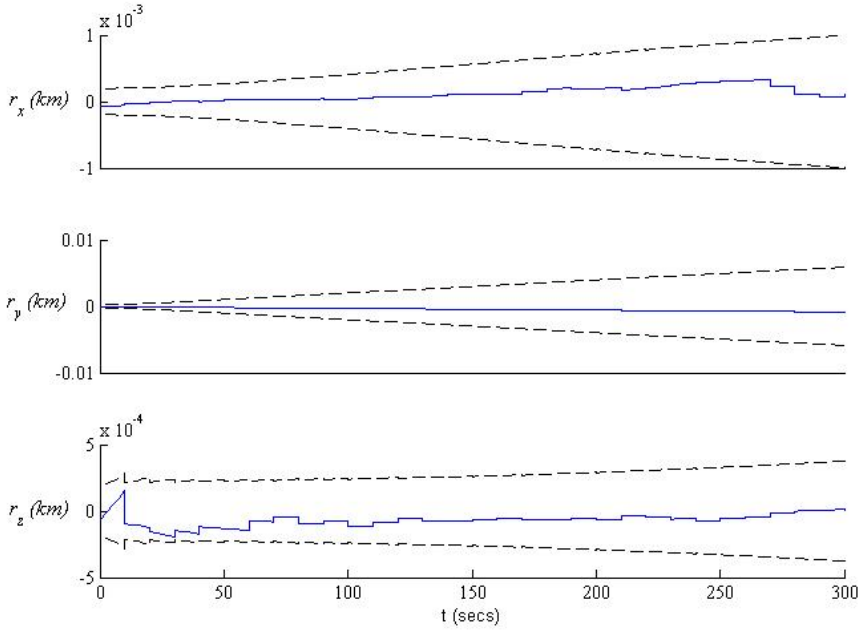


Figure 43: Position Estimation Errors for ODTBX

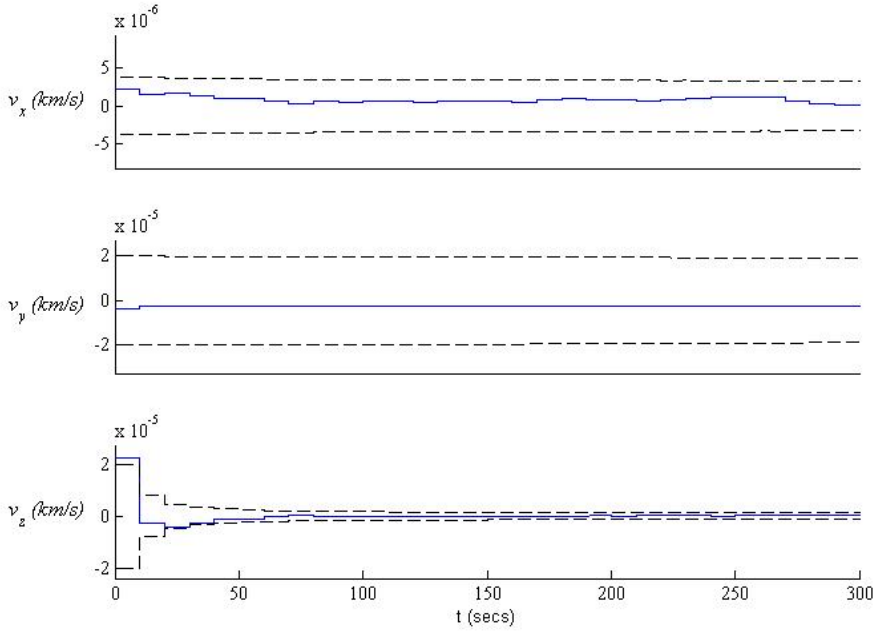


Figure 44: Velocity Estimation Errors for ODTBX

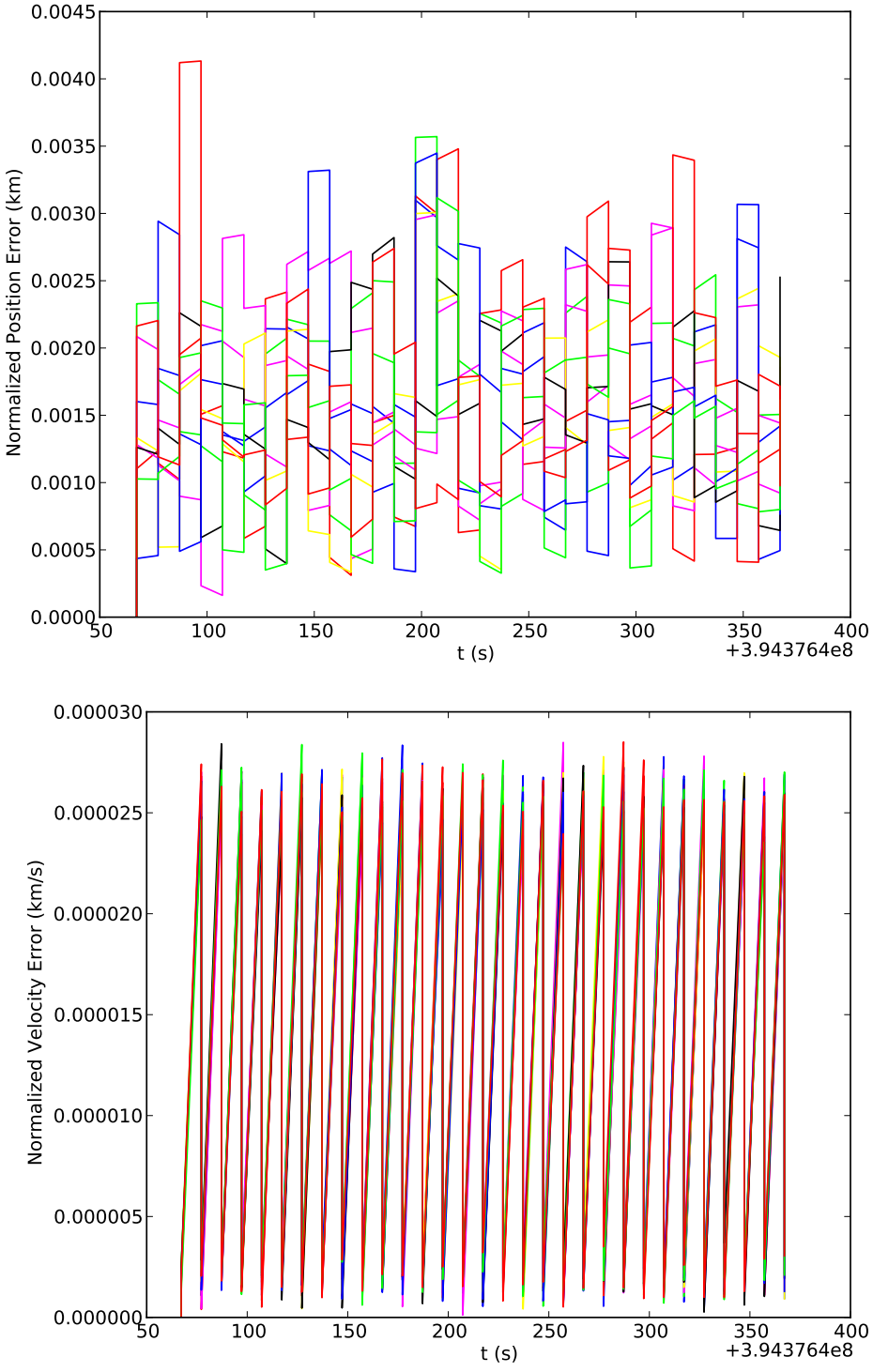


Figure 45: Position and Velocity Normalized Estimation Errors for Framework

The results of the ODTBX analysis (F5) are shown in Figures 43 and 44. The corresponding errors from the framework's estimator (F6)is captured in Figure 45. The two

different estimators provide similar order of magnitude errors for the estimated state for this simple analysis case. It was attempted to use ODTBX to validate MSL error estimation. To enable the integration of SPICE trajectory as a truth reference to be used requires modifications and additions to the software package. For this reason, this analysis provided here serves as a simple validation of the capability of the filter. Additionally, the performance of the filter will be directly captured through the simulation framework by comparison to the truth state for a variety of measurement updates.

6.4 Framework Implementation Validation

With the individual functions verified against other analysis packages, it is now possible to do further studies to gain more insight into the deep space navigation process. This case study allows for a survey of the capabilities of the implemented state estimator to a range of input scenarios. These test cases[3] have been chosen to build intuitive understanding of the navigation process and the impact of measurement frequency, content, and error. These analysis cases also further reinforce the capabilities of the navigation system and simulation framework. While the above scenarios provide a verification of the analysis functionality, the test cases serve to demonstrate the capability of the simulation framework, and act as verification of the development. As such, they show that the tool can be used to answer the research questions needed in order to test the previously stated system-level hypotheses.

6.4.1 Effect of Initial Error

The first analysis (V1) focuses on the effect of the initial error of the onboard state. For an implemented scenario, the spacecraft will not have perfect knowledge of its state for there are always higher order forces which have been neglected in state propagation as well as simple numerical roundoff errors that limit exact knowledge. It is assumed that the transferring spacecraft will spend a set amount of time in local Earth orbit to perform checkout operations prior to injection into a Martian transfer orbit. During this time, the spacecraft will also be required to determine its initial state. This will typically be done via

[3]Each test case supporting validation is identified as V1, V2...

Table 15: Initial State Error Analysis Space

Variable	Min	Max	Units	Distribution
Initial Position Estimate Error	1E-2	1E4	km	Log Uniform
Initial Velocity Estimate Error	1E-4	1E0	km/s	Log Uniform
Time Between State Measurements	86400(1)	1209600(14)	s(day)	Uniform

ground-based assets, such as radar tracking and initial DSN observation. GPS observations could also be possible in local Earth orbit, although these receivers are not typically installed on deep space craft due to their limited usage and vehicle size and volume constraints.

It is not possible to directly calculate this relationship due to the stochastic nature of the problem, particularly the state estimation process. Therefore a Monte Carlo Simulation of the design space is performed to allow for analysis of a wide range of input values to capture a wide swath of potential error states. This case focuses on the effect of initial error with a fixed set of state updates. These state updates are assumed to be standard DSN position and velocity measurements with a fixed error. The parameters under study include the statistics of the initial position and velocity error as well as the time between ground-based state updates. This will allow for analysis of the robustness of the filter to initial input noise and verify its performance for a fixed input case. The ranges of input variables are given in Table 15. In the table, the notion Logarithmic Uniform is used to identify a range of values that are distribute linearly across the given orders of ten.

The results of this analysis focused on the capture of integrated position and velocity as a function of the input parameters. To analyze the space, a Monte Carlo run of 500 cases was performed. The results are plotted below in Figures 46 and 47. The first figure shows the lack of a correlation between the initial state and the computed velocity error. A similar effect was observed for position error as well. This demonstrates the robustness of the filter to starting error and shows that the filter's convergence is not dependent on the initial state. This is an expected behavior as the filter is designed to enable the update of state for a range of initial conditions. Comparatively in Figure 47, the time between measurements has a very strong effect on the integrated position and velocity terms. This is also expected, as the increasing time between measurements provides a longer amount of

time for the estimated state to be propagated forward, along with its errors, which increase over time.

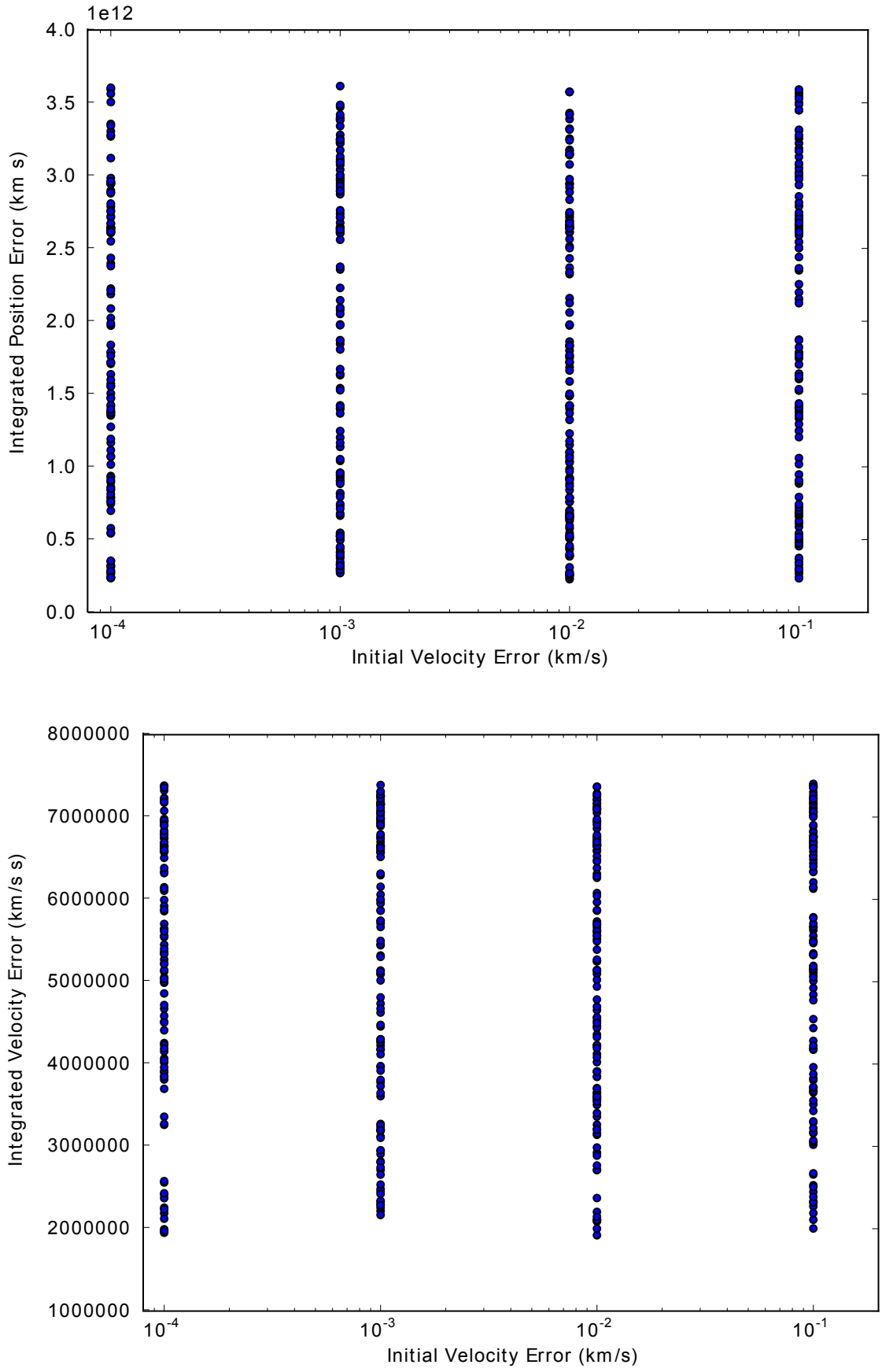


Figure 46: Effect of Initial Velocity Estimate Errors on Integrated Error

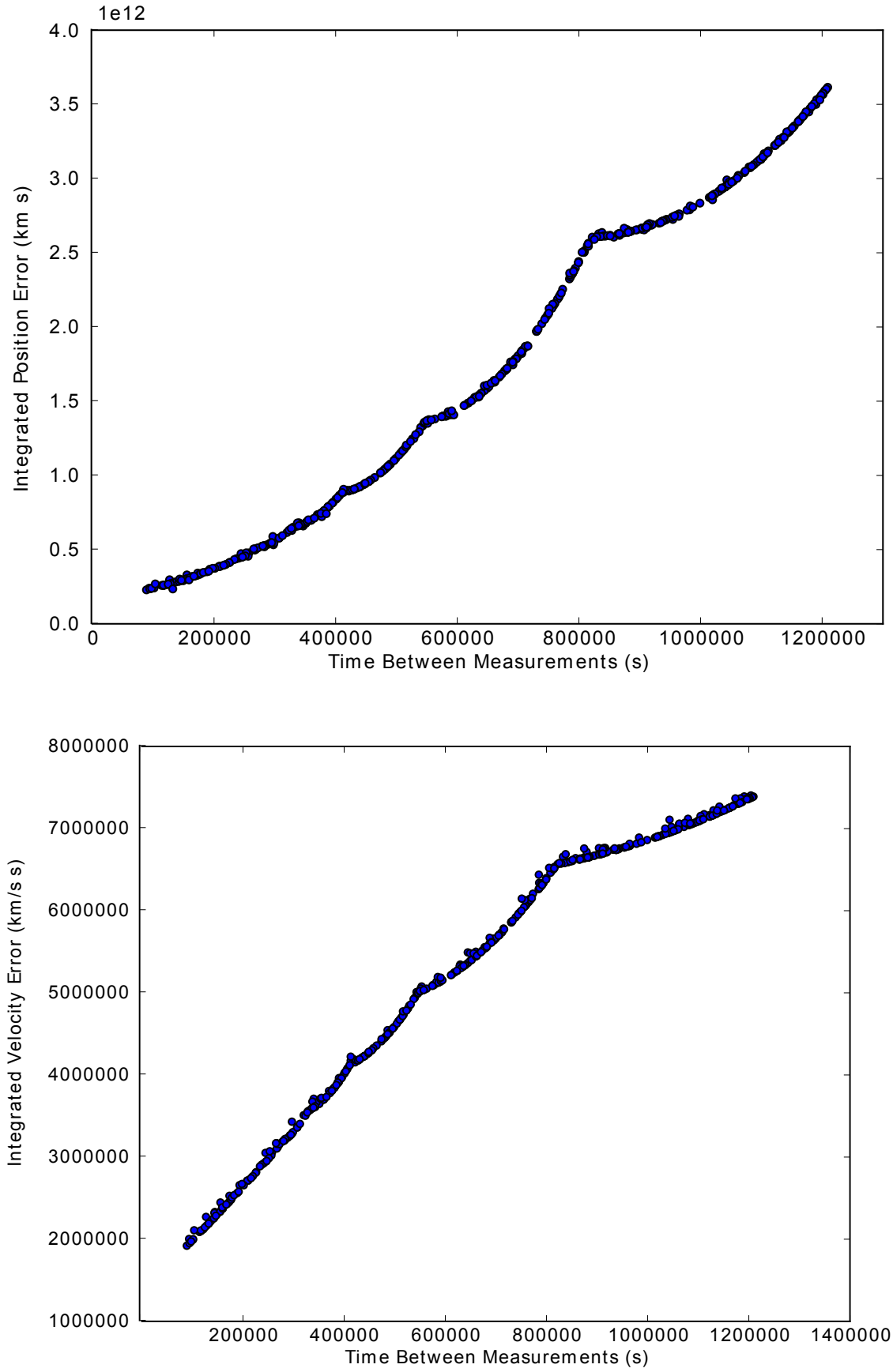


Figure 47: Effect of Time Between Measurements on Integrated Error

Table 16: Position Measurement Error Analysis Space

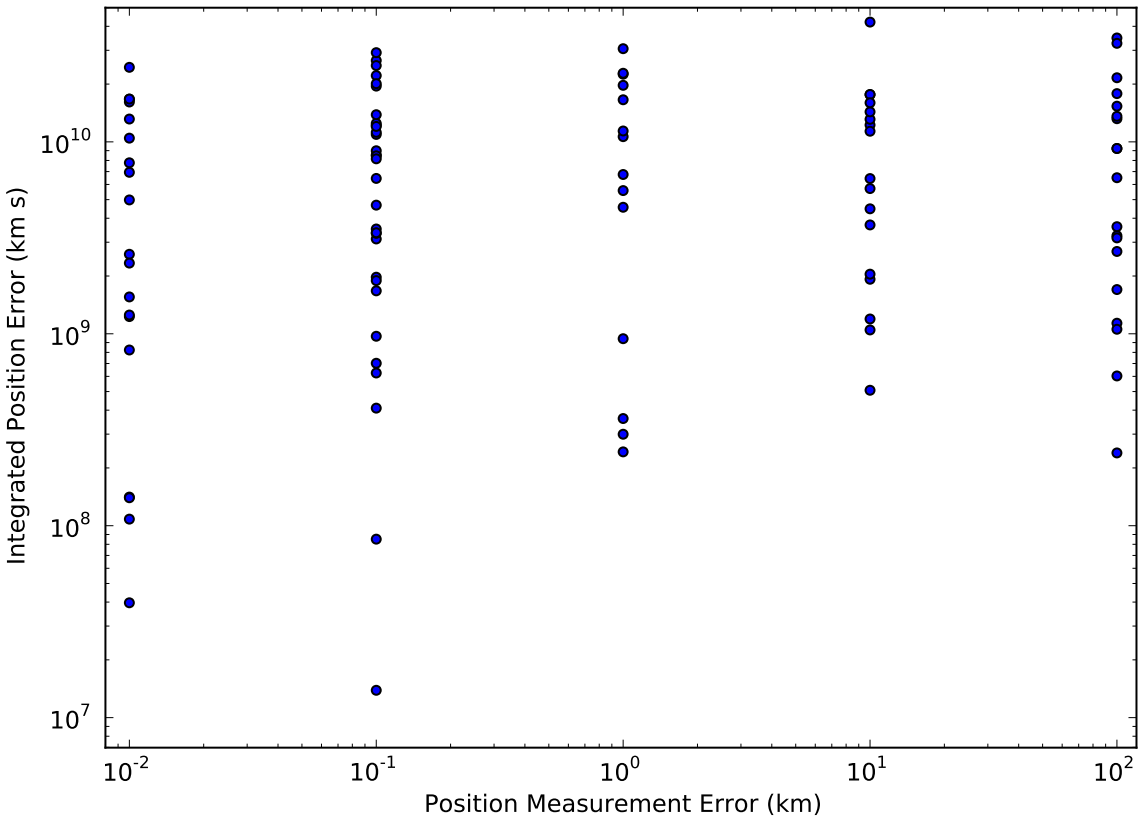
Variable	Min	Max	Units	Distribution
Position Measurement Error	1E-2	1E3	km	Log Uniform

6.4.2 Effect of Measurement Error

The second analysis case (V2) investigates the robustness of the state estimation design to errors in state measurements. The same example trajectory and spacecraft implementation as given above is again used as the basis of this study. The measurement in this analysis case focuses on a position update. Neglecting a velocity state update allows for a more straightforward analysis of the capability of the filter to process input measurement noise. The design parameters in this case focus on the accuracy of the state updates with a variable update frequency. For this study, it is assumed that the filter has been tuned to expect a certain level of noise in its processed data. The specific variables and their ranges are given in Table 16.

The results for the study are summarized in Figures 48, 49, and 50. The first set of plots demonstrates the effect of the error of the position measurement on the integrated error over the course of the simulation. This plot shows the measurement noise and the integrated state estimation error on logarithmic scales. The main conclusion from this visualization is that the estimator's performance is not directly tied to the error of the measurements, reinforcing behavior already described. This is further supported by Figure 49, showing a strong correlation between the time between measurements and the estimation error. This also matches with theory, as the increased times allow for a longer integration of state errors between measurement updates. This observed behavior is also shown to logarithmically increase for small dt's, reaching a stable upper value driven by early estimation errors. This behavior can be observed in Figure 50, where the position errors as a function of time are displayed. For the top plot, the time between measurements is fixed, and thus the position measurement term drives the stable value of the estimation error. This is demonstrated by the time-based plots stacking at various of ten. The bottom chart allows for a varied time between measurements, and the effect on integrated error is due in large part to the

propagation of the initial error to the first measurement update. Also the time between measurements can be seen to have a strong effect on the dynamic noise characteristics.



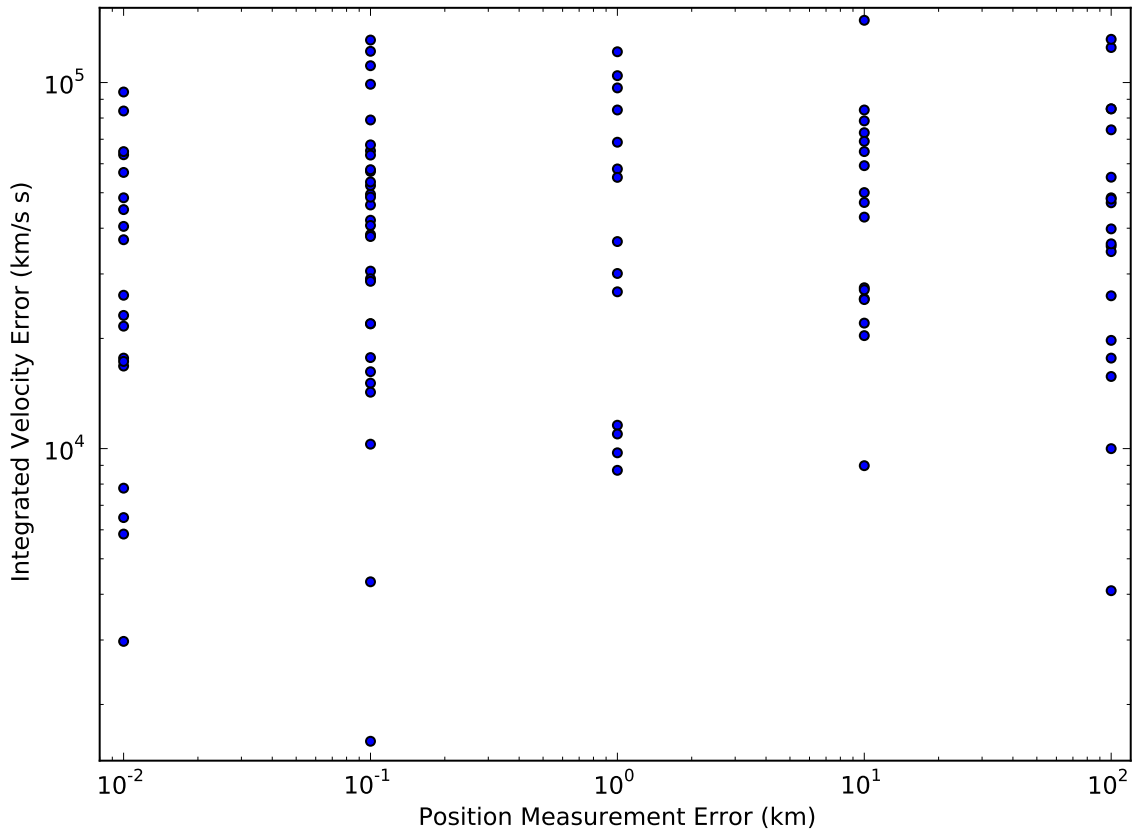
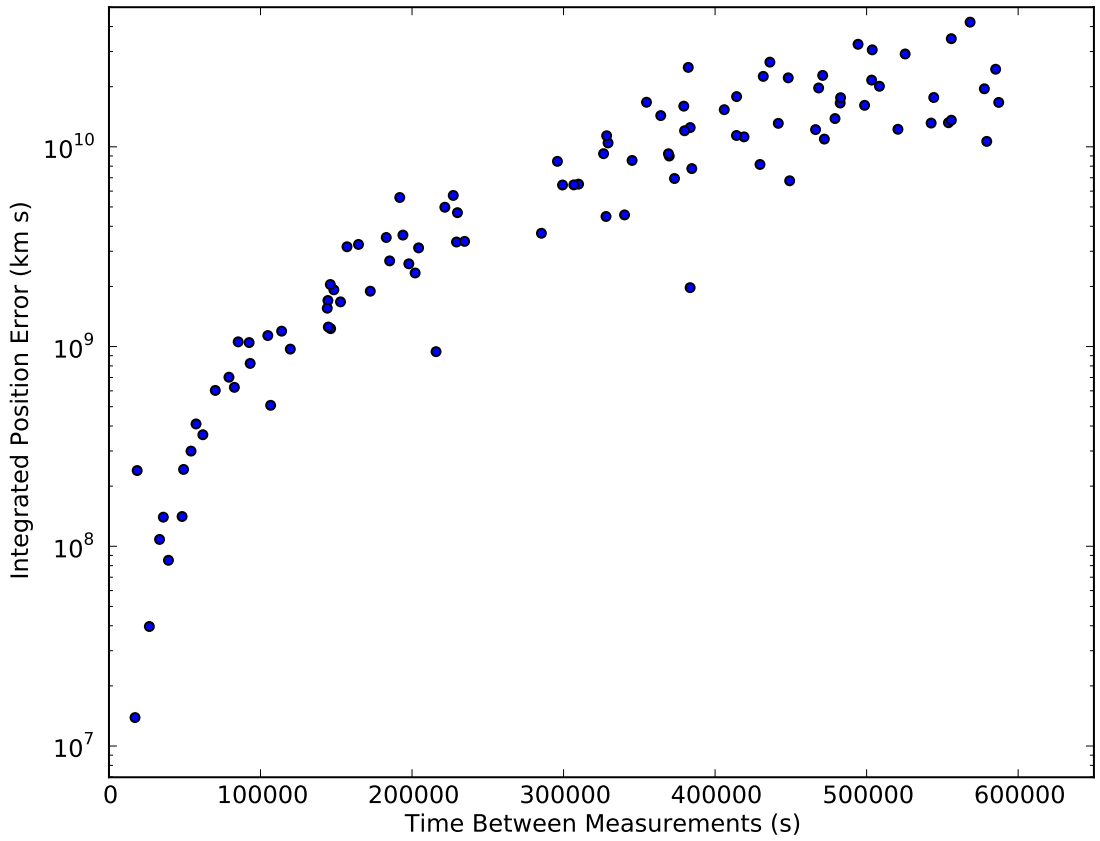


Figure 48: Effect of Position Measurement Error on Integrated Error



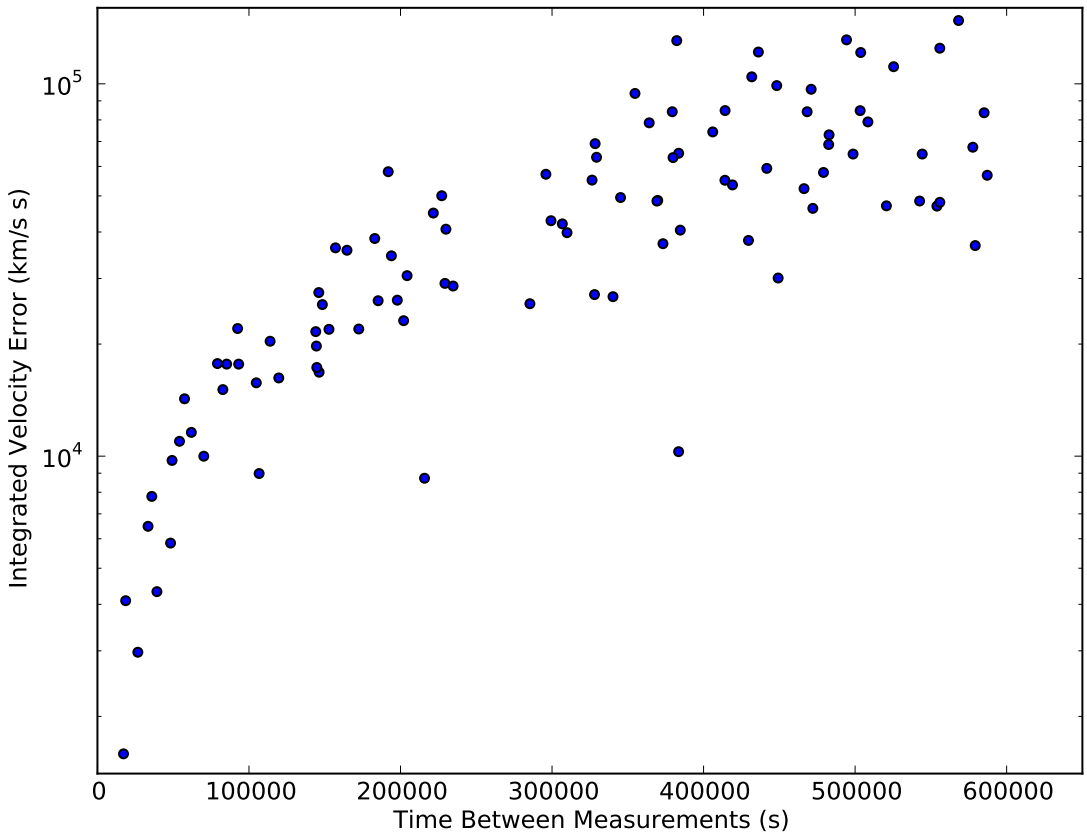


Figure 49: Effect of Time Between Measurements on Integrated Error

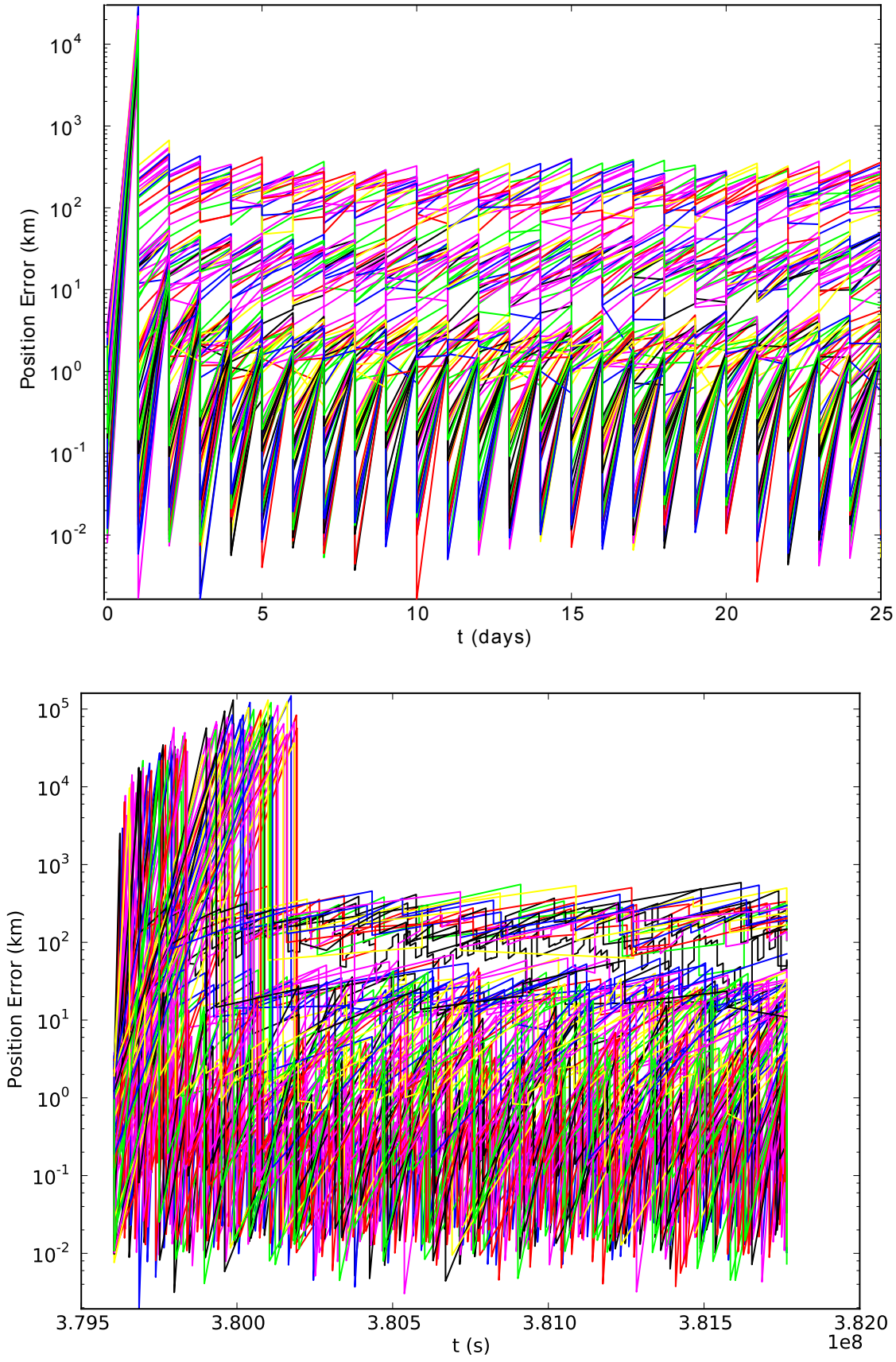


Figure 50: Comparison of Estimation Error for Fixed Dt (top) and Variable Dt (bottom)

Table 17: Timing Behavior Input Variables

Variable	Value
h_0	1E-19
h_2	1E-20
Time Between State Measurements	608400 s
Time Between Position Measurements	86400 s
Time Between Clock Measurements	86400 s

6.4.3 Timing Behavior Analysis

Another important factor in the proposed timing-based measurement approach is to analyze the capability of the onboard estimator to both track and correct for stochastic oscillations in the spacecraft's clock. The timing stability is captured using the timing errors outlined previously. For this case, two estimators are compared: a six state version which only contains position and velocity factors (V3) and the eight state filter with estimation of clock bias (V4). The performance of these two will be compared to capture the effect of timing stability on state estimation errors for a fixed measurement set of weekly state and daily position updates. This test case represents processing of weekly DSN updates with daily state measurements during communication windows.

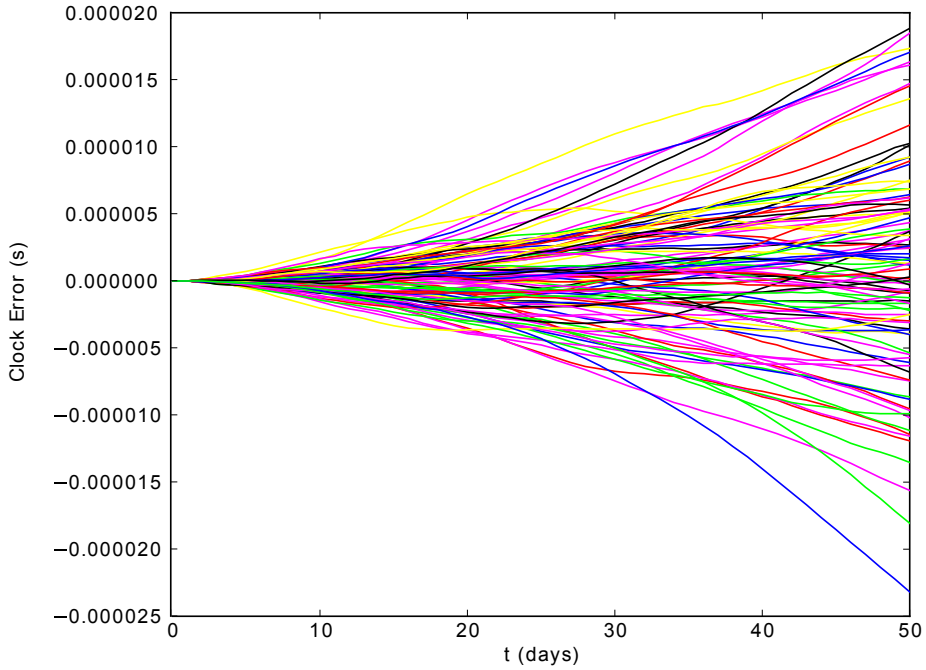


Figure 51: Dynamic Clock Behavior without Measurements

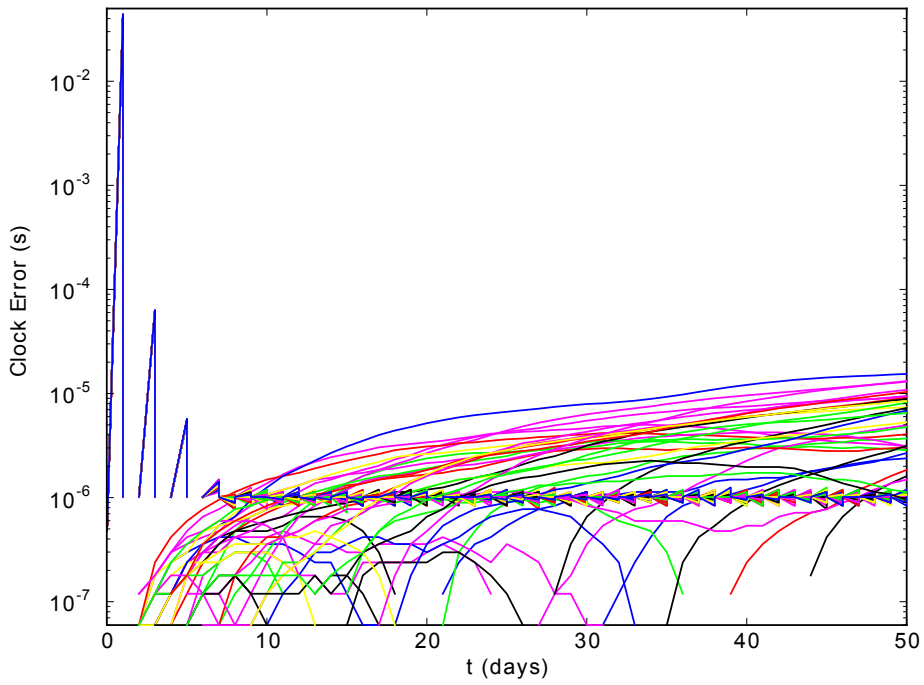


Figure 52: Dynamic Clock Error with Measurements

Table 18: Timing Measurement Analysis Space

Variable	Min	Max	Units	Distribution
h_0	1E-24	1E-15	n/a	Log Uniform
h_{-2}	1E-24	1E-15	n/a	Log Uniform
Time Between Measurements	86400(1)	604800(7)	s(day)	Uniform

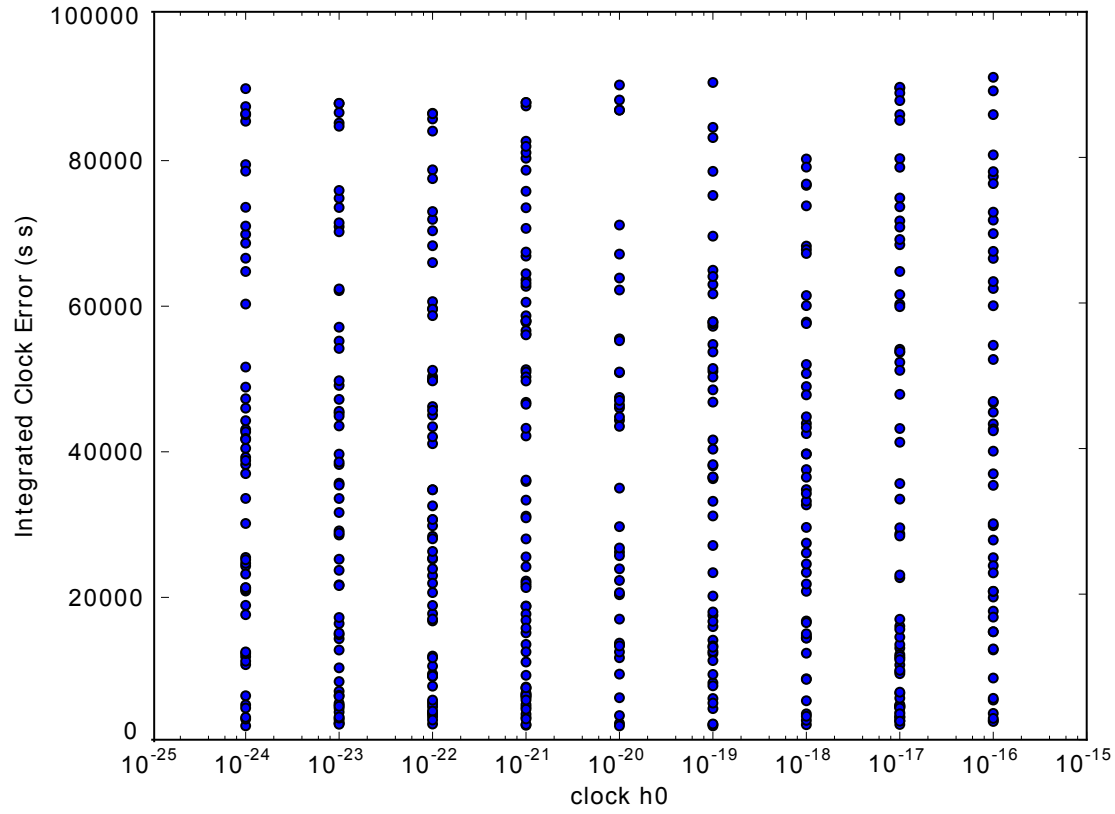
The stochastic nature of the clock's behavior over time is shown in Figure 51. For this measurements case, the simulation parameters are set as given in Table 17, except that the state updates are disabled. This seeks to capture the dynamics of the clock state. As seen, due to the lack of measurements to be processed, the two filters exhibit very similar behavior. Figure 51 thus verifies that both have similar dynamic clock properties.

In comparison, Figure 52 includes the effect of state measurement updates. Both filters utilize state updates. Additionally, the eight state filter is capable of processing timing measurements to process through the filter. The benefit of this is clearly seen in the resulting plots of clock error over time. Due to the lack of filter tracking, the errors of the non-updated state can be seen as continuing to grow over time. It is also shown that with measurement updates, particularly of clock state, the filter is able to maintain a stable clock error over time. This shows the benefits of including the clock parameters in tracked spacecraft state.

An additional factor under analysis is to capture the need for Earth-based timing measurements (V5). Pure timing state measurements allow for greater stability and enhances position and state estimation by improving the accuracy of state estimation procedures as well as onboard estimates of other vehicle state. This capability is analyzed by varying the accuracy and frequency of clock bias measurements. This factor is calculated by comparing the transmitted expected time of arrival and the receiving spacecraft's clock at the time of reception. Variables and their input ranges are described in Table 18.

The results of the Monte Carlo Analysis are given in Figures 53, 54, 55, and 56. The first two scatter plots visualize the interaction between the clock stability terms and the integrated position and velocity estimate errors. These visualize the robustness of the state estimation routine to the clock's stability, captured by its ability to model, predict, and correct the clock's estimated bias and drift parameters. The larger factor in the clock

correction performance is due to the time between clock measurements. Figure 55 clearly demonstrates the effect of more frequent clock updates, reducing both integrated clock and position error. The last figure visualizes the dynamic capability of the estimator to correct the estimated clock value in reference to true time. In Figure 56, the clock error is plotted on a logarithmic scale to show the stability of the state estimation algorithm. With clock measurements, the filter is able to maintain a high level of accuracy in the corrected onboard timing prediction.



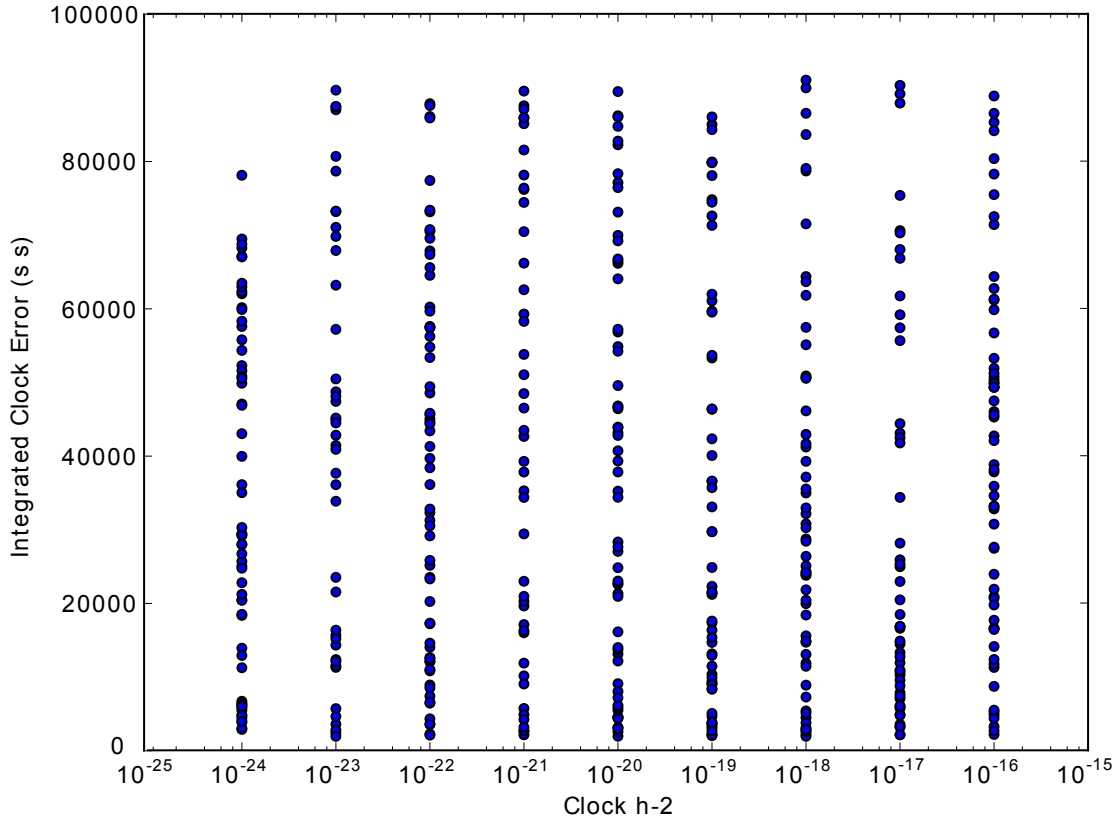
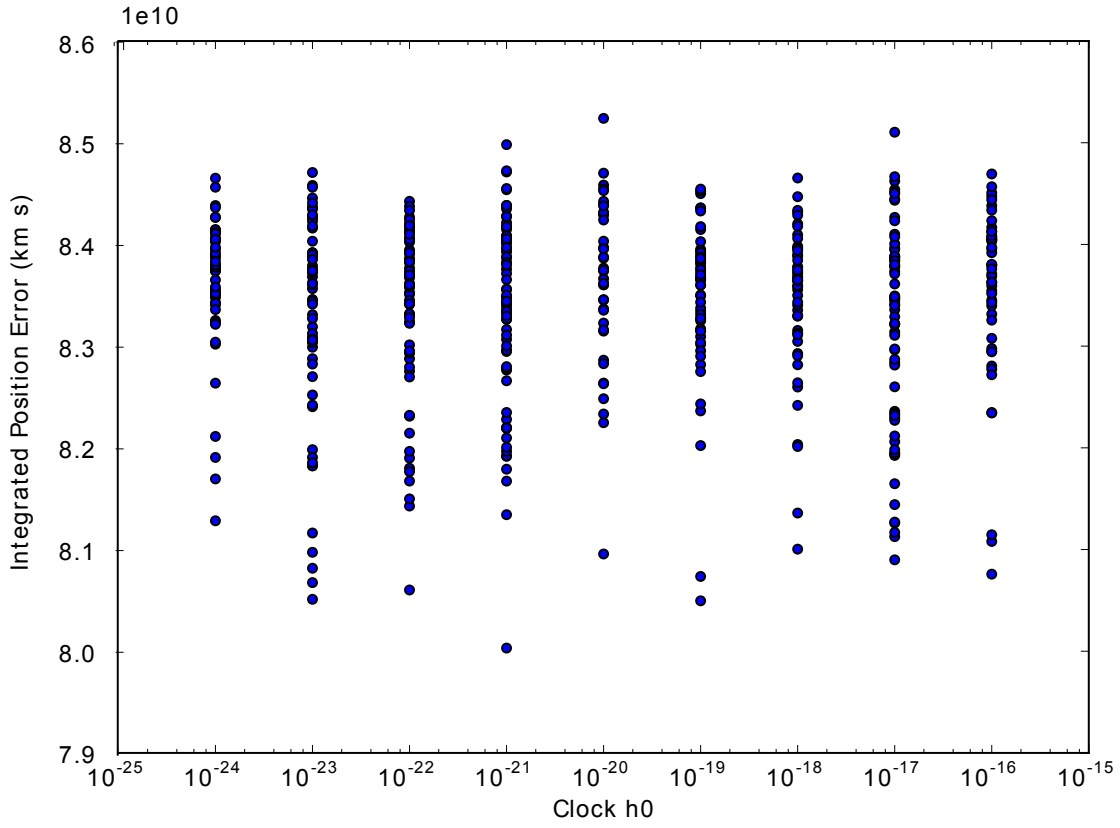


Figure 53: Comparison of Integrated Clock Error for Variable h_0 (top) and h_{-2} (bottom)



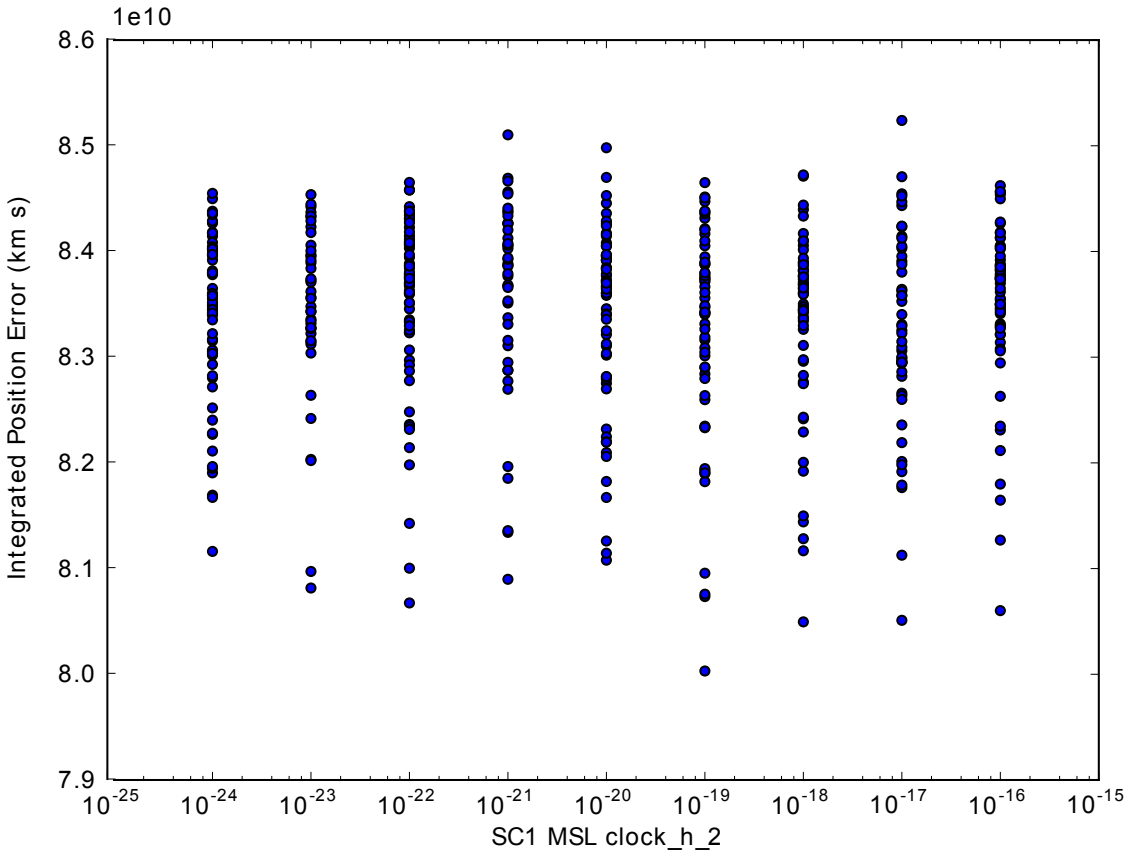
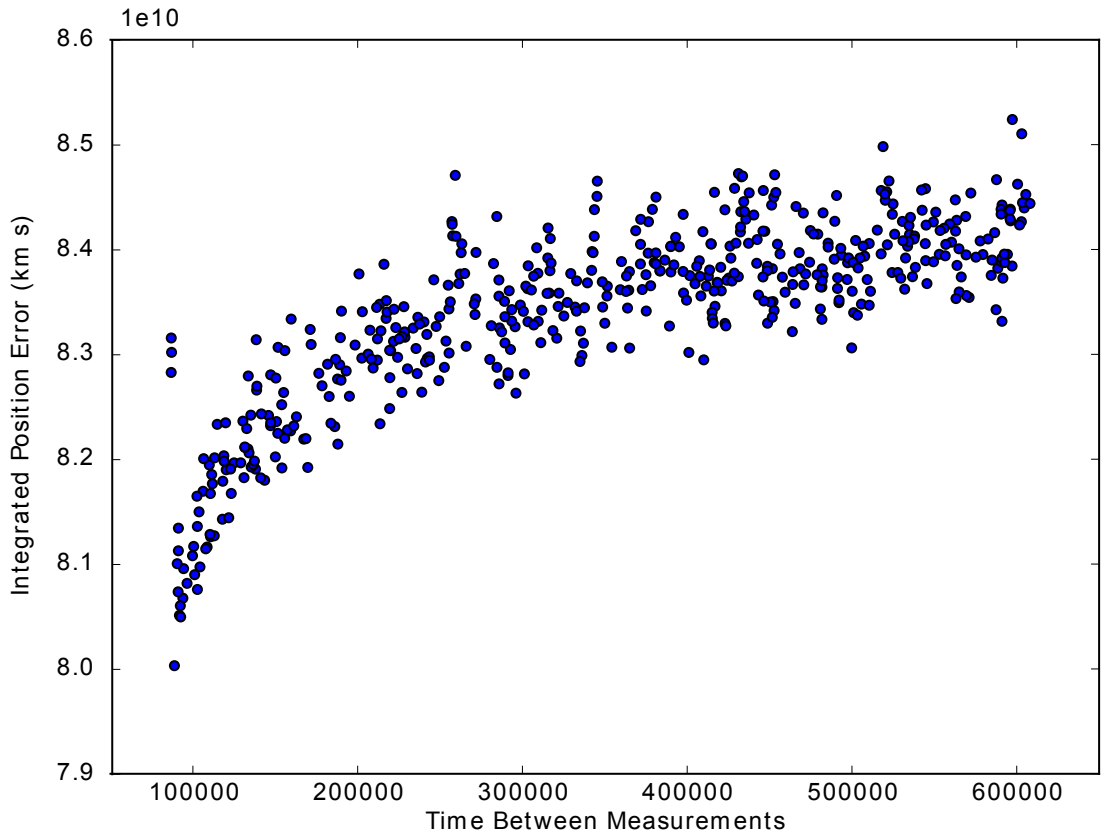


Figure 54: Comparison of Integrated Position Error for Variable h_0 (top) and h_{-2} (bottom)



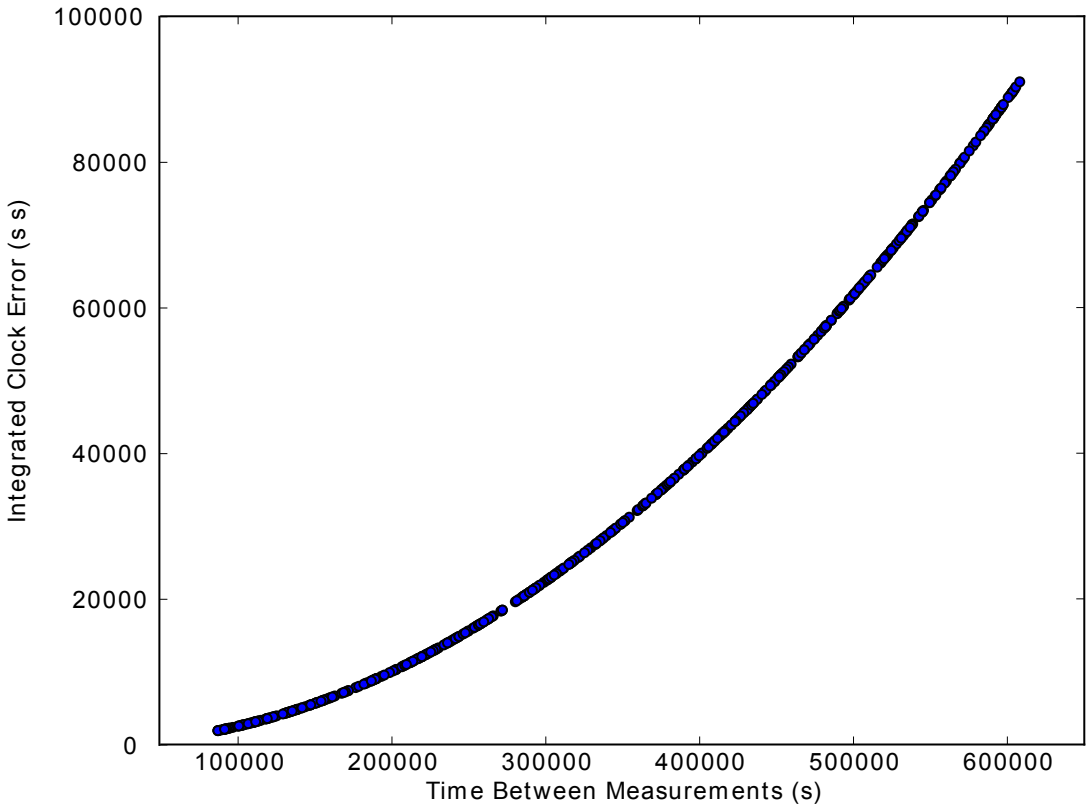


Figure 55: Effect of Time Between Clock Measurements on Integrated Position (top) and Velocity Errors (bottom)

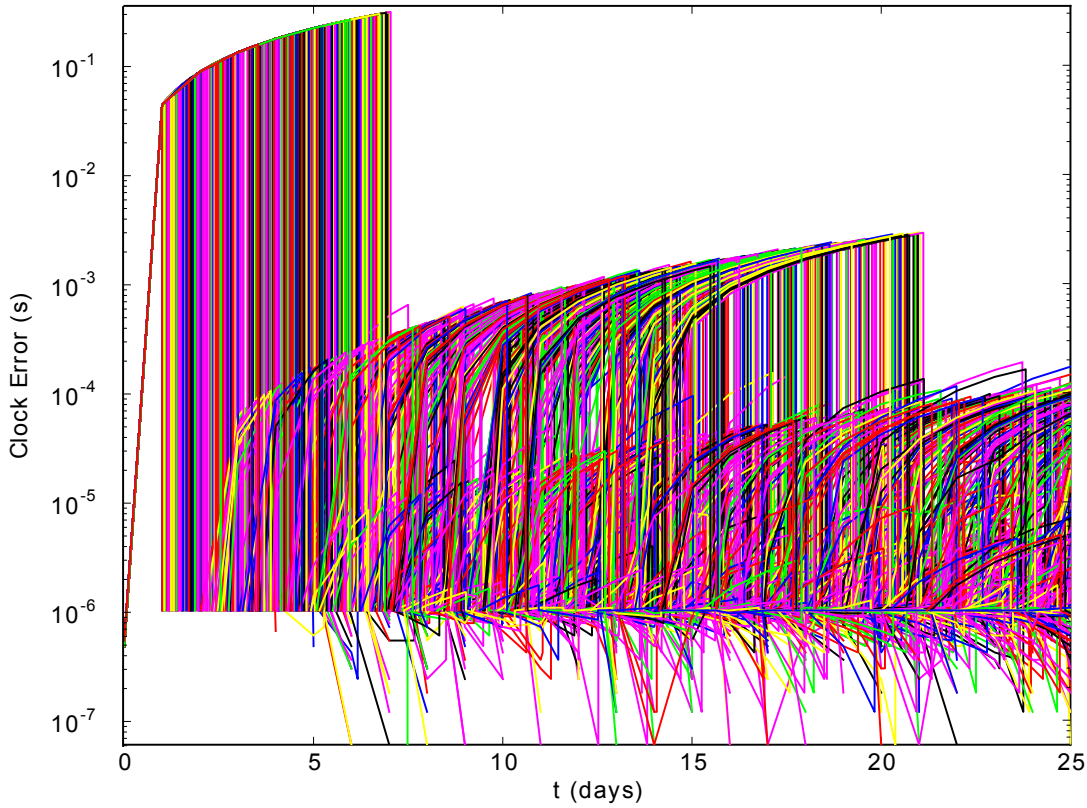


Figure 56: Dynamic Clock Error

6.4.4 Integrated Sensitivity of Measurements and Timing Uncertainty

In addition to timing measurements, it is important to understand the effect of the rate of state measurements on the estimation process (V6). The initial studies presented the effects of error of a fixed-time state update as well as the effect of measurement noise. This analysis relaxes that constraint, opening up the design space to determine the overall effect of measurement frequency on the estimation errors in terms of position, velocity, and clock accuracy. This analysis case is presented for a range of position measurement accuracies to additionally capture the capability of the estimator for a swath of frequencies and measurement properties. The analysis case is again given for the Martian cruise phase. The input variables and their ranges are identified in Table 19.

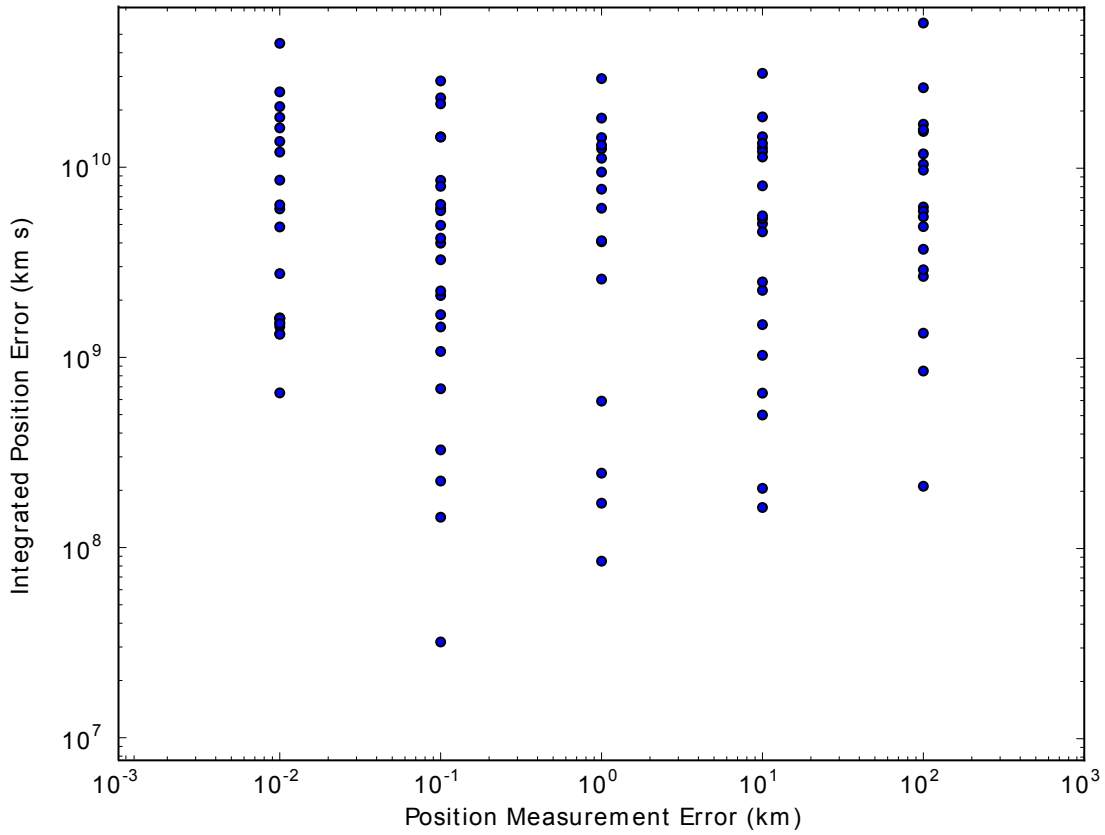
Table 19: Position Measurement Analysis Space

Variable	Min	Max	Units	Distribution
Position Measurement Error	1E-2	1E3	km	Log Uniform
Time Between Measurements	3600(1/24)	604800(14)	s(day)	Uniform

The results of this analysis follow similar trends to those already observed and reinforce the qualities of the filter, and the main performance drivers. Similar to previous cases, the filter is shown to achieve relatively stable state estimation error values independent of measurement error, as seen in Figure 57. This is largely due to the filter predicting the expected noise in the measurement. Also, like other analyses, the time between measurements is shown to be a key driver for the expected performance. This relationship is plotted in Figure 58. The visualization of the data clearly displays the strong correlation between integrated position and velocity error with time between measurements.

This behavior can be explained by two key factors. First, an increase in the amount of time between measurements increases the effect of propagation of initial errors, thus driving up the integrated error terms. The second factor is that as the time between measurements is decreased, the filter is also able to process more data, allowing for a better statistical exploration of the space, helping to reduce errors in the estimated state by the increased number of processed values.

The long term error levels, however, are strongly driven by these measurement errors. This is clearly showed in Figure 60, which displays the relationship between the final state errors with the position measurement error. Although the integrated error may be independent of these terms, the final error level is strongly driven by the error of the processed state updates. From this, it can be observed that with one single measurement (absolute position), the filter can only estimate its state on the order of the observed data. The state estimator is able to achieve and maintain high levels of velocity accuracy in order to maintain the position state estimates.



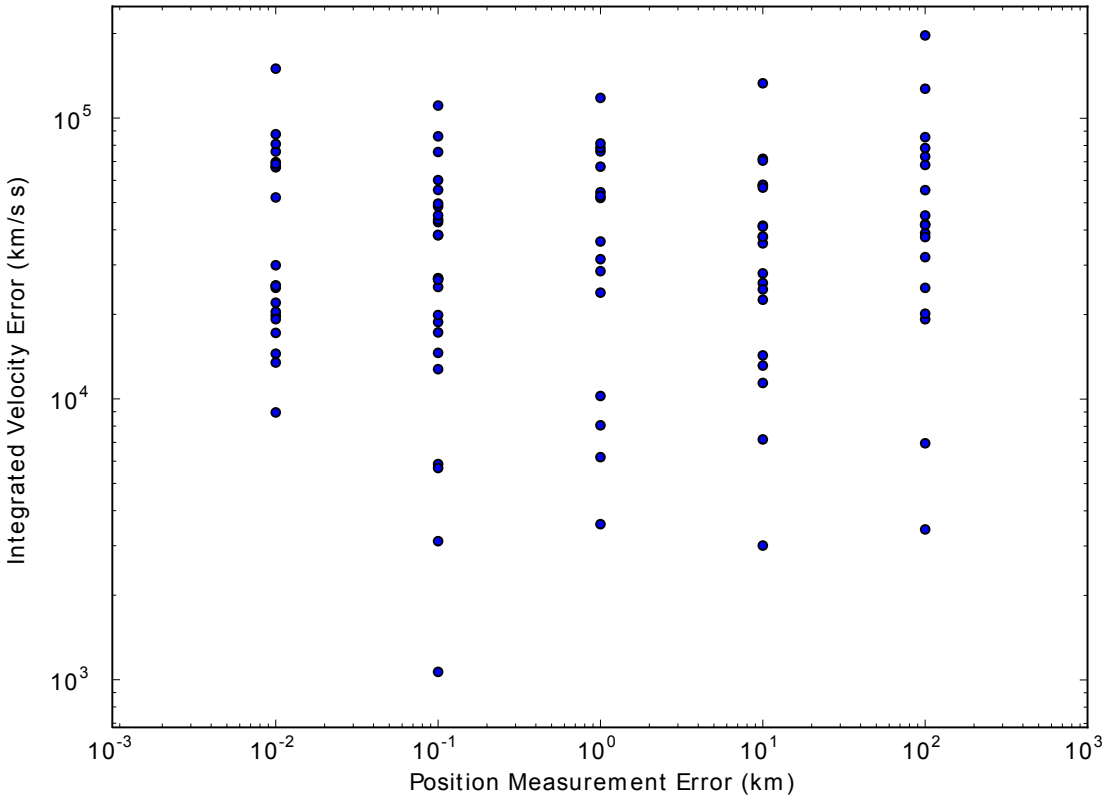
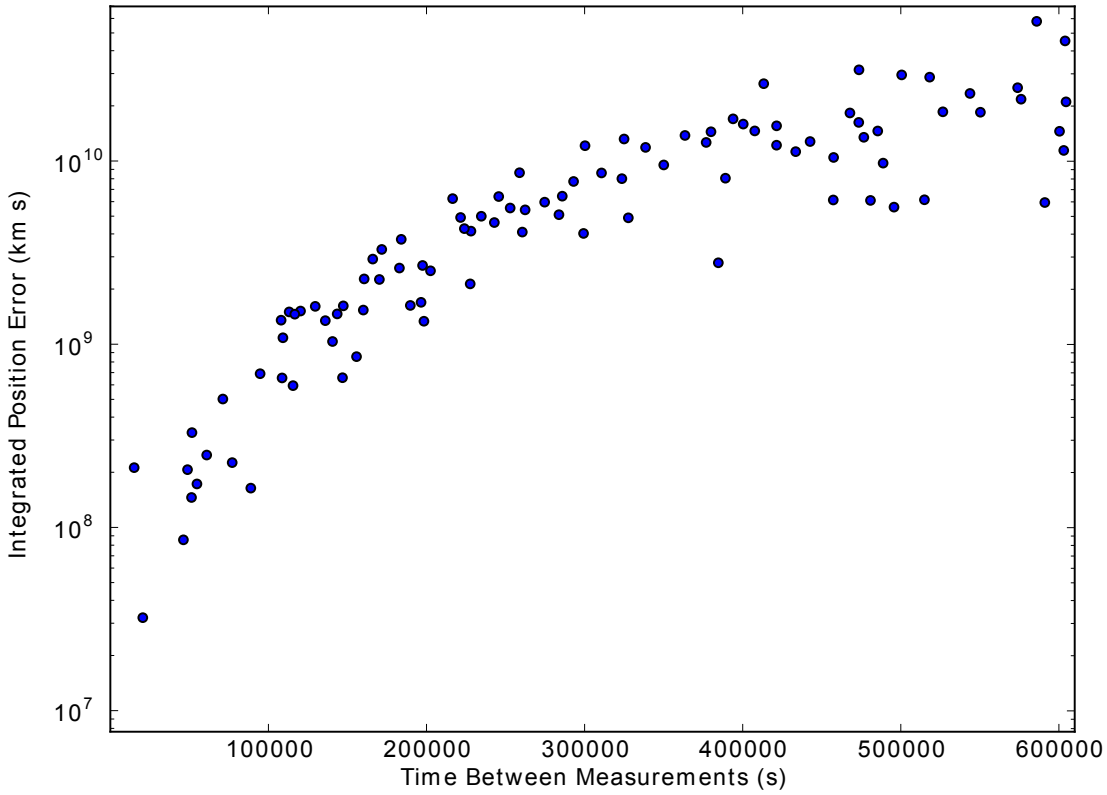


Figure 57: Comparison of Integrated Errors Versus Position Measurement Error



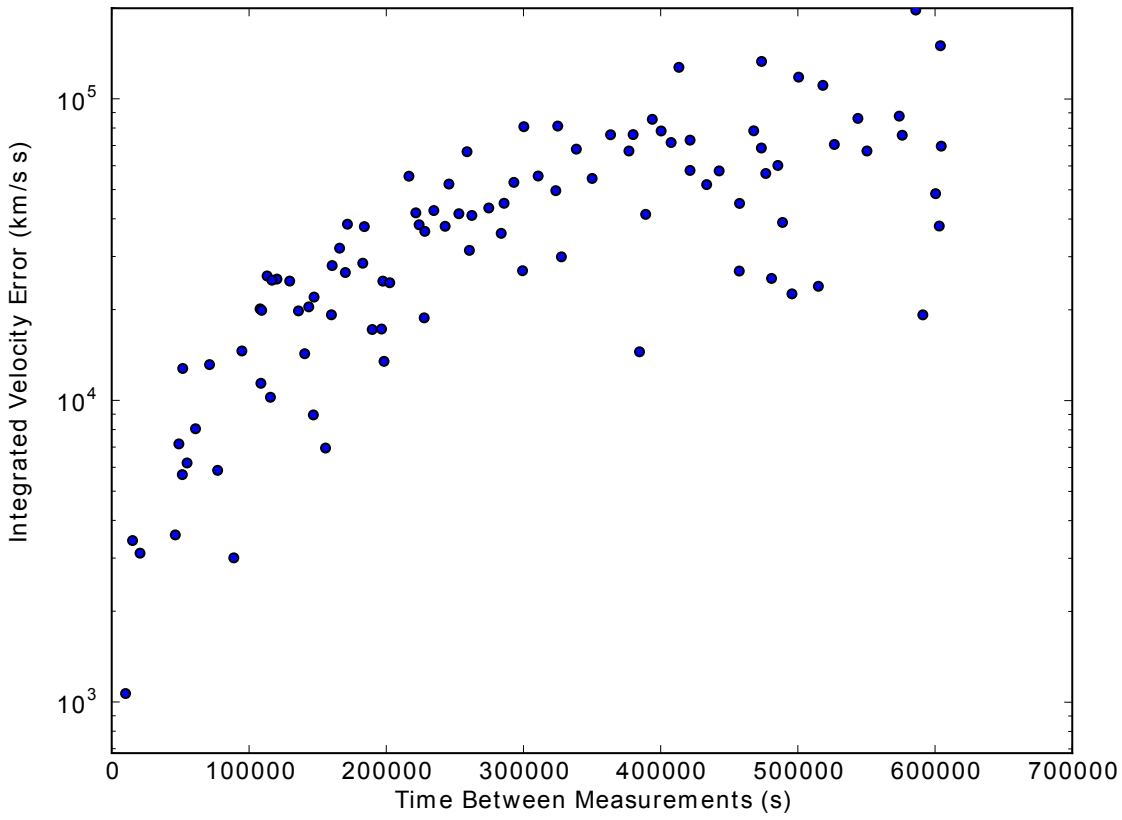


Figure 58: Comparison of Integrated Errors Versus Time Between Measurements

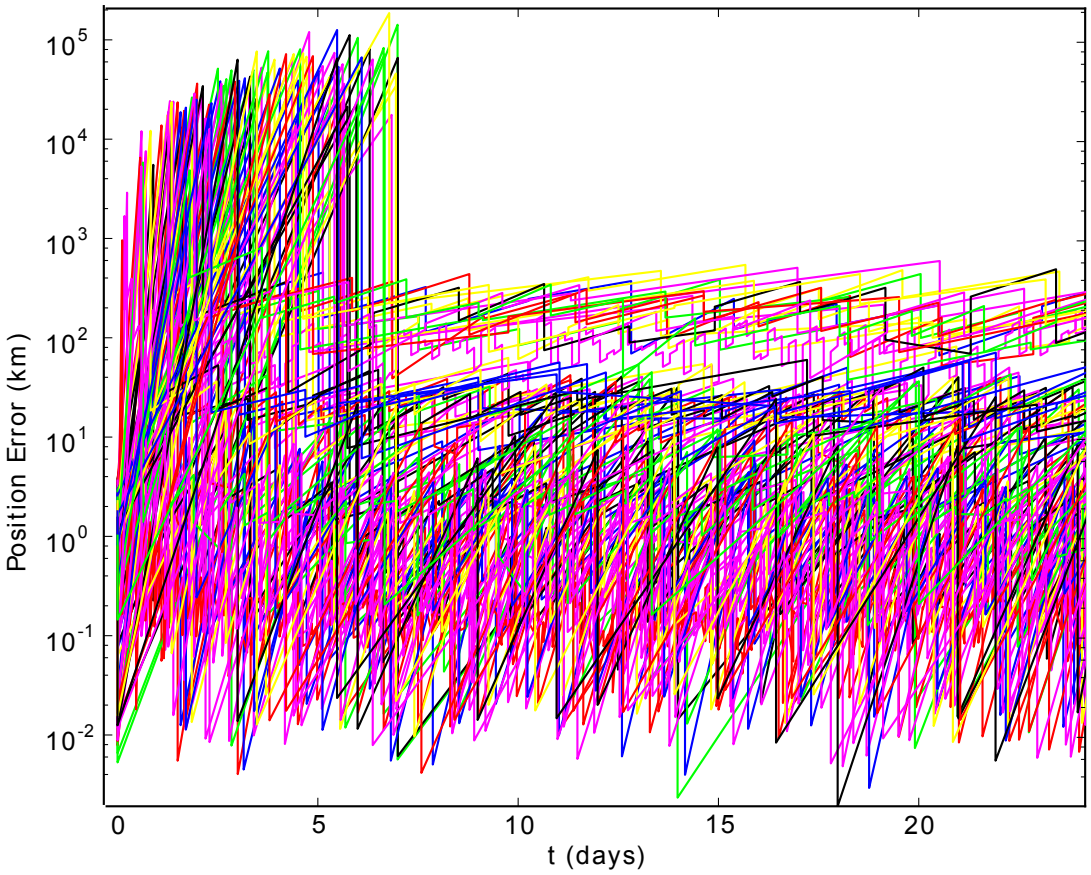


Figure 59: Position Error

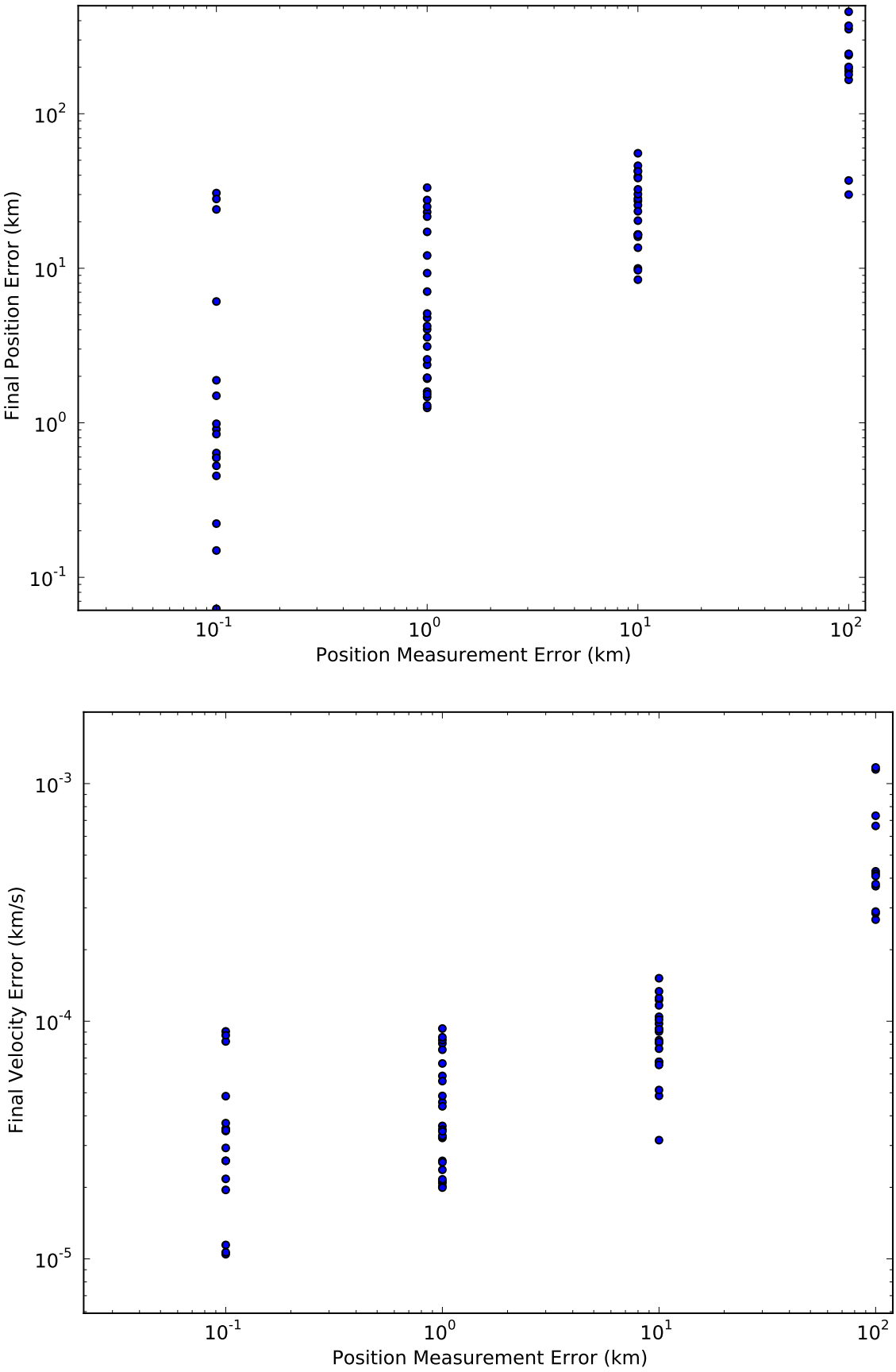


Figure 60: Comparison of Final Errors Versus Measurement Error

Table 20: Measurement Content Analysis Inputs

Variable	Value
Position Error	100 km
State Position Error	1 km
State Velocity Error	.1 km/s
Range to Earth Error	100 km

6.4.5 Measurement Content Comparison

The above studies have looked in particular at state updates. Another relevant comparison is the comparison of measurement content, focusing on the comparison of state versus position versus range. This study directly captures the flexibility and robustness of the filter to input variables. For this study, the frequency and the error of each measurement type are varied to capture the performance for comparison. This allows for a demonstration of the capability of the state estimator to the amount of content in a measurement update. The inputs for each trade are described in Table 20. For these cases, several analysis were performed. For position (V7) and range measurements (V8), it was assumed that the observation batches consisted of 10 packets separated by 60 seconds. A design trajectory was run for measurement intervals of one day and one week. The performance of state measurements (V9) was only run at one week intervals to capture current standards.

Plots of the estimated position error are given in Figures 61 and 62. The first plot shows a comparison of the predicted position errors for the various measurement contents and frequencies. From this analysis, the benefit of the state update is clearly beneficial, though the long-term performance of position measurements is within 100 kilometers, with similar errors post-update. The range observations alone do not show strong performance in correcting the vehicle's estimated state. Though with daily updates, it is shown that the error does begin to decrease.

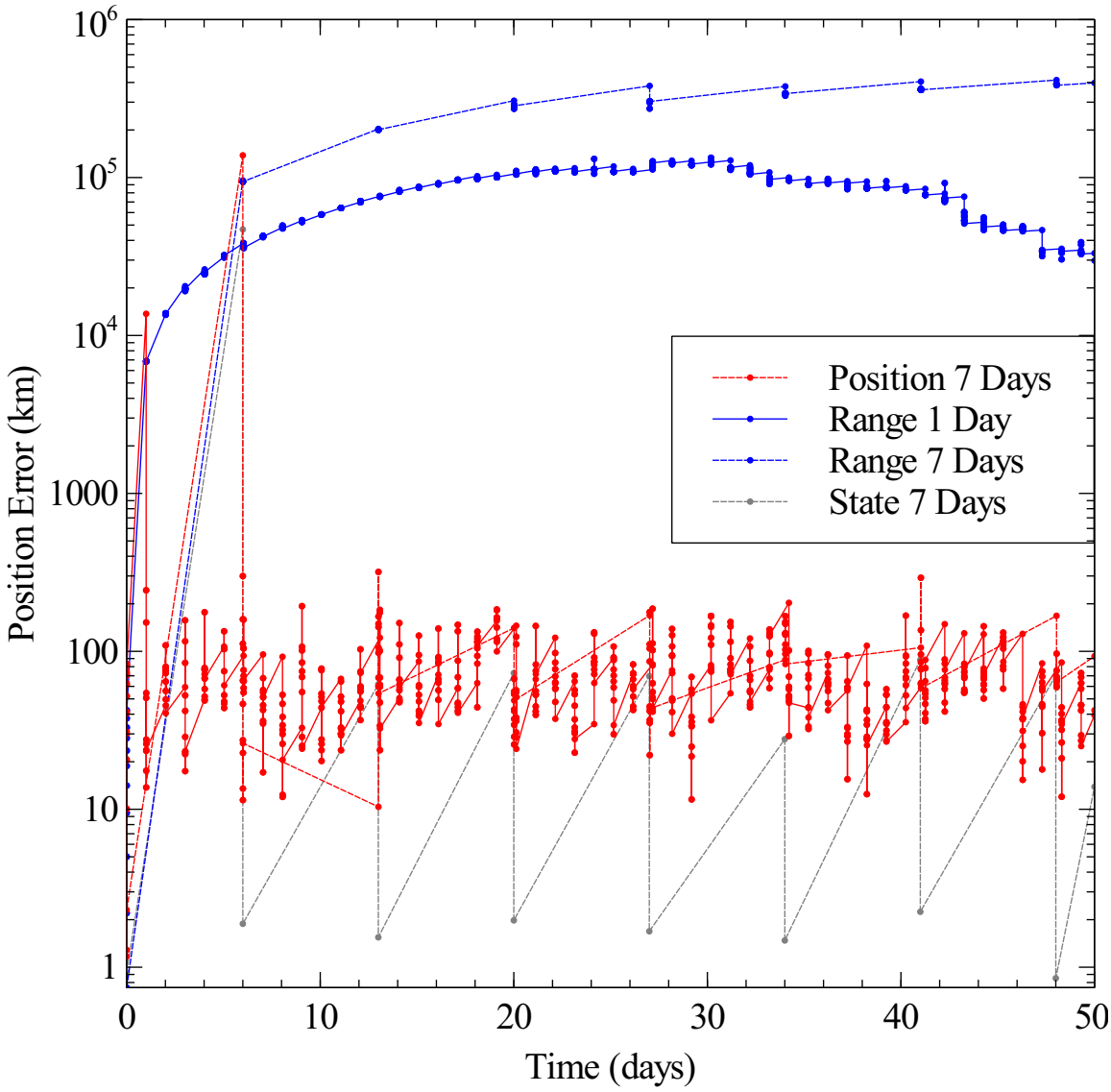


Figure 61: Position Error as Function of Measurement Content

Figure 62 gives a detailed view comparing state and position updates. From this analysis, it is shown that the daily position measurements does bound error better than weekly updates. This follows previous trends described above. The per-measurement value also shows a clear reduction in measurement error over the course of the process. Also of note is that even though the measurement noise is 100 kilometers, the estimator is still able to predict performance to half of that level due to its data processing algorithms. Lastly, the state update shows its strong performance, enabling very high accuracy state updates. This

is due to the complete state observation and the expected reduced noise capability.

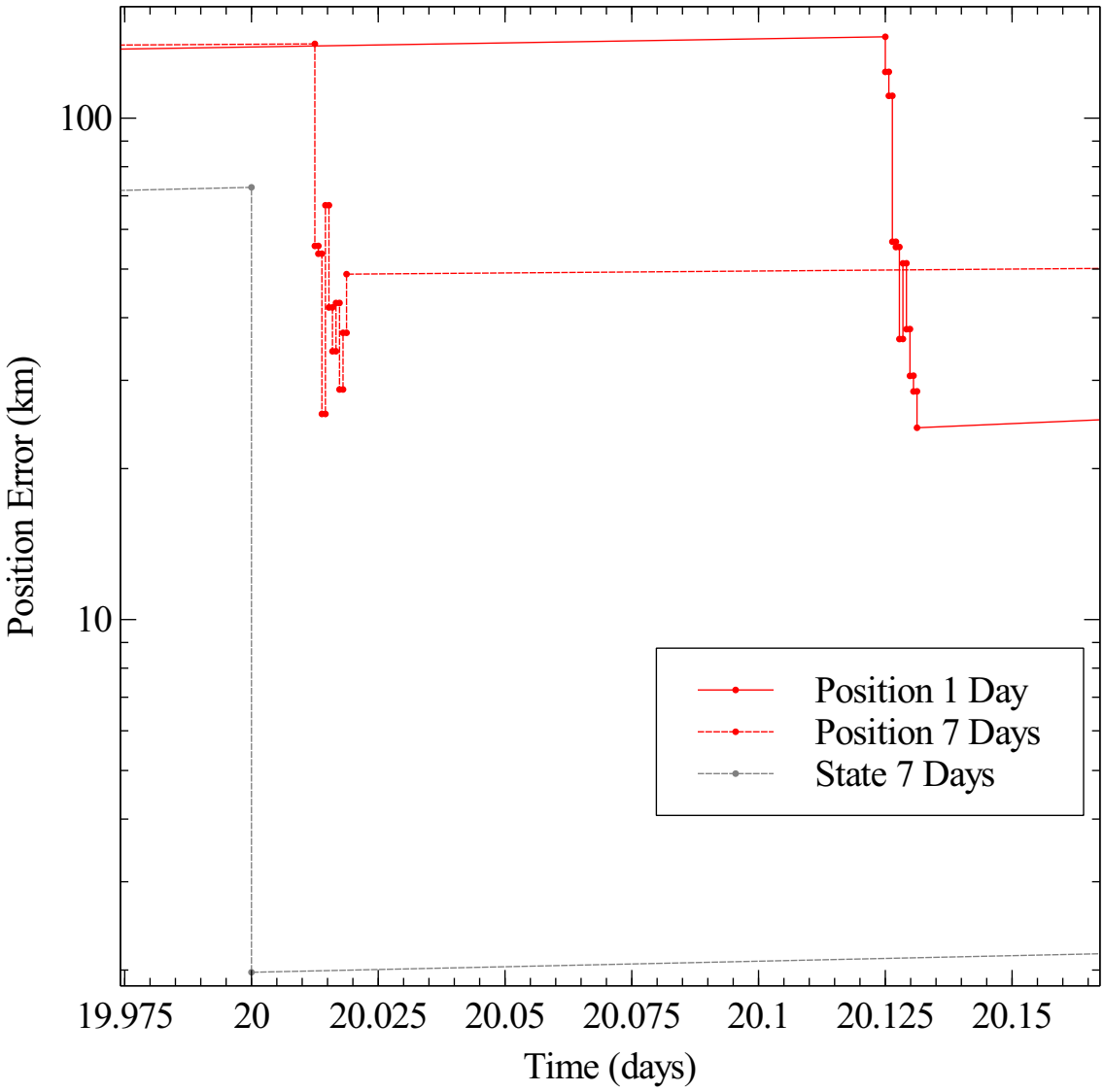


Figure 62: Detailed Comparison of State versus Position Updates

6.4.6 Overall Sensitivity Analysis

Even though the above trades provide insight into the navigation architecture as a whole, they do not allow for the integrated analysis of these variables and their interactions. Therefore, a final overall design case (V10) is proposed to demonstrate the capability of the integrated framework to capture complex variable interactions. For this study, a larger subset of variables will be relaxed to identify key drivers of performance for integrated position,

Table 21: Position Measurement Delay Analysis Space

Variable	Min	Max	Units	Distribution
Time to State Update	86400(1)	604800(7)	s(day)	Uniform
Time Between State Updates	86400(1)	604800(7)	s(day)	Uniform
Position Measurement Error	1E-3	1E4	km	Logarithmic
Time to Position Update	86400(1)	604800(7)	s(day)	Uniform
Time Between Position Updates	86400(1)	604800(7)	s(day)	Uniform
Time to Clock Update	86400(1)	604800(7)	s(day)	Uniform
Time Between Clock Updates	86400(1)	604800(7)	s(day)	Uniform

velocity, and timing error measures. The performance will be captured by running a Monte Carlo Analysis over all of these variables, a sufficiently large number of cases will be used to enable capture of general trends. Additionally, statistical tools will be used to identify correlations between variables. This calculation of covariance will be used to identify key performance drivers and verify general design trends. The full list of design variables is shown in Table 21.

Statistical tools can be used to calculate the correlations between the various variables considered in the analysis. For this analysis, the MATLAB software package was used to perform the calculations. The correlation coefficient, or normalized covariance, was used to identify relationships between pairs of variables.

For the input and output variables the inputs and outputs are collected in addition to the p values. These values represent the statistical likelihood that a statistical property can be calculated randomly. For example, low values of p correspond to statistically significant relationships. For this analysis, a p value of .05 (or a five percent chance of the relationship being random) was used to identify the variables. The calculated covariance and p-values are given in Figures 63 and 64.

	State DT First	State DT Meas.	Position Meas. Error	Position Meas. DT First	Position Meas. DT	Time Meas. DT First	Time Meas. DT	Final Position Error	Final Velocity Error	Final Clock Error	Integrated Pos. Error	Integrated Vel. Error	Integrated Clock Error
State DT First	1	0.0016	0.107114	0.059989	0.0248	0.00373	0.1171	0.13364	0.10282	0.079	0.5193527	0.5074254	0.018534
State DT Meas.	0.0016	1	-0.04515	-0.07865	-0.0815	0.03251	0.0118	0.1741	0.15573	-0.04	-0.016543	-0.027747	0.0386603
Position Meas. Error	0.1071	-0.0451	1	0.030074	-0.0029	-0.0073	-0.058	0.46866	0.51416	-0.105	0.0807388	0.0567203	0.0422441
Position Meas. DT First	0.06	-0.0787	0.030074	1	-0.0692	-0.0807	0.0169	0.08374	0.11481	-0.078	0.5125931	0.5131383	0.0644159
Position Meas. DT	0.0248	-0.0815	-0.0029	-0.06922	1	-0.0114	-0.023	-0.04248	-0.0803	0.079	-0.009693	-0.008231	-0.0396361
Time Meas. DT First	0.0037	0.0325	-0.00726	-0.08073	-0.0114	1	-0.032	-0.06036	-0.0515	0.026	-0.053624	-0.043216	-0.1035722
Time Meas. DT	0.1171	0.0118	-0.0576	0.016906	-0.0227	-0.0321	1	0.047	0.03842	0.062	0.0459332	0.0532557	0.1109569
Final Position Error	0.1336	0.1741	0.468655	0.083738	-0.0425	-0.0604	0.047	1	0.8592	-0.115	0.1701363	0.1507038	0.0679824
Final Velocity Error	0.1028	0.1557	0.514163	0.114809	-0.0803	-0.0515	0.0384	0.8592	1	-0.127	0.1485271	0.1306149	0.0477026
Final Clock Error	0.0786	-0.0402	-0.10473	-0.07762	0.07874	0.02621	0.0616	-0.11533	-0.1269	1	-0.060985	-0.068567	0.0865629
Integrated Pos. Error	0.5194	-0.0165	0.080739	0.512593	-0.0097	-0.0536	0.0459	0.17014	0.14853	-0.061	1	0.9518043	0.0072229
Integrated Vel. Error	0.5074	-0.0277	0.05672	0.513138	-0.0082	-0.0432	0.0533	0.1507	0.13061	-0.069	0.9518043	1	-0.0091645
Integrated Clock Error	0.0185	0.0387	0.042244	0.064416	-0.0396	-0.1036	0.111	0.06798	0.0477	0.087	0.0072229	-0.009165	1

Figure 63: Correlation Coefficient Values

	State DT First	State DT Meas.	Position Meas. Error	Position Meas. DT First	Position Meas. DT	Time Meas. DT First	Time Meas. DT	Final Position Error	Final Velocity Error	Final Clock Error	Integrated Pos. Error	Integrated Vel. Error	Integrated Clock Error
State DT First	1.000	0.972	0.017	0.180	0.580	0.934	0.009	**0.003**	**0.021**	0.079	**0.000**	**0.000**	0.679
State DT Meas.	0.972	1.000	0.314	0.079	0.069	0.468	0.792	**0.000**	**0.000**	0.370	0.712	0.536	0.388
Position Meas. Error	0.017	0.314	1.000	0.502	0.949	0.871	0.198	**0.000**	**0.000**	**0.019**	0.071	0.205	0.346
Position Meas. DT First	0.180	0.079	0.502	1.000	0.122	0.071	0.706	0.061	**0.010**	0.083	**0.000**	**0.000**	0.150
Position Meas. DT	0.580	0.069	0.949	0.122	1.000	0.799	0.612	0.343	0.073	0.079	0.829	0.854	0.376
Time Meas. DT First	0.934	0.468	0.871	0.071	0.799	1.000	0.474	0.178	0.250	0.559	0.231	0.335	**0.021**
Time Meas. DT	0.009	0.792	0.198	0.706	0.612	0.474	1.000	0.294	0.391	0.169	0.305	0.235	**0.013**
Final Position Error	0.003	0.000	0.000	0.061	0.343	0.178	0.294	1.000	0.000	0.010	0.000	0.001	0.129
Final Velocity Error	0.021	0.000	0.000	0.010	0.073	0.250	0.391	0.000	1.000	0.004	0.001	0.003	0.287
Final Clock Error	0.079	0.370	0.019	0.083	0.079	0.559	0.169	0.010	0.004	1.000	0.173	0.126	0.053
Integrated Pos. Error	0.000	0.712	0.071	0.000	0.829	0.231	0.305	0.000	0.001	0.173	1.000	0.000	0.872
Integrated Vel. Error	0.000	0.536	0.205	0.000	0.854	0.335	0.235	0.001	0.003	0.126	0.000	1.000	0.838
Integrated Clock Error	0.679	0.388	0.346	0.150	0.376	0.021	0.013	0.129	0.287	0.053	0.872	0.838	1.000

Figure 64: Probability Coefficient Values

From these numerical values, it is straightforward to identify correlated variables and identify key relationships. For the final position error, the main terms are from the state measurement parameters, particularly the time to the first measurement and time between measurements. Additionally the error of the position update has a strong effect. This relationship can be explained by the state estimate being a complete position and velocity measurement providing the maximum information to the state estimator. Additionally, the errors in the position measurement can induce errors into the estimated state.

The velocity errors, though, show correlation with the parameters of the state and position measurement, except for the time delta value. This behavior can be explained by the high dependency on the errors of the velocity measurement in the state update, which

provides the only direct observation of velocity. The correlation with the time to the first position measurement and error is due to the loss of accuracy of position compared to state updates. As such, an early position measurement may induce error in the velocity state due to the high initial value of the state's covariance. These early measurements can affect the state estimator's settling time to its stable error state.

The integrated errors, both of velocity and position are strongly dependent on the time to the first measurement. This correlation is observed due to the effect of increasing the initial state errors over a longer period of time. As the time to first measurement increases, the integrated state error likewise increases, especially due to the large amount of integrated error built up early in the mission.

The main factors affecting the integrated clock error are the time between and the time to first measurements. This is demonstrative of the estimators algorithms, in that the clock states are only directly being measured through this update. The other updates to the estimated timing values are due to the propagated covariance and predictions of the oscillator instability. As described above, the majority of integrated error is shown to also occur early in the timeline here, seen by the large dependence on time to first measurement.

6.5 Measurement Optimization

The above results demonstrate the functionality of the design space analysis tools built into the implemented simulation framework. The experiments performed provide insight into the design space and allow for enhanced understanding of the navigation capabilities in terms of specific measurement functionality. In addition to these test cases, the ability to perform system optimization with integrated tools must be demonstrated. This is being done to complete the validation of the implemented framework. Two separate analysis cases will be exercised, to demonstrate a basic confirmation of the optimizer's functionality and to provide an optimal measurement operational concept.

6.5.1 Optimization of Position Measurement

The first optimization case (V11) provides a straightforward analysis of the design space to insure the proper operation of the implemented tool. In this case, the optimizer is allowed to

Table 22: Position Measurement Optimizer Inputs

Variable	Min	Max	Units	Distribution
Position Measurement Error	10	10000	km	Uniform
Time Between Measurement Batches	86400(1)	604800(7)	s(day)	Uniform
Time Between Individual Measurements	1	3600	s	Uniform
Measurements in a Batch	5	100		Uniform

Table 23: Position Measurement Optimizer Outputs

Variable	Units
Final Position Error	km
Final Velocity Error	km/s
Final Clock Error	s
Integrated Position Error	km s
Integrated Velocity Error	km/s s
Integrated Clock Error	s s

vary measurement noise and frequency in order to meet the performance goals of minimized state estimation error. For this scenario, it is assumed that the spacecraft receives fixed state updates at weekly intervals. To perform this analysis the tool utilizes its interface to the genetic algorithm library. The input ranges of the variables, and output performance variables are given in Table 22 and 23. The results of this analysis should also be derivable from the previous design analysis cases.

The optimizer was run twice (V11 and V12) with equally weighted goals of minimizing each design variable. Each utilized ten generations of forty individuals. The values of the inputs are plotted in Figures 65, 66, 67, and 68. These show the average value of each set over the course of the optimization. The errors bars on the charts track one standard deviation of the input for all members of that generation. Distribution of a population's values is observable from the error bars. For an optimization run that is converging to an optimal solution, these bars should be minimized, indicating that an optimal configuration has been obtained. For this solution, all design variables show this convergent behavior. The two runs also allow observation of the repeatability of the simulation.

For the input values capturing the batch properties, time between and number of measurements in a batch, these are seen to find a common solution minimizing the time between

measurements and maximizing the number of measurements. Additionally, both cases seek to minimize the error in the position measurement, which would be expected in order to minimize the final navigation state errors. The only discrepancy between the two optimizations is in regards to the time between measurements. The results show two distinct solution approaches.

Due to the frequency of the measurements, the second optimization down-selected to a timing situation where the measurements were linked such that the state updated very near the end time of the simulation. This allows for low position and velocity errors, though at the cost of the integrated error terms. This also shows the potential trouble with optimizing for final state error when more interested in bulk performance of the navigation state estimation. Additional constraints on the analysis case or different weightings can reduce the likelihood of this scenario occurring.

The first example provides a more expected result. This optimization case converged to a solution with the minimum time between measurements. Combined with the minimum time between measurements in a batch and the maximum number of observations in each batch, this solution provides for frequent large batches of error measurements. This large number of near-simultaneous observations allow for improved statistics in regards to getting an average update, and reducing the effect of measurement noise.

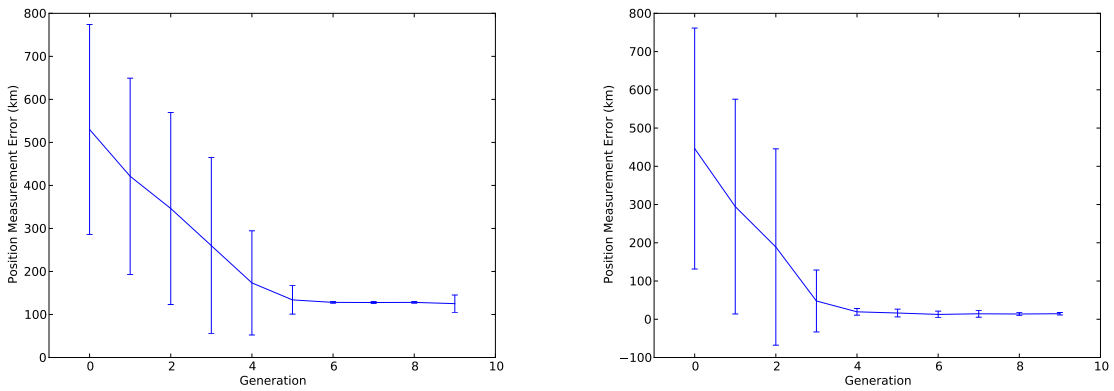


Figure 65: Measurement Position Error over Optimization

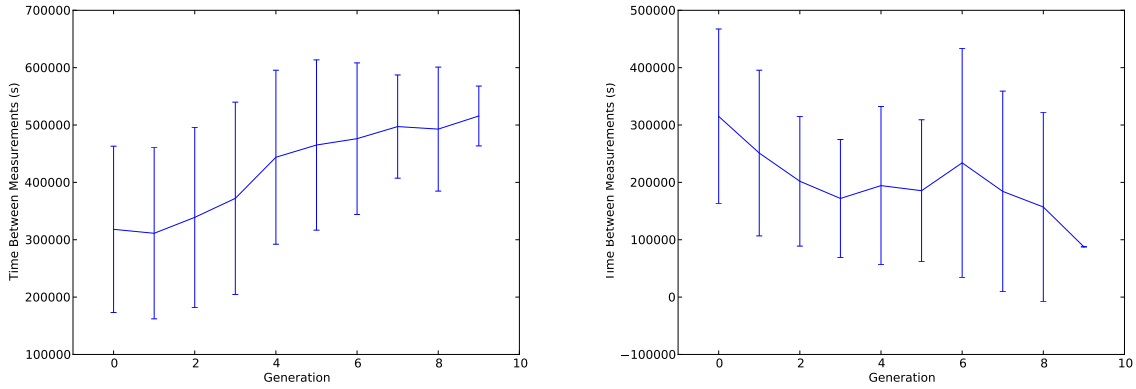


Figure 66: Time between Measurements over Optimization

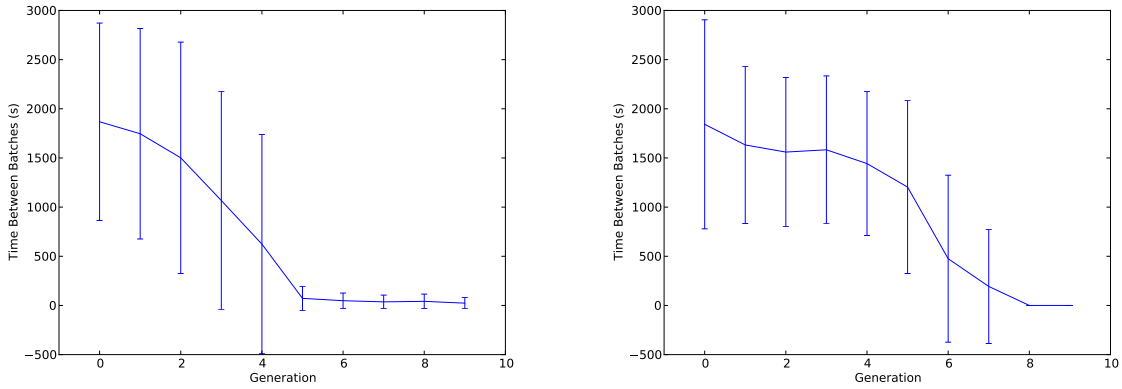


Figure 67: Time between Batches over Optimization

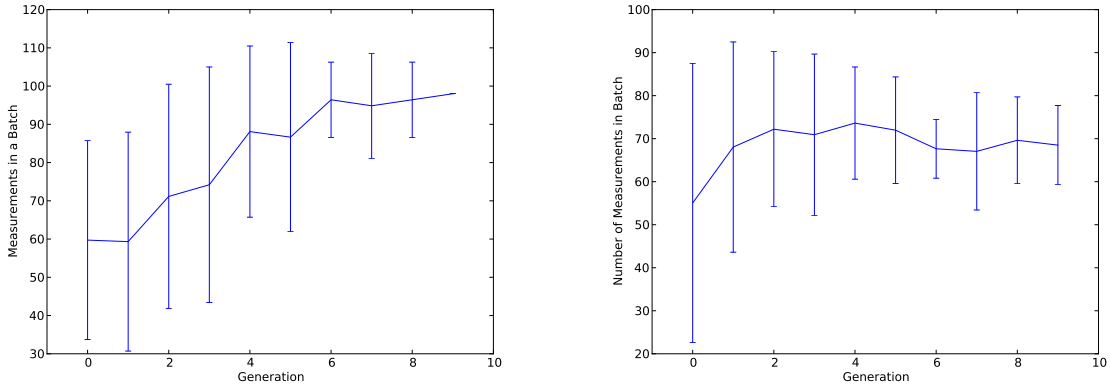


Figure 68: Number in a Batch over Optimization

Figures 69, 70, and 71 show the dynamics of the state estimation errors for the final generation in the optimizer. For ideal operation of the optimization process, each member of the population should exhibit very similar error dynamics over the course of the simulation. As seen in the plots, there is not much deviation between the state estimation performance of the two final populations. Both cases perform equally well in terms of position and velocity errors. Figure 71 shows an interesting result, that is expected from the previous analysis. Without any clock updates, the clock error continues to increase over time. This shows the importance of the timing update.

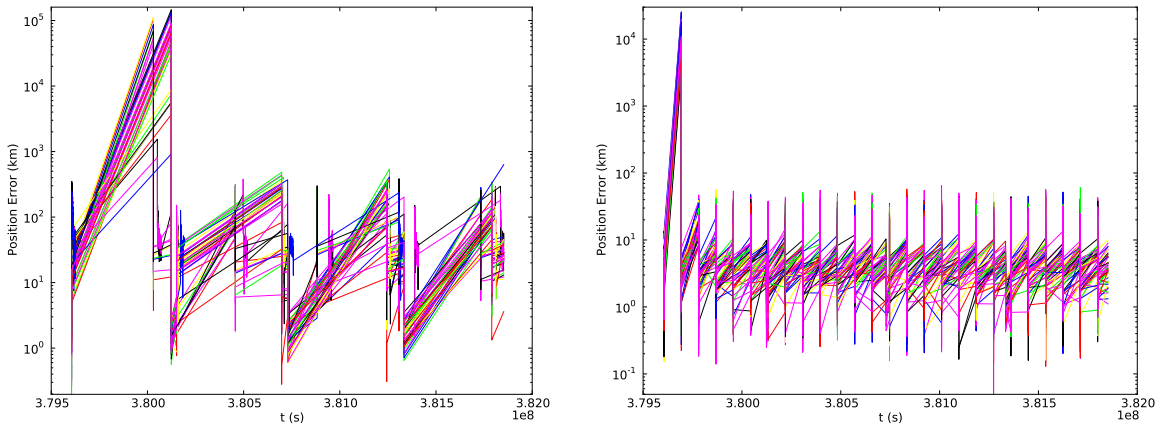


Figure 69: Position Error in Last Generation

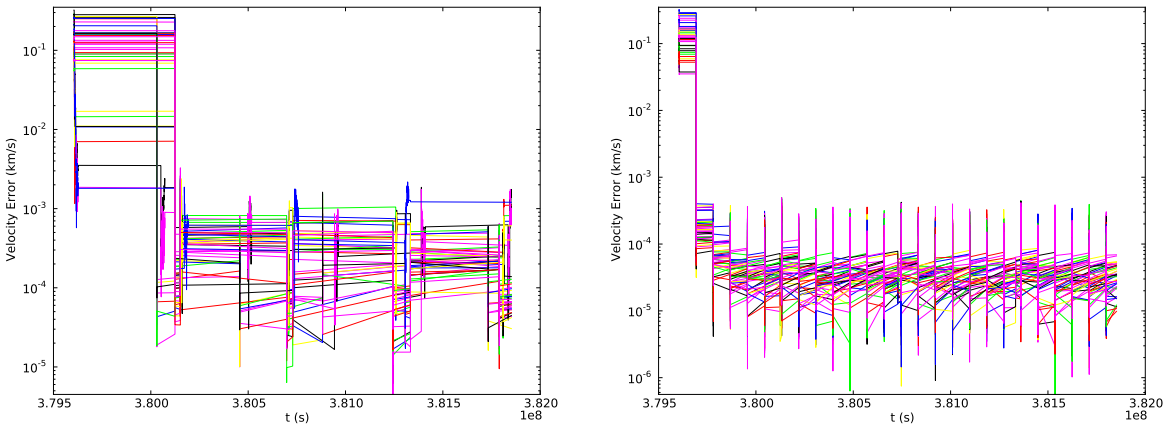


Figure 70: Velocity Error in Last Generation

Table 24: Optimizer Results Comparison

Variable	Opt. 1	Opt. 2	With Time Measurement
Position Measurement Error(km)	125	15	10
Time Between Measurement Batches (s)	520000	90000	150000
Time Between Individual Measurements (s)	20	1	1
Measurements in a Batch	100	70	80

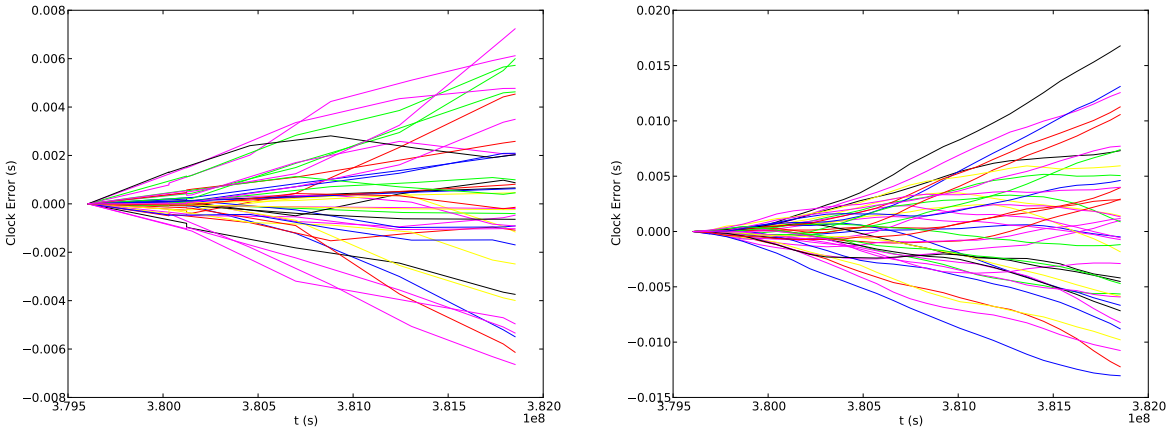


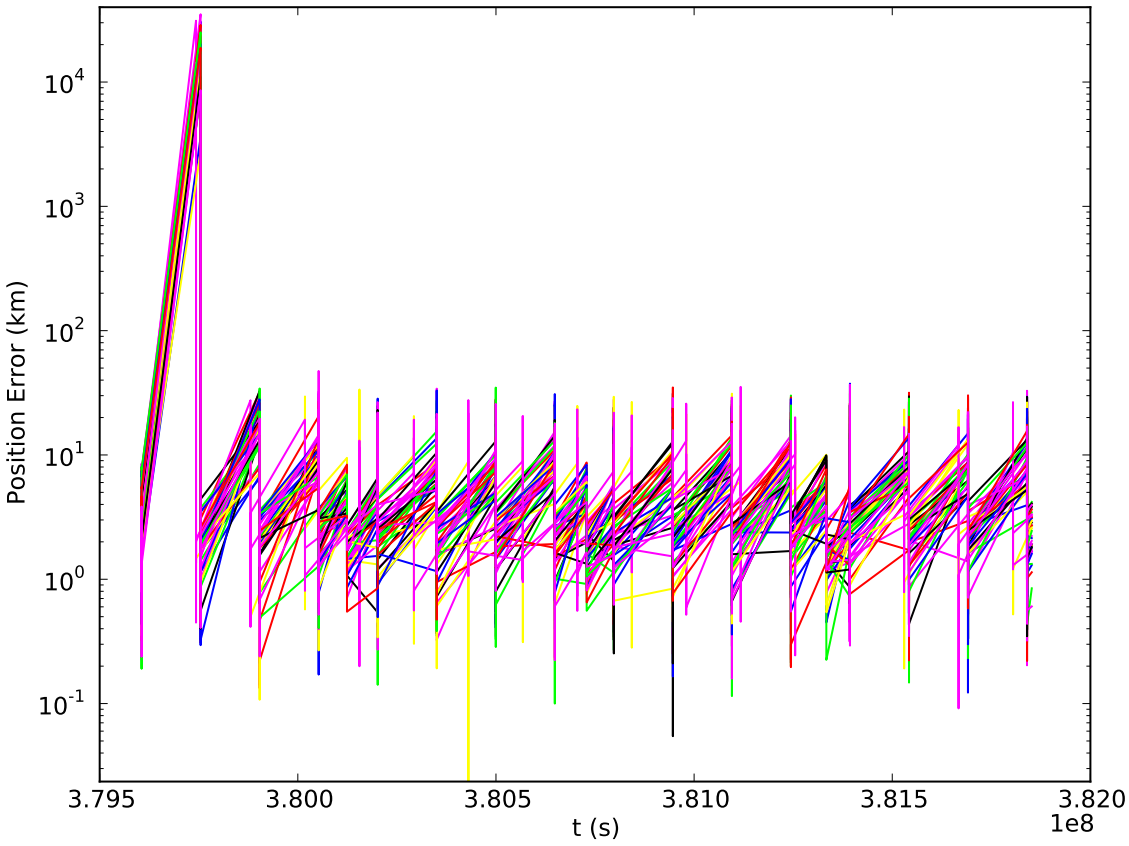
Figure 71: Clock Error in Last Generation

6.5.2 Effect of Timing Measurements on Optimal Parameters

From the large observed effect of timing measurements on the clock estimation error, an additional optimization run was performed (V13). The inputs and outputs are the same as listed in Tables 22 and 23. The substantial difference for this analysis case is the inclusion of weekly timing updates to occur with the same frequency as the state updates (at weekly intervals). The optimization was re-run for this analysis with the resulting values given in comparison to the initial two runs given in Table 24.

For this analysis case, the optimizer minimized the time between measurements and measurement error while increasing the number of measurements in a batch. The main difference is the time frequency of updates. For this case, with the added time measurements, the state estimator is able to obtain an improved estimate and therefore able to propagate its state for a longer interval between batches. This shows the strong benefit of including a

timing measurement into the navigation architecture. The resulting dynamics of the error states is given in Figure 72 and 73.



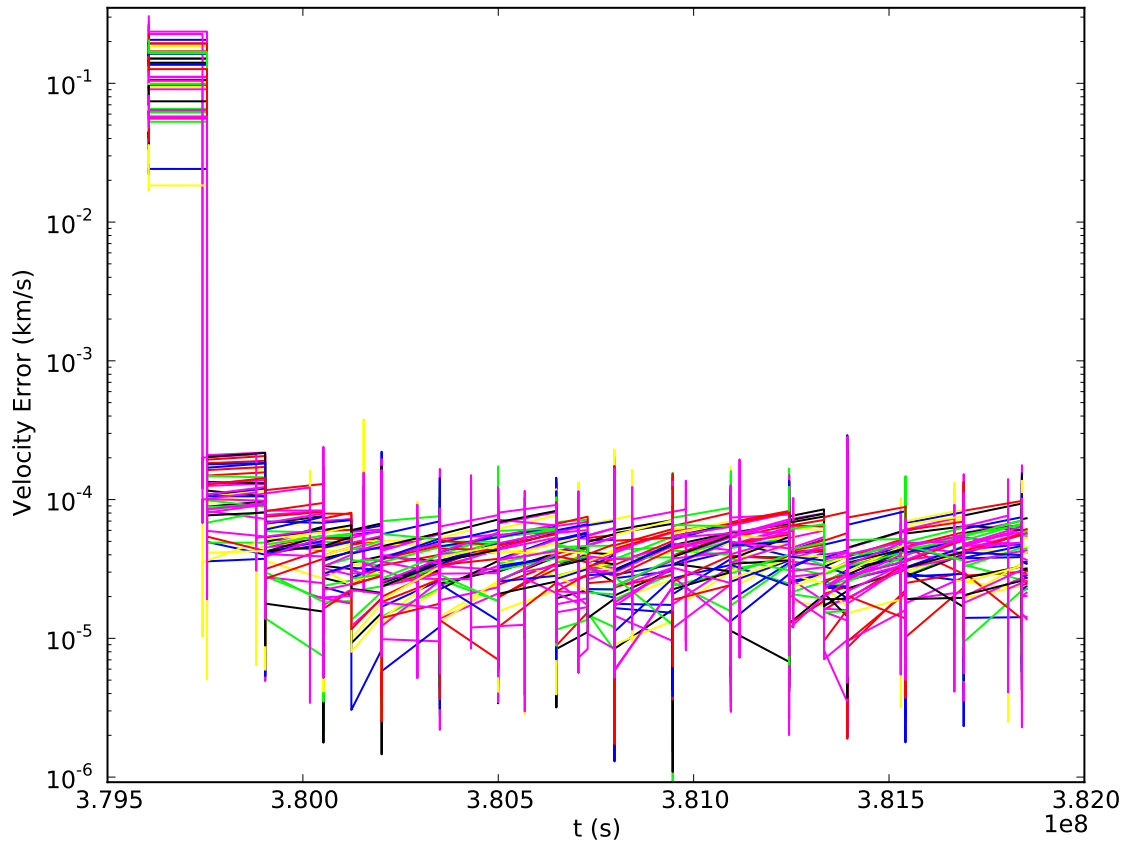


Figure 72: Dynamics of Error State with Clock Measurement

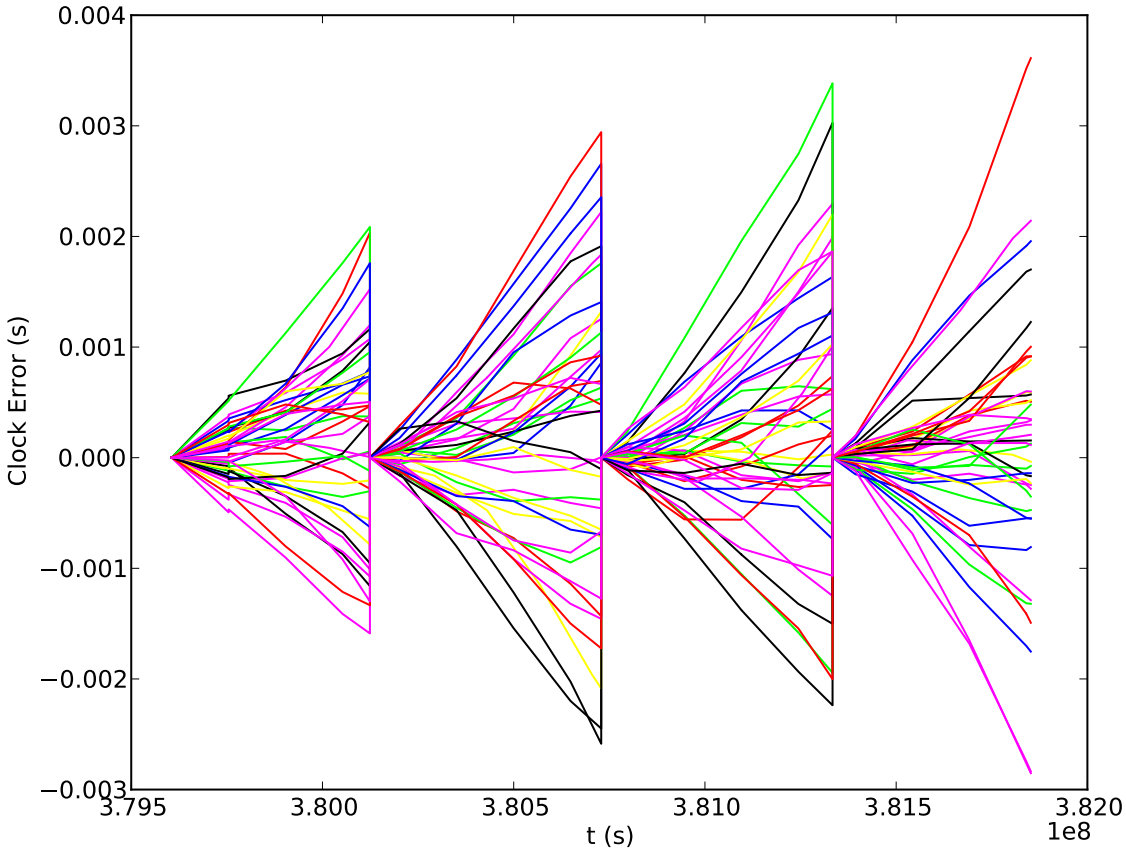


Figure 73: Dynamics of Clock Error State with Time Measurement

6.6 Summary of Verification and Validation

This chapter has presented a series of test cases to both verify and validate the SNAPE framework and its software implementation. In order to verify the modeling and simulation environment, the primary functional blocks that drive the navigation performance, state propagation and estimation, are compared to industry standard tools. Due to the lack of a pre-existing end-to-end navigation analysis framework, the verification was performed at a functional level. From a review of the currently available analysis tools, the state estimator was compared to STK and the state estimation filter was compared to ODTBX. These analysis cases are summarized in Table 25.

After verifying SNAPE's functional performance, the framework's analytical capability was validated through a series of test cases. These demonstrate typical navigation system

Table 25: SNAPE Implementation Functional Verification Cases

Section	6.3.1 Propagator Verification				6.3.2 State Estimator	
Case	F1	F2	F3	F4	F5	F6
Mission/Reference	MSL Mars Cruise Trajectory				LEO Trajectory	
Simulation Duration	20 days				300 seconds	
Primary Agent	MSL				Notional Vehicle	
Onboard filter	N/A				EKF - 6 State	ODTBX - Sequential EKF
State Propagator	STK - 2 Body	STK - HPOP RK4	STK - HPOP RK7(8) w/ SRP	SNAPE - dopri5	SNAPE - dopri5	ODTBX 2-body dynamics
Other Agents	none				Earth - DSN	
Nav Packet Source	N/A				N/A	
Nav Packet Frequency						
Analysis Approach					Monte Carlo - Vary Initial Error	
Packet Content					N/A	
Measurements					Earth-DSN Range, Range-Rate every 10 seconds	

analysis and performance evaluation. The validation demonstrates the capability of the integrated framework to perform a wide variety of analysis, from system analysis to optimization. A series of experiments was performed in order to provide additional knowledge and build a database of the capabilities of the filter for deep space navigation, looking particularly at direct state measurements. A summary of these cases is given in Table 26. The results of these test cases showed a strong correlation to time between measurements affecting the integrated error states, and the strong effect of the inclusion of a clock update. These demonstrate the capability of the framework to perform the analysis directives described previously and required to evaluate the capability of a deep space navigation system. This provides capabilities that will now be applied in a more complex problem, focusing on the analysis of performance for the NNAV architecture.

Table 26: SNAPE Framework Validation Cases

Section	6.4.1 Effect of Initial Error	6.4.2 Effect of Measurement Error	6.4.3 Timing Behavior		6.4.4 Timing Sensitivity	6.4.5 Integrated Performance	6.4.6 Measurement Content			6.4.7 Parameter Sensitivity	6.5 Measurement Optimization		
Case	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
Mission/Reference	MSL Mars Cruise Trajectory												
Simulation Duration	25 days				50 days	25 days							
Primary Agent	MSL												
Onboard filter	EKF - 6 State		EKF - 8 State										
State Propagator	Dopri 5												
Other Agents	Earth-DSN												
Nav Packet Source	N/A												
Nav Packet Frequency													
Analysis Approach	Monte Carlo - Vary Initial Error, Time Between State Updates	Monte Carlo - Vary Position Error, Frequency	Monte Carlo - Initial Error		Monte Carlo - Clock Stability, Time Between State Updates	Monte Carlo - Position Measurement Accuracy and Frequency	Monte Carlo - Initial Error			Monte Carlo - Time between Measurements, and Time to First	Genetic Algorithm Optimization of Position Meas. Error and Frequency		
Packet Content	N/A												
Measurements	State - DSN	Position - DSN	Weekly State, Daily Position - DSN	Weekly State, Daily Position, Weekly Time - DSN	Weekly State, Variable Timing - DSN	Weekly State, Weekly Timing, Variable Position - DSN	Weekly or Daily Position - DSN	Weekly or Daily Range - DSN	Weekly or Daily State - DSN	Variable State, Position, Time - DSN	Weekly State, Variable Position DSN		Weekly Time, Weekly State, Variable Position - DSN

CHAPTER VII

EVALUATION OF NETWORK-BASED NAVIGATION (NNAV)

To evaluate the capability of Network-Based Navigation (NNAV), a set of systems analysis tools is needed. Due to the unavailability of flight data, which would typically be used to inform design analysis, experiments must be performed in order to capture the expected performance of a navigation system. Due to the large costs and extended schedule required for an actual in-flight experiment, a software-based simulation has been developed to allow for investigation into the navigation architecture and in order to perform design trades and analysis on the given concept.

The previous chapters focused on the potential paths of development for such a framework. As shown by the research, no available tool exists for the analysis of communication-based navigation packets in a modular environment that allows for the rapid comparison of multiple measurement sources and varying navigation packet content. In order to capture the requirements and behaviors of the system under study, Model-Based Systems Engineering techniques are utilized to capture the system and design space. In addition to defining the analysis scope, these models further provide the basis for the software implementation by both capturing algorithm development and interface design. The test cases from the previous chapter were used to compare the tool against other analysis software's capabilities and to verify performance. With the framework verified and in place, it is now possible to return to the original problem and capture the performance of NNAV.

7.1 Analysis Scenario Description

The proposed method of communication-centric navigation has applicability to a wide range of mission scenarios and concepts. These can vary from a local transfer between Earth and the Moon to an outer planets exploration mission. The case study under analysis depends on two primary factors: the underlying communication infrastructure and the trajectory under study. The two must coincide to provide for an analysis case that allows for the

design and trade-off of the navigation update processes. Additionally, the initial analysis should focus on a near term concept that could be used for a potential flight experiment to verify in-space performance. In terms of trajectory designs, it is important to select an analysis case with a large degree of relevance to current mission designs and infrastructure implementations.

In order to capture the nearest term implementation of NNAV, this concept evaluation will focus on mission test cases defined within the scope of Earth-Mars transfers[1]. Several aspects of this mission provide a strong area for analysis of the proposed methodology. Although NNAV can be performed using Earth-Based assets, a greater capability is envisioned with a larger growing telecommunications deep space network.

This infrastructure has already begun to be implemented in the local Martian environment. MRO currently operates as a data relay for several Mars ground assets, such as the MER Opportunity and the Mars Science Laboratory Curiosity. The orbiting spacecraft has performed this role incredibly well, greatly enhancing the amount of data transferred from Mars to Earth, by utilizing its increased power and data transmission capability.

To enable the analysis of multiple navigation packet sources, the trajectory under study has been chosen to allow for communication with both Earth and Martian assets. The design mission focuses on an Earth to Mars transfer orbit. For this analysis case, the mission focuses on a first step implementation of NNAV, allowing for the integration of Earth-Based state updates (through DSN observation) which are augmented with the embedded navigation packets. This is chosen to enable demonstration of the capability of the navigation packets and to provide comparison to currently used methods. The unique dynamics of this analysis case are applicable to a large number of missions, as robotic missions continue to be sent to Mars at regularly intervals and the need for advanced navigation to support pinpoint landing increases to maximize science return. Additionally, these missions allow for analysis of a growing network, including study of Earth-only, and Earth+Mars relay studies. This will provide insight as to the effects of a growing network.

[1]Each test case included as part of the navigation evaluation is identified as N1, N2...

Selection of this analysis scenario also supports demonstration of the autonomous capabilities of the proposed navigation system. The vehicle's autonomy is captured in its ability to process state updates, and measurements independent of Earth-Based orbit determination. The embedding of navigation headers into the packets requires no a priori analysis, simply insertion of the variables of interest, such as transmission location and time. As such, the inclusion of this information forms an automated, integrated part of the transmission protocol and data packaging. Additionally, as the packet protocols are embedded into spacecraft among a growing network, the navigation process onboard each spacecraft is increasingly autonomous of Earth. For these initial studies, the analysis does include the generation and modeling of DSN state estimates, resulting from ground-based orbit determination. These states are still processed autonomously onboard by means of a loosely coupled filtering system, similar to that used in GPS receivers. As opposed to overwriting the onboard estimate with the DSN state update and resetting the filter, the packet is treated as a measurement with a given estimated uncertainty, allowing for optimal integration with the onboard estimate.

In order to maximize the relevance to possible missions, the Martian transfer trajectory of Curiosity is again chosen as the spacecraft under analysis. The published availability of the as-flown trajectory of MSL[2] and MRO[3] provides a truth reference for the navigation system and provides for a comparison case to capture state estimation performance of the packets and measurements under study. The specific measurements under consideration are the difference between the onboard estimated state and the true state (from the reference trajectory data). The trajectory chosen also simplifies the analysis case, reducing the number of potential bodies interfering with communications due to the orbital dynamics at the time of transfer. The trajectory under study is given in Figure 74, which shows the orbital path of Earth, Mars, and MSL.

Per measurement state estimation accuracy is captured by storing the error between true and predicted position, velocity, and time of the spacecraft's estimator and the reference

[2]http://naif.jpl.nasa.gov/pub/naif/MSL/kernels/
[3]http://naif.jpl.nasa.gov/pub/naif/MRO/kernels/

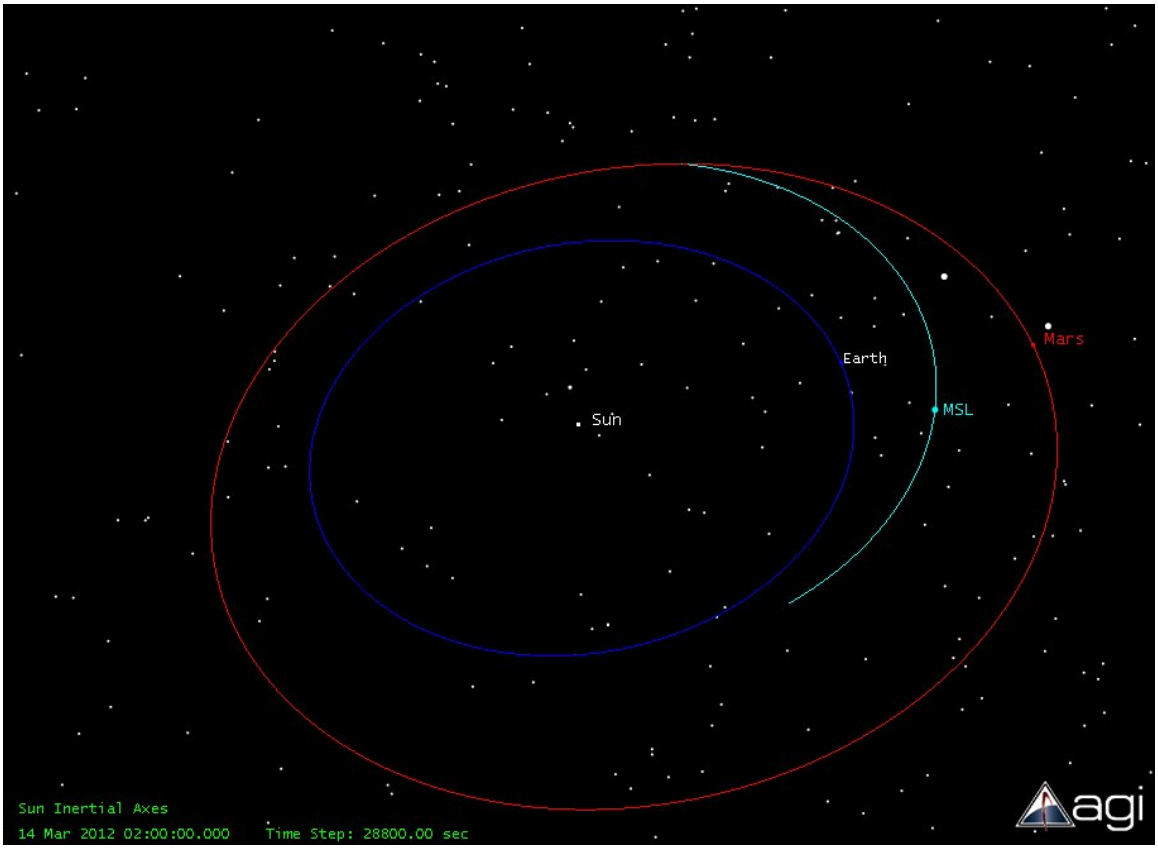


Figure 74: MSL Cruise Design Trajectory

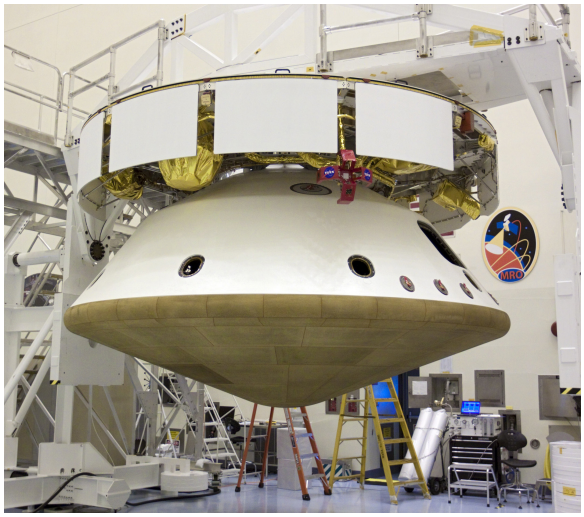


Figure 75: MSL Cruise Configuration (NASA/JPL)

trajectory. This data is captured over the entire analysis case to provide for a time-based history and trending analysis of the estimation errors. Additionally, the initial error state is captured to inform stochastic analysis of the design space, enabling linkage of the performance parameters to initial random state. Capturing the error state at the end of the simulation provides for a measure of capability at the end of the mission, which is very important for local orbit insertion and atmospheric entry. Another measure of performance is needed to capture the overall performance of the navigation system. To capture this capability, the integrated error terms are used. These give a direct capture of the total error incurred over the course of the mission and also capture well the errors in propagation between state estimation updates.

7.2 Implementation and Vehicle Definitions

The as-flown description of the MSL cruise stage and MRO final configuration are used as the vehicles of interest are given in Figure 75 and Figure 76. For the simulation, the main parameters of interest relate to the communications systems on the cruise stage [75]. During cruise, the spacecraft flies with a constant roll rate and its solar panels pointed towards the sun.

The main antenna to communicate with earth is the onboard Small Deep Space Transponder, which connects to the Medium Gain Antenna (MGA) which feeds through the solar

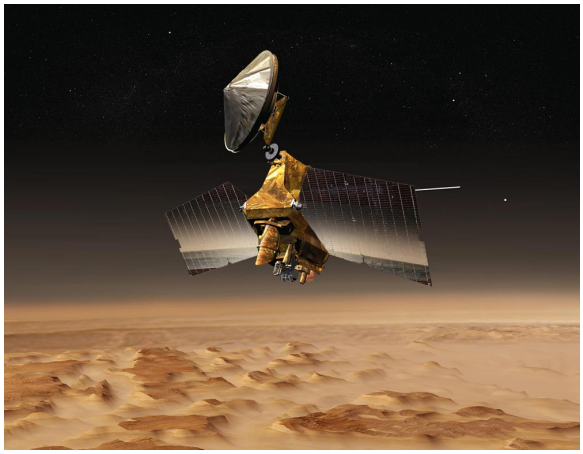


Figure 76: MRO Vehicle Model (NASA/JPL)

panel structure via a traveling wave tube amplifiers (TWTA) to receive signals from earth. This is the antenna typically used for Earth-communication during its cruise phase. Additionally, several lower power antennas are on the other side of the spacecraft, such as the Parachute Low Gain Antenna (PLGA), which are used during the descent portion of flight. This antenna would primarily point towards Mars during cruise.

This study does not address the attitude of the spacecraft during cruise, but does make the assumption that upon reception of a signal from MRO, it has the capability to perform a flip maneuver and point towards Mars (or alternatively have a similar antenna pointing out of the vehicle's nose for Mars-based communications). Both scenarios will be analyzed to capture the capability of communication with MRO's High Gain Antenna (HGA) and Medium Gain Antennas (MGA). Block diagrams of the two spacecraft's telecommunications systems are provided in [75] and [119].

An additional assumption of the analysis is that the simulation begins post vehicle checkout, with the spacecraft operating with a fairly accurate state estimate from initial orbit determination during vehicle check out procedures. To capture this assumption, the initial position and velocity of the onboard state estimation are set to truth and the accuracy of DSN state determination (one kilometer and one tenth kilometers per second). This uncertainty is applied as a random normal with given standard deviation to the reference trajectory state. Over the course of the simulation, the truth state is continually referenced

to the loaded trajectory data. Between packet receptions, the spacecraft propagates its own state by means of a Runge-Kutta 4(5) integration algorithm. At each time step, the vehicle estimates the forces being applied by using the onboard ephemeris, DE421, and its current state estimates of position, velocity, and time. As such, uncertainties in the onboard state directly affect the autonomous state propagation and integration of the vehicle.

In addition to the usage of space-based assets, the study also captures the performance of the navigation packets to enhance and reduce the reliance on traditional Earth-Based navigation state updates. As such, Earth-Based antennas must also be modeled for completeness. Due to the global spread of the DSN assets, the individual assets are not modeled. Rather, the communication infrastructure is captured by representative dishes at the three main ground stations (Spain, California, and Australia), the 34m and 70m dishes. Due to the global coverage and location of these dishes, the spacecraft can always be in potential contact with one of the sites. The main parameters of interest to the packet design are the operating frequency, line losses, receiver and transmitter gain, and transmission power. This data was obtained from that published in the open literature [119] [75].

7.3 Packet and Measurement Content

The navigation packet content and the measurements defined in the simulation have been chosen to match the above described mission scenario. The first step in the analysis is to capture the effect on state estimation of DSN-based state updates. This measurement data includes both position and velocity measured to a high level of accuracy during extended navigation passes. Over the course of a navigation pass, range and range-rate are measured to the spacecraft by the ground stations via two-way ranging and Doppler observations. By using advanced orbit determination software and observation data over long passes, ground analysts are able to determine the vehicle's state to a high degree of precision. This modeling environment captures the overall performance of these state estimation techniques as opposed to directly simulating the ground-based orbit determination process. The analysis assumes a DSN accuracy of one kilometer in position with one-tenth of a kilometer per second in velocity. These values are average values for distances over this trajectory [121].

This accuracy ultimately reduces with increasing range [74], but this value is reasonable for the early parts of Martian transfer on which the analysis focuses.

The state update is modeled as a packet containing the spacecraft's state at the expected time of reception with applied random error to account for uncertainties in the orbit determination and propagation process. The actual value is taken in terms of the true trajectory position from the loaded SPICE data with standard normal error applied to each measurement. It is assumed that the ground can accurately propagate the vehicle state forward such that the state in the packet is the vehicle's position and velocity at the expected time of reception. The bulk error uncertainty includes effects of ground timing uncertainty and atmospheric transmission effects. Upon reception, this state update packet is processed onboard as an instantaneous measurement of position and velocity. The capability exists to additionally model range, range-rate, and timing state updates, all of which are determined from the vehicle's reference truth data source.

Several data types are included in the packet analysis approach. For this analysis, the packet content is limited to the use of ranging packets. Two sets of data can be included that are processed at run-time to simulate real-time measurement. The minimum packet content required is the time of transmission of the packet. As described previously, in order to process the data header, the spacecraft must have some estimate of the transmitting body's state. In an operational scenario the current best known state of each body in the network is uploaded to the spacecraft. This information, in addition to the onboard planetary ephemeris models such as DE421 [39], is used to allow the vehicle to process received packets into state measurements that can be utilized via the state estimation subsystem.

The main trade in packet content is to analyze the required accuracy of this estimated data. For other bodies in the network, state propagation using incomplete gravity and dynamics models will cause error growth in estimated state over time. To address this declining accuracy of external asset information over time, the packets can also allow for the updating of the ephemeris data to the current best known state. As such, the other main data to be included in the navigation packet is the current position of the transmitting asset

at the time of transmission. This can be used to update the spacecraft's onboard estimated state to aid in both measurement processing as well as future pass planning. Additionally, this data represents the spacecraft's best known state and links back to the capability of the network to auto-correct and transmit updated state information throughout the network by consistently updating each node's onboard estimates.

7.4 Simulation Variables of Interest

To answer the research questions outlined above, several trade studies are required. These trades capture the performance of the navigation packets. Additionally, the simulation should provide demonstration of the integration of standard state measurements with the navigation packets. The framework must also be capable of comparing the performance of the new navigation architecture to traditional measurements. To support these trades, several design variables are identified to explore the design space. These settings allow variation of several parameters, focusing on the generation and frequency of both state measurement updates and packet generation. The main parameter for the ground-based position and velocity measurement is the time between successive measurements. The error in measurement is assumed constant and based upon literature descriptions [74][121].

To capture the effects of the navigation packets, several additional parameters must be included. Each navigation contact between two agents is modeled as a transfer of a batch of packets containing information. The underlying data transmission protocol defines the exact structure and error correction of the transfer. Each group of navigation packets is modeled as a batch of arriving information. The main parameters identified in the state analysis are the time between measurement batches (to capture the idle time required between transmissions), the time between individual measurements (during a transmission exchange, how often are the navigation packets included), the number of measurements in a batch, and the data content of the packet. These identified parameters allow for analysis of the packet structure and analysis behavior.

7.5 Sensitivity to Packet Content

With the analysis framework implemented and the design cases set, it is now possible to utilize the framework to analyze the performance of NNAV. For this analysis, it is assumed that there is an Earth-Based state update once a week, and the communication between Earth and MSL occurs once a day, allowing for a batch of measurements per day. This set of data cases focuses on the importance of the measurement states and the direct information content of the individual packets. In this design case, it is again assumed to have once-daily received navigation packet batches from Earth with weekly state updates. The number of measurements in a batch and the time between each is fixed.

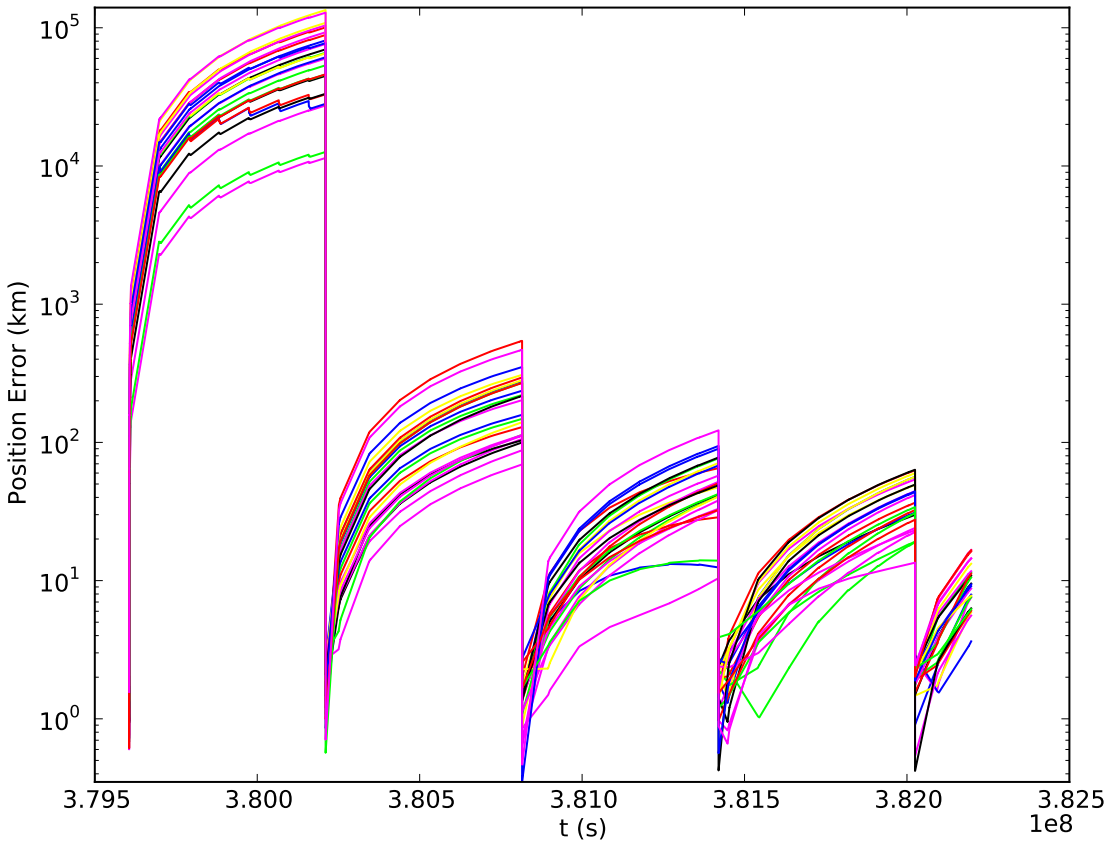


Figure 77: (N1)Position Error for Packet with Full Content

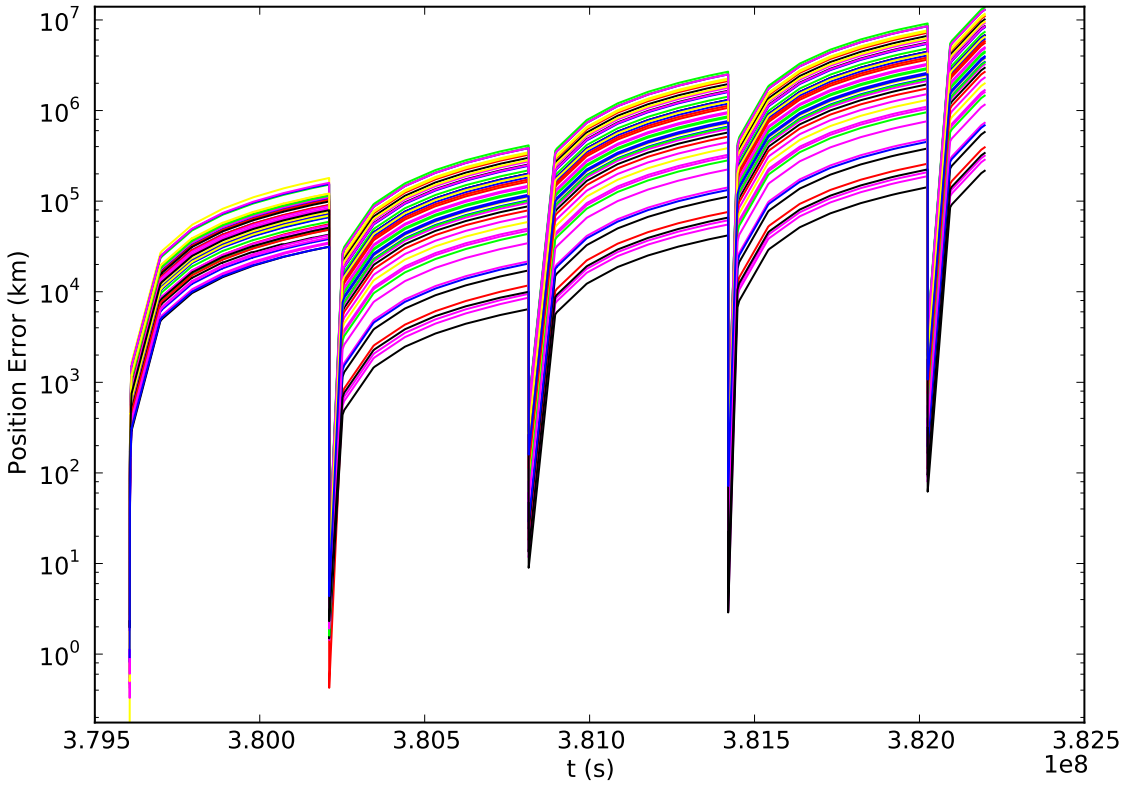


Figure 78: (N2)Position Error for Packet with only Timing

As seen in Figure 77 and 78, the importance of the data in the packet is clear. With only a time update, the spacecraft must rely on its internal propagation techniques to process the measurement. Over time, if not updated, this will cause increasing errors. Corrections to this include the capability to have improved dynamic models or include the other vehicle's state in the main vehicles total estimated state. With the transmission time and the transmitting spacecraft's best estimate of its location, the performance is much improved.

7.6 Packet Measurement Performance

To capture the capability of the packet measurements, it is possible to analyze the navigation system specifically looking at the onboard state estimation capability in terms of range and range-rate observations. Utilizing measurements allows for reduction in onboard error effects

by decoupling the measurement from other onboard error sources. This allows for analysis of the capability of these types of measurements and studies focused on the geometry of the space assets involved. These also allow for comparison to other currently implemented navigation approaches, such as one-way ranging integrated with Doppler observations, that allow for onboard measurement of range and range-rate (such as Electra).

For this analysis, it is assumed that the vehicle can measure its range and range-rate relative to a satellite in Martian orbit (MRO). The reference trajectory used is pulled from the published MSL data, and serves as the truth for calculation of the state estimation position and velocity errors. It is assumed that the vehicle has a daily scheduled pass with the other satellite allowing for a batch of 30 measurements, with a time interval of 10 seconds. For this test case, N3, the range and range-rate error were varied to capture the overall performance possible using this type of observation.

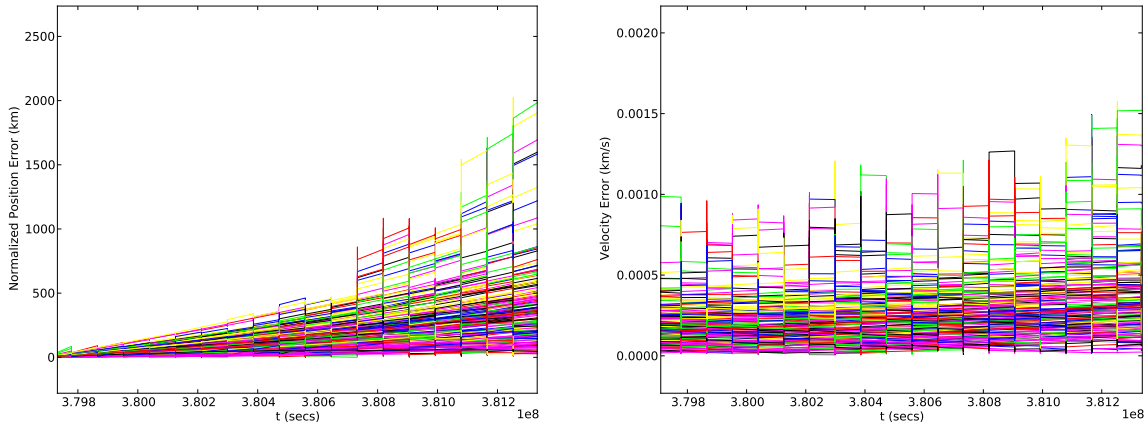


Figure 79: Dispersed Simulation Errors

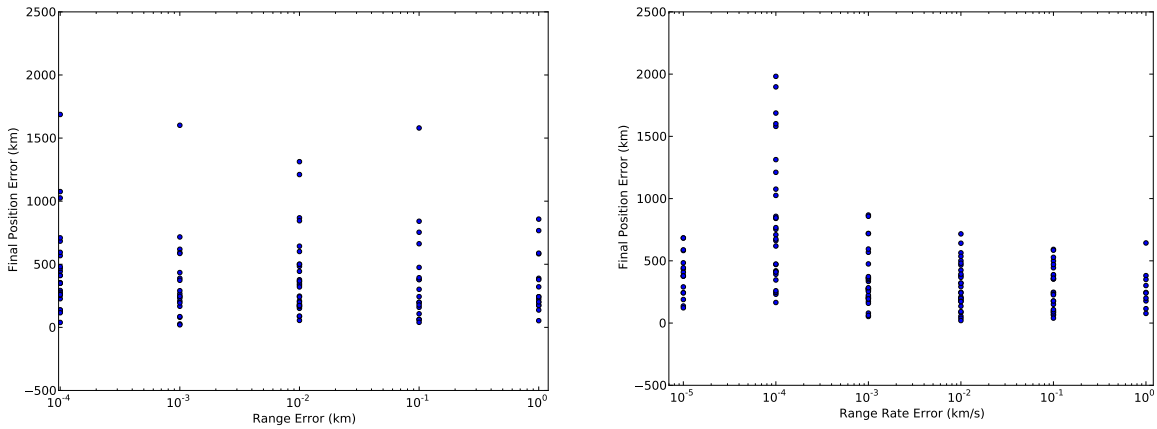


Figure 80: Effect of Measurement Errors on Final Position Error

Figure 79 captures the simulated position and velocity errors over the course of each simulation with the varying range and range-rate measurement uncertainties. This displays the dynamics of each scenario case, and shows that at this level of initial accuracy and onboard dynamics, a range of position error is achievable. In order to analyze the effect of the individual measurement errors, the final position error is plotted against the range and range-rate error settings. This is shown in Figure 80. These two plots demonstrate that the final error is not a strong function of the measurement errors. The lack of a clear relationship is due to the accurate state dynamics models and assumed onboard measurement uncertainty used in the state estimation filter.

To provide further analysis of this data, the collected data for the simulation runs was integrated with the final state errors and integrated state errors to generate a series of surface plots. These are given in Figure 81 and Figure 82 which capture the final position and velocity errors and the integrated position and velocity errors. The x- and y-axes of this plot represent the base ten logarithm of the defined range and range-rate errors terms. The individual circles capture the analysis points. Due to use of the Monte Carlo Analysis, there may be multiple observations for one combination of range and range-rate error. A two-dimensional interpolator using a Delaunay triangulation was used to determine the individual points of the plotted surface. As observed, the data do not show any clear trends in terms of the combined effects of the two input error terms in terms of either final

or integrated error. These plots demonstrate that the main driver to the state estimation performance is due to the stochastic nature of the simulation. This is a factor of the specific errors of each measurement, and the onboard estimation process, which is driven by the defined process noise and dynamics models. As such, it is likely that the filter can be tweaked in terms of its operational parameters to improve performance.

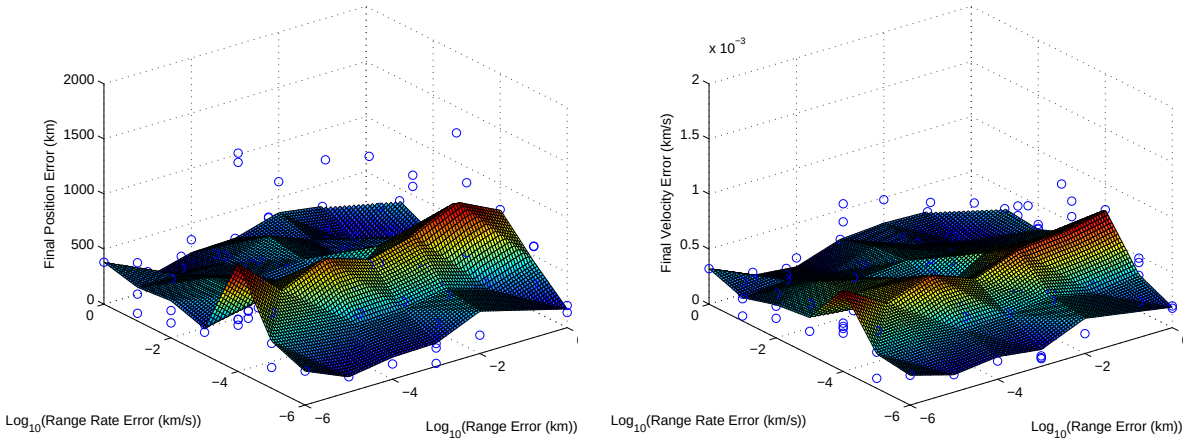


Figure 81: Final State Errors vs. Measurement Errors

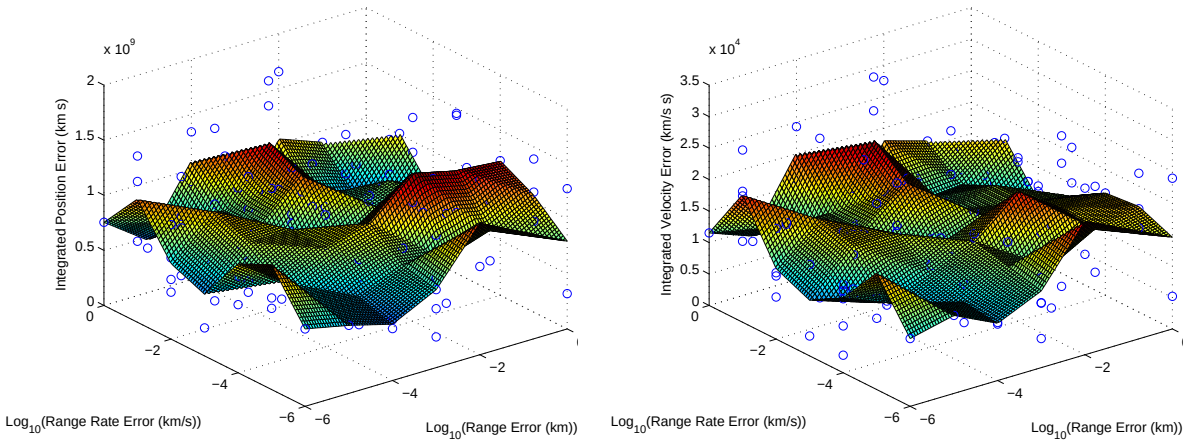


Figure 82: Integrated State Errors vs. Measurement Errors

7.7 Packet Timing Optimization

With an overall view of the navigation system performance, it is important to also optimize the packet frequency in order to minimize the error states. This test case is identified as N4. This utilizes the Genetic Algorithm interface to search the design space for an

optimal solution. The evaluation consisted of equally minimizing the final state errors as well as the integrated error terms. To match with the developed concept, the time between measurement batches was fixed to one day. The time between measurements in a batch was allowed to vary between one and two minutes, and the number of measurements had an allowable range between five and one hundred. Additionally, the time to the first measurement could be between an immediate update and a one day delay.

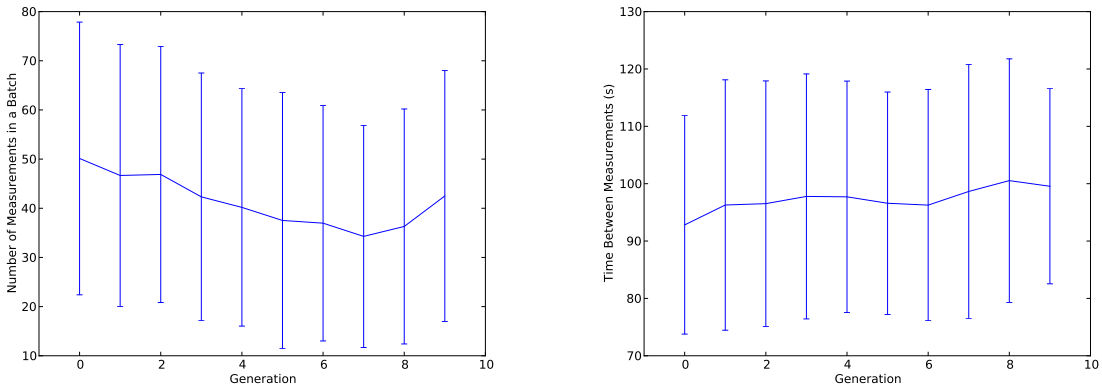


Figure 83: Input Variables over Optimization

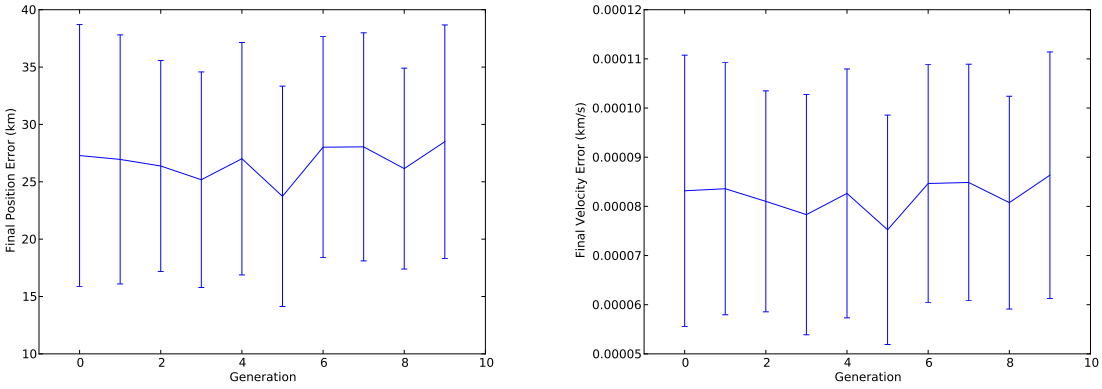


Figure 84: Final Errors over Optimization

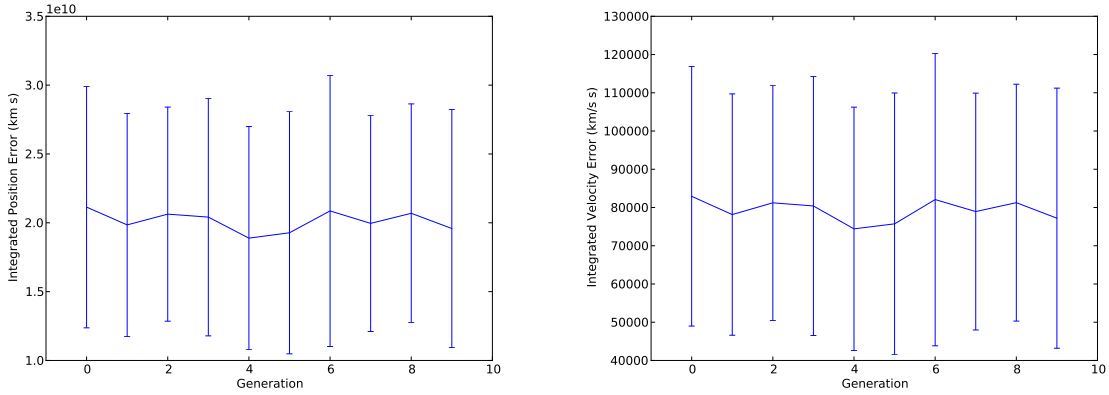


Figure 85: Integrated Errors over Optimization

As seen in the summary graphics of Figures 83, 84, and 85, the existence of the packet helps to maintain a very low error state. It also shows that for this analysis case, the onboard dynamic models and weekly updates integrated with these packets allow for very low dynamic noise levels. It is expected that as the change in time between state updates increases or using less accurate dynamic models with more noise, the packets will have improved performance.

7.8 Comparison to Current Methods

With the general functionality and performance of the packets captured, these results can be used to feed into capability analysis. This allows for optimal analysis (in terms of packet content and frequency) of the state estimation performance of the approach and comparison of multiple navigation techniques. The following sections specifically address the current baseline capability, focusing on state updates, an analysis scenario collecting the performance of only using navigation packets, and an integrated case with the packets augmenting traditional state measurements. The published MSL trajectory data is used for the truth trajectory for all analysis cases, and includes allowance for internal state estimation errors and uncertainty.

This trajectory also serves as the reference for all error calculations. The normalized errors are captured as the square root of the sum of the individual error in each axis. For

these the onboard estimated state is compared to the true trajectory data. The integrated error terms are simply the integration of the position, velocity, or time errors at each simulation time. These are calculated numerically to form an integrated error to serve as an overall performance metric of the entire trajectory.

7.8.1 Baseline Analysis

In order to understand the effect of integrating navigation packets with traditional measurement techniques, it is first necessary to capture the performance of a mission scenario utilizing these methods. This analysis (N5) focuses on the case of a navigation system utilizing state updates at a weekly time interval to its onboard state estimation filter. The generation of these state updates is assumed to be performed by ground-based orbit determination systems. These process two-way or three-way ranging and Doppler observations captured over extended navigation passes (on the order of eight hours in length).

Integration of this data with high fidelity dynamics models, error estimation models, and orbit determination routines allows for accurate prediction of the spacecraft's position and velocity. It is assumed that this information is combined with the planned time of transmission to determine the spacecraft's state when it receives the state update. Upon receiving this packet, the onboard state estimation routine processes this as a measurement of its position and velocity also utilizing the expected uncertainty of the observation.

The bulk error properties in terms of normalized position and velocity error capture the performance of the ground estimation routines, as opposed to detailed modeling and simulation of those processes. This allows the analysis presented to focus on the capability of the onboard filter to autonomously process these state updates. For this simulation, which focuses on the early section of the MSL transfer trajectory, assumes a ground orbit determination error of one kilometer in position and one-tenth kilometers per seconds in velocity. In order to generate these measurements, the true values are obtained from the published MSL SPICE trajectory. The identified errors terms are applied as the standard deviation of a random normal distribution to the true state.

In order to capture the stochastic effects of the measurements and errors, a Monte

Carlo Analysis was performed on the design space. The central parameter being varied is the initial state errors, with an assumed mean of one kilometer and one-tenth kilometers per second. This initial error is applied as the standard deviation of a random normal distribution to the true state at the simulation start time. The simulation was propagated forward for twenty days from the start of the transfer trajectory. The analysis scenario was repeated for 20 iterations, capturing a range of initial conditions and state estimation errors. The results are given in Figure 86 and Figure 87, capturing the difference between the onboard estimated position and velocity and that of the truth trajectory reference.

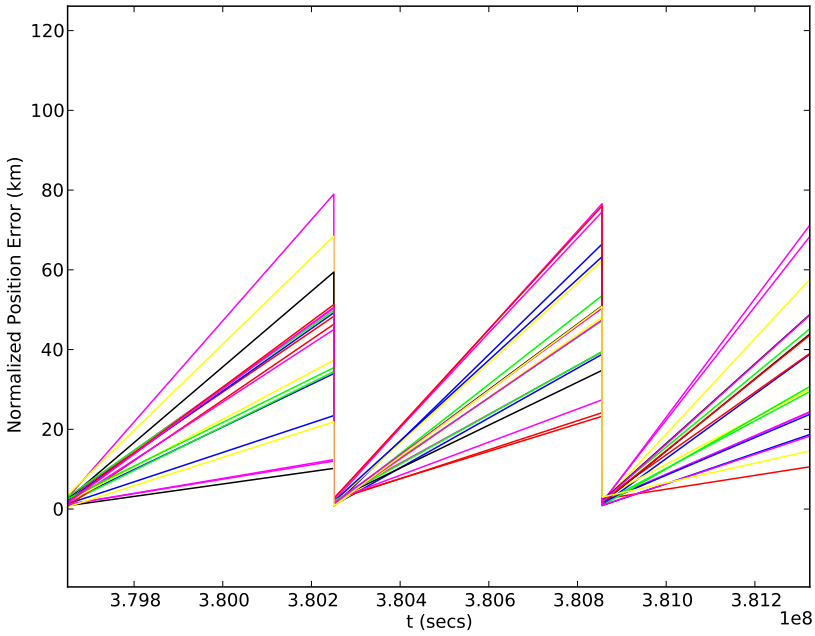


Figure 86: Position Error with Weekly State Updates

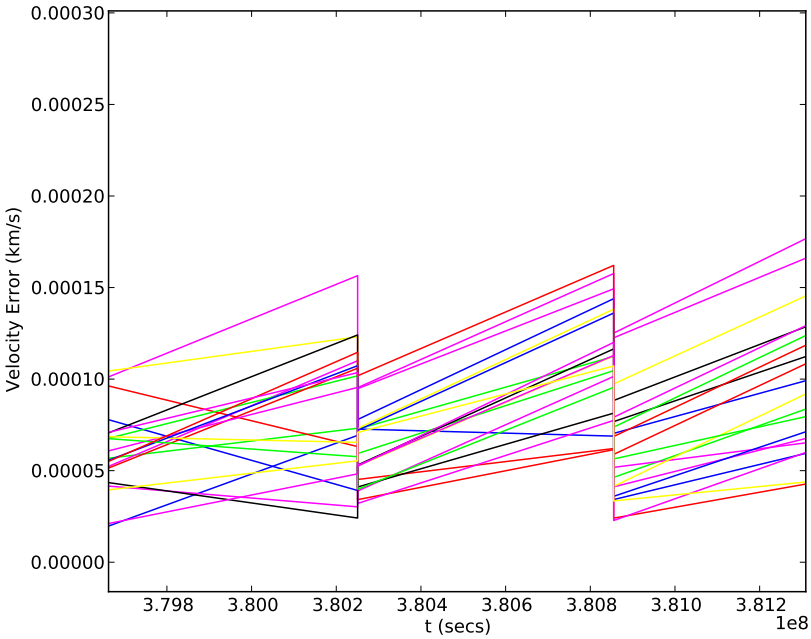


Figure 87: Velocity Error with Weekly State Updates

As can be seen in the presented data, the state updates serve to provide a very accurate estimate of the spacecraft position and velocity. This demonstrates the capability of such accurate measurements that contain information specific to each axis, allowing for six degrees of freedom. This performance can be observed through analysis of the propagated error between state updates. As seen, with these state updates, the onboard estimation routine is able to maintain position errors below eighty kilometers between state updates and velocity error below a meter per second. This additionally demonstrates the capability of the onboard propagator to integrate the vehicle's state.

7.8.2 Packet Autonomous Operation

In contrast to this state-of-the-art analysis case, it is also important to analyze the capability of just using the navigation packets. For this test case, N6, it assumed that the vehicle is initialized by means of a high accuracy state update to reduce its initial errors and uncertainty. This is implemented the same way as the previous analysis, but applied only once, 1 day into the 20 day simulation window. After this one measurement, the updates are limited to navigation packet updates.

The two navigation sources considered are an Earth-Based asset (assuming no delays due to atmospheric effects, in order to focus on capturing the effect of the scenario's geometry on onboard state estimation) and MRO. For the truth data of each, the published SPICE ephemeris was used. It was assumed that each has a very accurately maintained clock, as would be expected for nodes in a high bandwidth relay satellite, and known position. The navigation packets are defined to include the host's position and time at transmission. The time of reception is measured based on the truth position of the assets and calculated one-way light travel time, calculated via SPICE.

In order to account for timing errors, a stochastic model is utilized for the MSL timing measurement system. It is assumed that the onboard oscillator has specified h_0 and h_{-2} of 1E-19 and 1E-20 to match the capability of a standard onboard crystal oscillator [13]. Thus, it is possible to capture the stochastic bias and drift of the onboard clock, which directly drives the uncertainty.

At the time of true reception based on the truth dynamics, the packet is considered to be received by MSL. Upon reception, the vehicle notes its onboard time as the time of reception. This time includes the stochastic errors and effect of timing bias and drift. Upon reception of the first packet, the vehicle is able to process the information as a range measurement. After receiving an additional packet, it can use the combine the information about the last two received packets to additionally estimate a range rate. The range is measured in terms of the time difference between transmission and reception, and modeled onboard based on the received position data and onboard estimate. The range rate is measured as the ratio of the time between transmissions to the time between receptions of sequential packets. This value is modeled in the state estimation procedure using the current state estimate with the received transmission host state.

Similar to the previous section, the simulation was propagated for twenty days. It is assumed that the packets from Earth and MRO occur at daily intervals separated by twelve hours, such that the spacecraft receives measurements from each host once per day. The two packet exchanges are offset by twelve hours in order to minimize time between measurements. The results are given in Figure 88 and Figure 89.

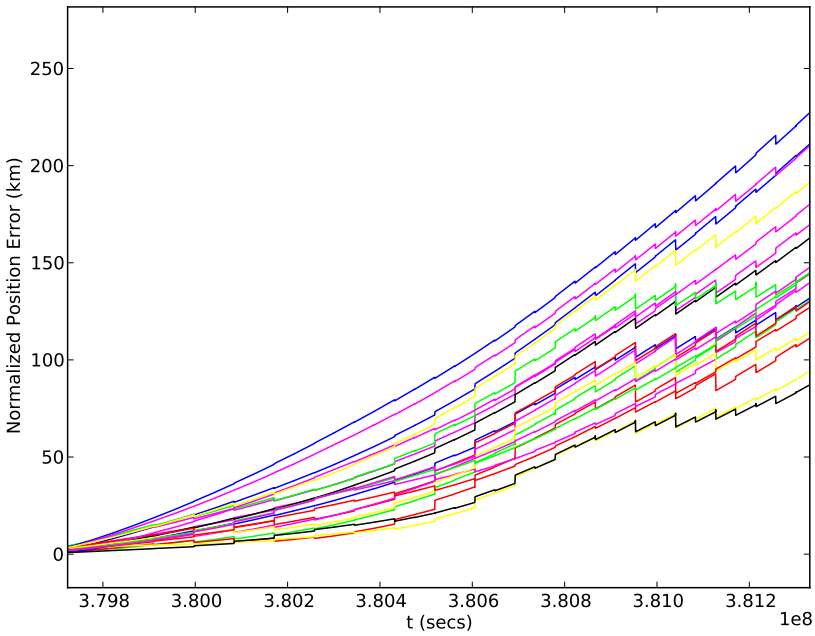


Figure 88: Position Error with Packet Updates

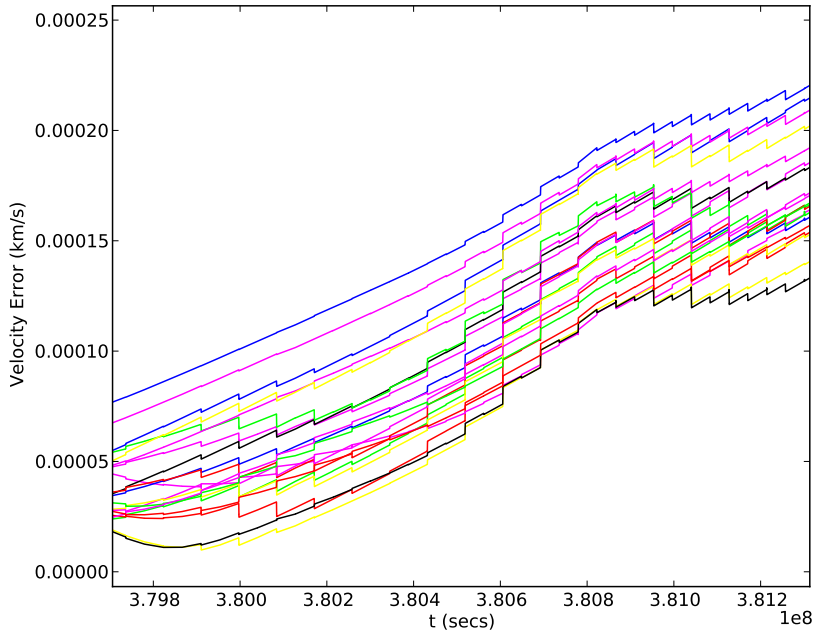


Figure 89: Velocity Error with Packet Updates

As presented in the two figures, using navigation packets from only two sources does not maintain the same performance accuracy as repeated state updates. This is expected

due to the reduced dimensionality of range and range-rate observations. Additionally, the stochastic onboard clock error limits the overall accuracy of the timing measurements. The packets though do limit the growth of estimation error over time and do serve to maintain the error within two hundred kilometers after twenty days of propagation and similar order of velocity error as the state updates presented above. This analysis demonstrates the potential for linking the state updates with the packet observations and the benefits of integrating the two systems, utilizing ranging packets to maintain high accuracy state estimates between state updates.

7.8.3 Packet Augmentation

From the analysis, it is clear that the state update coupled with a strong dynamics model, provides a very good state estimate over the course of the simulation. The packets do work to bound the error, but the effect is reduced by the frequency of the state updates. In order to capture this effect, an additional analysis was performed looking specifically at this.

For this analysis, two cases were analyzed, one without packets, N7, and one with, N8. A Monte Carlo Analysis was performed on the time between state updates. This approach was taken in order to capture the stochastic nature of the applied errors. The two scenarios were each run 100 times for a simulation duration of 60 days, the primary parameter of interest is the time between state updates, which was varied between 1 and 20 days. Following the analysis, the data for the final position error and the clock error terms were collected and analyzed. The resulting scatterplots are given in Figures 90, 91, and 92.

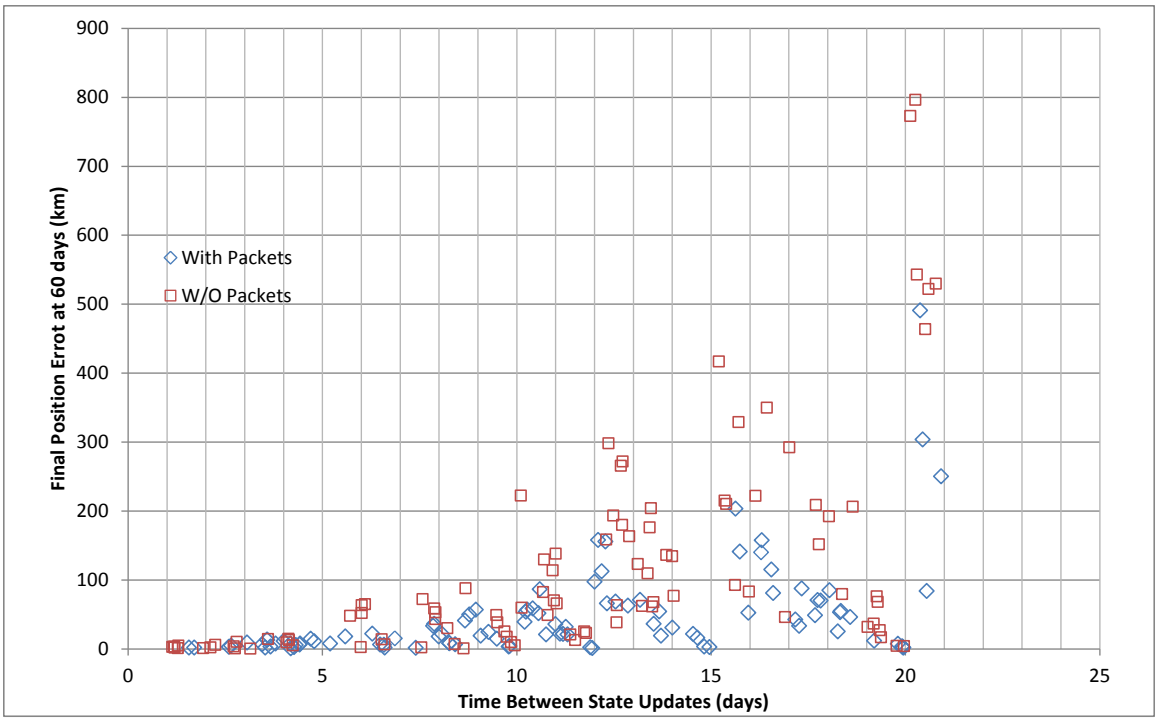


Figure 90: Position Estimation Performance after 60 days with and without Navigation Packets

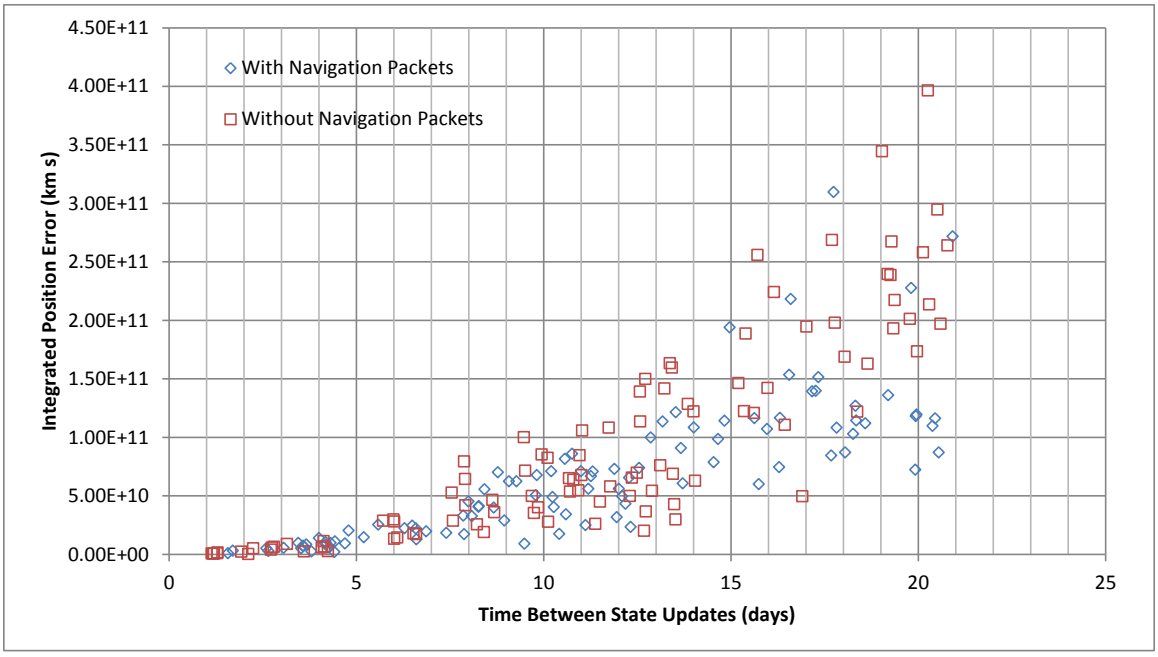


Figure 91: Integrated Position Error after 60 days with and without Navigation Packets

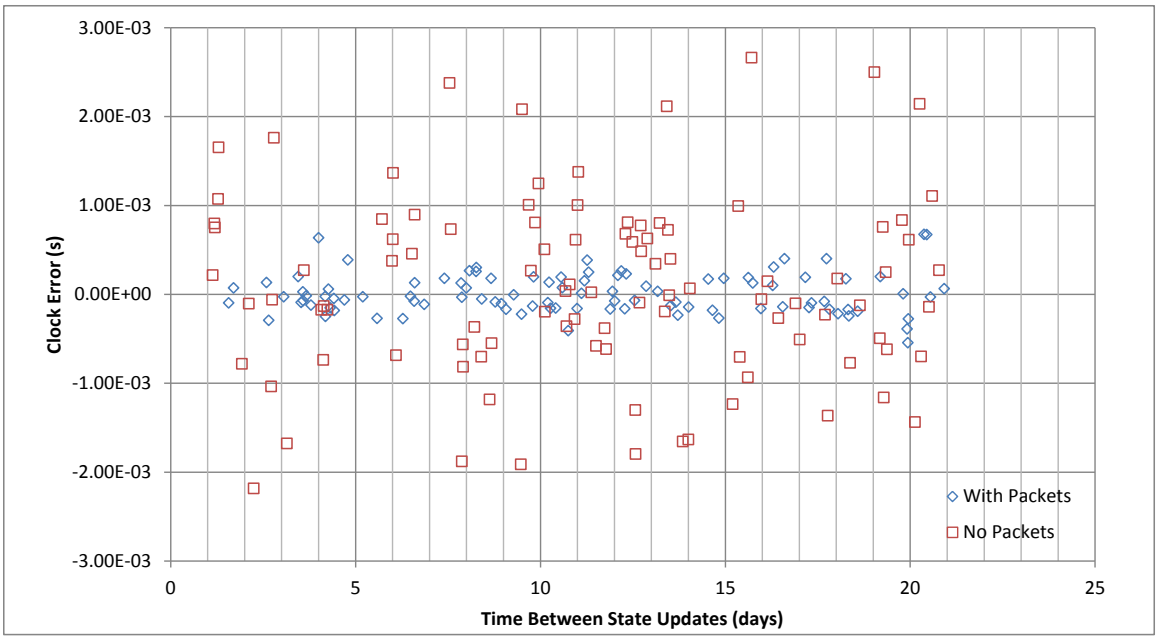


Figure 92: Clock Estimation Performance after 60 days with and without Navigation Packets

With these observations, the benefits of the packet based navigation are apparent. For frequent state updates, the position estimation performance is very similar. As the time between updates increases, the packets show a slight improvement to the navigation system performance. Similar results are exhibited in the velocity and integrated errors terms. An even stronger advantage is shown when analyzing the clock error. As can be seen in the Figure 92, the packets allow for a fairly standard control of clock independent of the time between state updates, whereas the state measurements alone do not track the error as well. This is primarily due to the packet processing algorithms directly including the effect of the clock bias in its models and allowing for capturing the clock uncertainty.

These test cases demonstrate a reduced reliance on DSN full state updates over the course of the trajectory. This both reduces the need for these expensive, ground analysis-intensive updates well also providing onboard autonomous navigation capability. The reduction in number of ground navigation updates reduces mission operations costs. The spacecraft is thus able to update and correct its onboard state estimate as soon as any communication packet with navigation header is received. This both reduces the latency of state

updates (as opposed to long ground passes, coupled with ground analysis, verification, and up-link) and allows the spacecraft to better plan its operations with the increased knowledge. Additionally, this reduced reliance on ground-generated state updates reduces the risk of ground failures affecting spacecraft mission performance. This is further expanded through the expansion of the navigation network to an increased number of space-based assets.

7.9 NNAV Limitations

As shown in the analysis, there are some limitations to the NNAV concept as demonstrated. Though the architecture does demonstrate a clear benefit in terms of reducing long term navigation error by augmenting traditional navigation methods, the navigation performance is still dependent on the periodic ground-based state updates. This is primarily due to the limited information inherent in the range and range-rate measurements to only two hosts. As the network grows, this accuracy will increase, and the reliance on ground orbit determination-based state update will continue to decrease. This is due to the simplified onboard dynamic models in addition to the limited Extended Kalman Filter implementation. As more measurements to a distributed network of spacecraft are observed, increased information is available, allowing for further improvement in the NNAV-based state update.

As additional stochastic forces are included in the truth trajectory, the capability of the navigation packet will continue to increase. This is due to their allowing more frequent state updates, and bounding of the state errors under increasingly uncertain dynamics. More complex filtering routines, as well as incorporation of onboard batch processing similar to AutoNav [96], will additionally improve the state estimation performance under increased uncertainty. The main drawbacks to this improvement is the increasing computational performance required for their operation.

7.10 Summary of Demonstrated Performance

This chapter has presented a range of analysis scenarios that utilize the SNAPE framework to analyze adn demonstrate the state estimation capability of the NNAV framework. A summary of the specific test cases from the NNAV evaluation outlined is given in Table 27.

This table references the cases analyzed and provides a summary of the variables of interest, agents involved, and analysis method (whether statistical design space exploration or optimization). A range of operational scenarios were analyzed to capture system performance, packet data content sensitivities, and to demonstrate the capability of NNAV.

As can be seen in this chapter, there is a tangible benefit to using packet-based navigation. This benefit is due to the reduced need for Earth-based state updates, and the increase in onboard autonomy by moving navigation routines to the spacecraft. But with the current state estimation implementation, it may not be able to independently reduce the state errors to low levels to match frequent DSN measurements. The architecture shows increasingly benefit in scenarios where state updates from the ground may be less frequent or where it is desired to reduce reliance on ground systems. This analysis captures the initial performance estimate of this method, and with additional development and improved filtering and data processing techniques, there is the potential for further improvements in navigation performance.

Table 27: NNAV Evaluation Cases

Section	7.5 Packet Content Study		7.6 Measurement Sensitivity	7.7 Packet Optimization	7.8 Comparison to Current Methods			
Case	N1	N2	N3	N4	N5	N6	N7	N8
Mission/Reference	MSL Mars Cruise Published Trajectory							
Simulation Duration	20 days							
Primary Agent	MSL							
Onboard filter	EKF-8 State							
Other Agents	Earth-DSN		MRO, Earth-DSN	MRO, Earth-DSN	Earth-DSN	MRO, Earth-DSN	MRO, Earth-DSN	MRO, Earth-DSN
Nav Packet Source	Earth-DSN		MRO	Earth-DSN	None	Earth-DSN, MRO	None	Earth-DSN
Nav Packet Frequency	Daily		Daily	Optimization Variable	N/A	Daily	N/A	Daily
Analysis Approach	Monte Carlo - Initial Error		Monte Carlo - Measurement Error	Genetic Algorithm	Monte Carlo - Initial Error	Monte Carlo - Initial Error	Monte Carlo - Time Between State Upates	
Packet Content	Transmission Time	Trans. Time and Location	Trans. Time and Location	Trans. Time and Location	N/A	Trans. Time and Location	N/A	Trans. Time and Location
Measurements	Weekly State - DSN		None	Weekly State- DSN	Weekly State- DSN	None	State Update - DSN	State Update - DSN

CHAPTER VIII

CONCLUSIONS

This thesis develops the Network-Based Navigation (NNAV) approach to deep space autonomous navigation. In order to characterize this navigation system and provide performance estimates of its capabilities, this research developed the Space Navigation Analysis and Performance Evaluation (SNAPE) framework. The CRAIVE approach utilized in this research is given in Figure 93. The arrows in this diagram capture the prerequisites of each individual step. This demonstrates the flow of the research from the initial definition of the navigation concept through analysis, implementation, and eventually to its evaluation. The analytical results can then feed back into the overall concept, providing for iteration and evolution of the navigation architecture. Each of these steps will be summarized below.

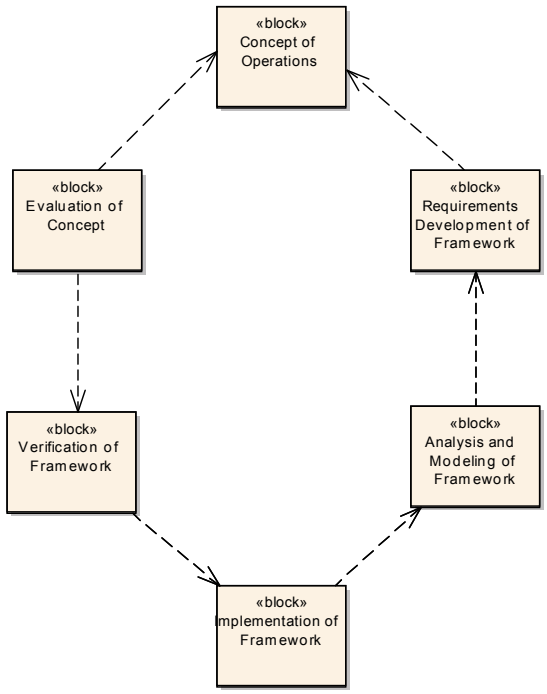


Figure 93: CRAIVE Research Approach and Structure

8.1 NNAV Concept of Operations

The first step in this research was to develop the operational **C**oncept of the NNAV navigation architecture. This served to capture the high level functions, capabilities, and interfaces in a way that provides insight into the analysis process. This concept formed the basis of the implementation and use of simulation tools. NNAV operates independently of ground analysis by taking advantage of an expanding space communication and navigation infrastructure. This integration is achieved by embedding navigation headers into the communication packets being sent between assets in the network. The concept of operations is given in Figure 94. This figure depicts a representative implementation of the proposed NNAV architecture and visualizes the transmission of integrated communication and navigation measurements.

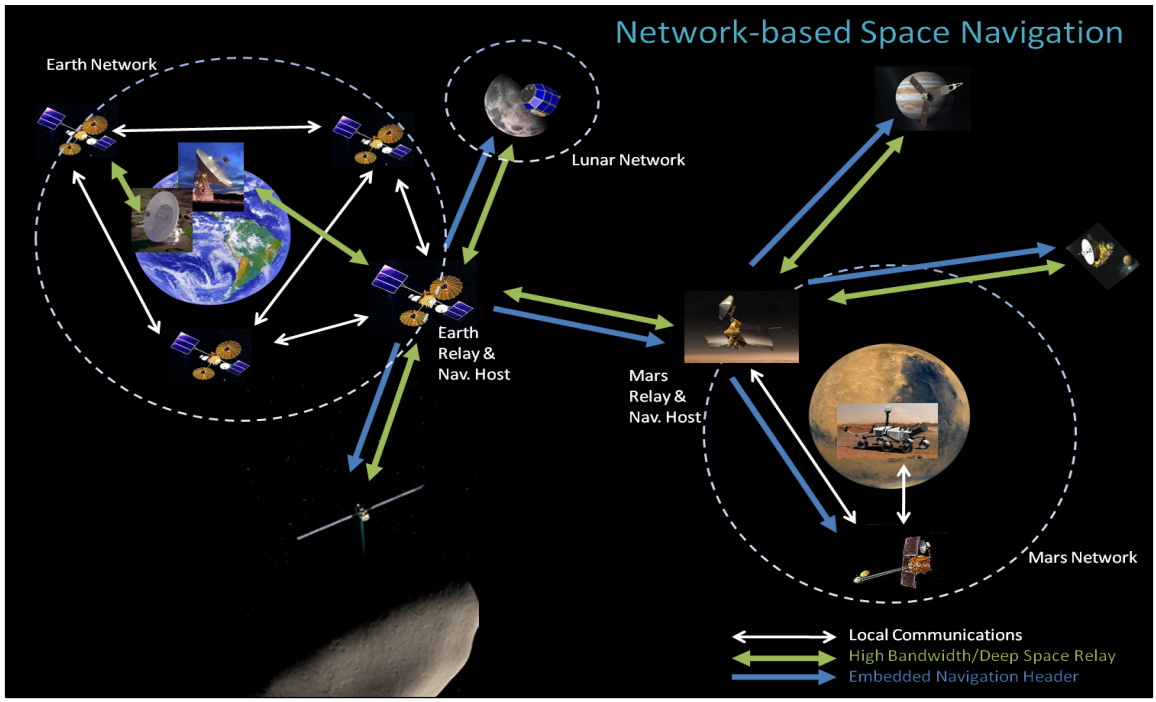


Figure 94: NNAV Concept of Operations

From this operational concept, a series of hypotheses were developed to characterize the predicted performance of this navigation architecture. For each of these hypothesis, a series of research questions were derived to capture the approach and specific goals to be

addressed. These requirements and their associated traceability are given in Figures 95 and 96.

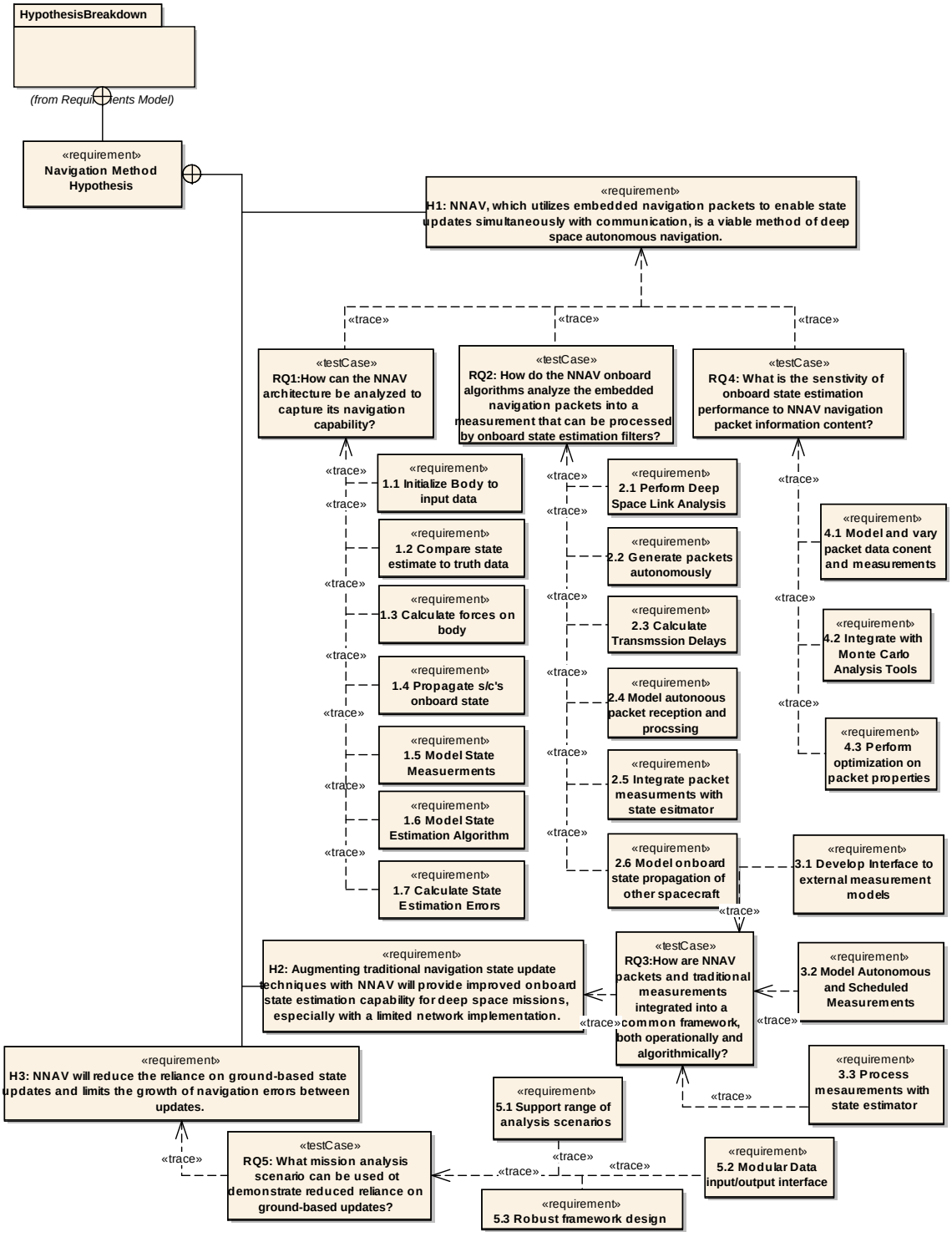


Figure 95: NNAV Capability Hypotheses (SysML Requirements Diagram)

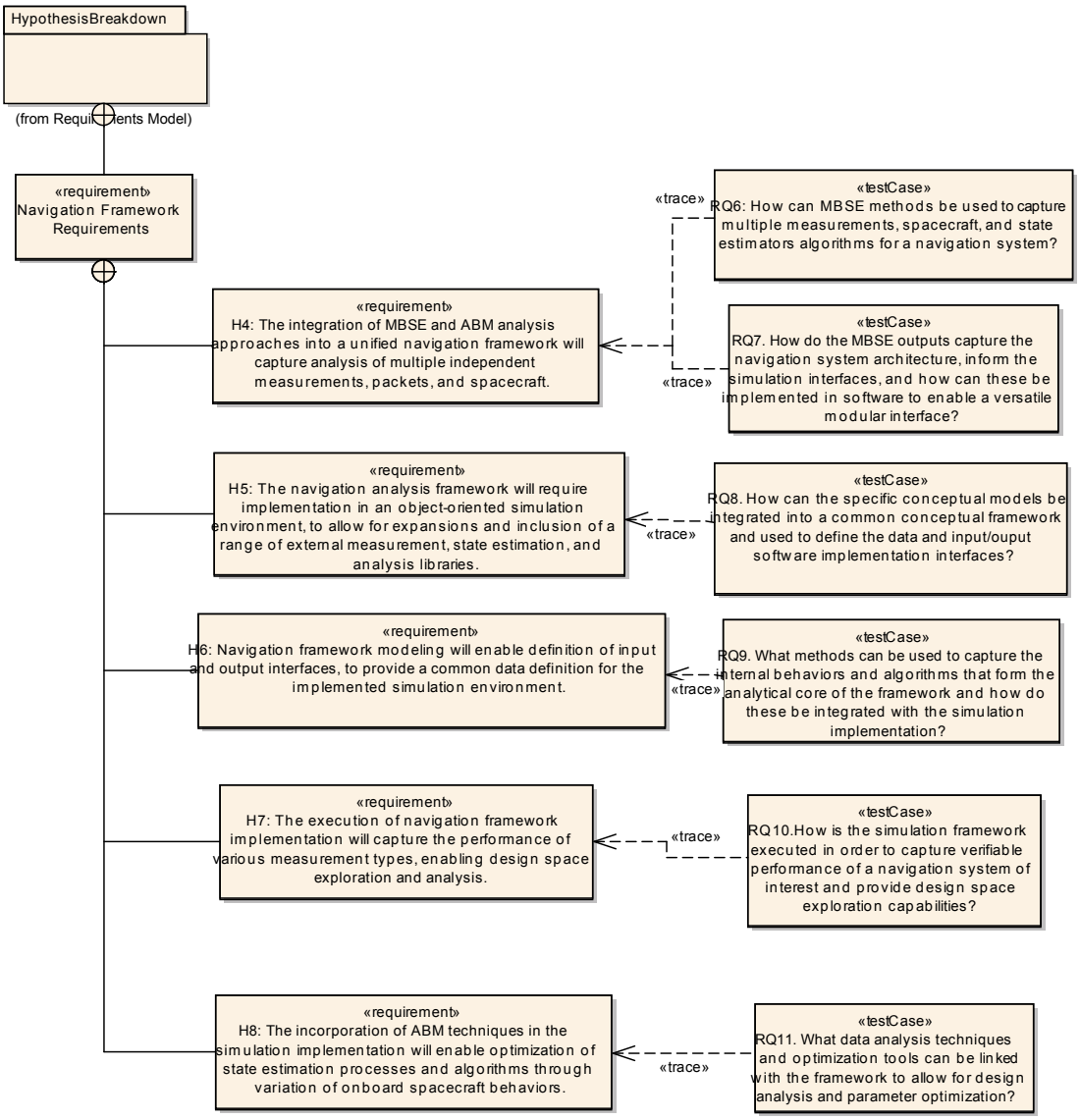


Figure 96: SNAPE Framework Hypotheses (SysML Requirements Diagram)

8.2 Requirements Development of NNAV

With NNAV described at a conceptual level, the next step was to analyze the **R**equirements of the navigation system architecture. This part of the process captured the organization and structure of NNAV as a conceptual navigation system. Model-Based Systems Engineering (MBSE) and Agent-Based Modeling (ABM)-architecture methods drove this part of the analysis through the development and use of SysML requirements diagrams to capture the

high level requirements within the conceptual framework of SNAPE. The continued application of model-based design allowed for the capture of the interactions between individual modules of the navigation system. This step supported identification and documentation of the navigation method being developed as well as the functionality to be implemented in order to perform the underlying simulation and performance evaluation.

With the initial requirements captured, the design of NNAV and the SNAPE framework was modeled using SysML tools. This allowed for description and analysis of the system composition, interactions, functions, sequence of events, and the conceptual framework structure. The implementation of the requirement, use case, state machine, and block diagrams are shown to successfully describe the navigation system under analysis. The use of these models provided a way to capture the behavior and structure of the various measurement methods and estimation algorithms being compared. This implementation additionally allowed the analysis of any deep space navigation system that can be considered to generate measurements of a vehicle's state, focusing on the capturing of high level requirements and interfaces. The developed models demonstrate how the system can be captured, which addresses **(RQ6)** and **(RQ7)**, providing the proof of **(H4)**.

8.3 Analysis and Modeling of SNAPE Framework

With the desired functions developed and analyzed, the next step was the detailed **A**nalysis and modeling of the SNAPE framework. The work performed in identifying the internal structure and operation of NNAV through the implemented models provided a baseline for the framework. The requirements, use cases, and operations of the analysis package were derived by starting with a generic navigation system. This linked the needs of the framework directly to the higher-level generic navigation system functions and provided traceability for the requirements at multiple levels of analysis. From these requirements, the operations, algorithms and structure were defined through the continued use of standard SysML modeling techniques. The developed models form the response to **(H4)** and were demonstrated through the analysis performed.

In addition to the high level functional requirements, these models were used to capture

the algorithms to address **(RQ1)**, **(RQ2)**, and **(RQ4)**. The first step in answering these questions is contained in the analysis and modeling of SNAPE, capturing both how the individual functions are performed as well as defining the interfaces between individual framework elements. Additionally these models served to capture the analytical algorithms through the description of the functional backend and physics-based models that drove the simulation and spacecraft operation. The defined interfaces to capture both data and functionality address **(RQ8)**, demonstrating the hypothesized capability of SNAPE from **(H6)**. The development of these models led directly into identification of the required system and object functionality and attributes.

8.4 Implementation of SNAPE Simulation Environment

The developed design and interface models of the SNAPE framework served as conceptual prototypes for the analytical simulation. This step of the research translated the developed sequence models and block diagrams into executable software. The models drove the implementation process, serving as a reference for the functional needs and the simulation architecture, linking to the high level navigation system requirements. This thesis presented one particular implementation of the SNAPE framework in order to support verification and validation. Additionally, this discussion demonstrates how the MBSE and SysML models feed into simulation and analysis.

The selection of Python as the programming language for implementation addresses **(RQ9)** and **(H5)**. SNAPE's design and implementation demonstrate the capability to load and use a series of external models into the simulation environment and integrate them with the defined analysis scenario. Additionally, the implementation of multiple measurement and packet models addresses **(H8)** and provides for performance evaluation and comparison of various navigation methods. The actual methods used to address these required capabilities are defined in the presented analytical models and systems of equations are described in detail in Chapters 4 and 5.

The development of the *SimLink* software interface directly allows for integration between multiple design and analysis tools. This interface transfers data to and from the

simulation framework. Through the use of this linkage, tools such as the DEAP evolutionary algorithm package and multiprocessing capabilities are integrated. With the built-in capabilities for design space exploration and visualization, these software modules capture the execution of the model and its integration with external tools, addressing **(RQ11)** which is derived from **(H8)**. The use of these modules additionally captures the design analysis functionality enabled by the tool, addressing **(RQ12)**, supporting **(H9)**.

***(H4)**. The integration of MBSE and ABM analysis approaches into a unified navigation framework will capture analysis of multiple independent measurements, packets, and spacecraft.* By following the analysis approach outlined above, the implemented models in Sections 5.3 and 5.5.1 demonstrate how these are captured in terms of the modeling framework and input specification. By developing a modular, expandable simulation with a well-defined interface, the SNAPE framework was developed to allow for a wide range of analysis scenarios. Chapter 6 captures these, particularly in its comparison of multiple measurement types from range to position to full state. Additionally, Chapter 7 demonstrates how the SNAPE implementation can be used to analyze various packet contents and operating frequencies. In the analysis of NNAV in Section 7.8.3, multiple spacecraft are included in the simulation. These analysis scenarios were all defined using the graphical user interface and all operate within the same simulation back-end. This capability for a wide range of scenarios and navigation sources is enabled by the model-driven simulation environment.

***(H5)**. The navigation analysis framework will require implementation in an object-oriented simulation environment, to allow for expansions and inclusion of a range of external measurement, state estimation, and analysis libraries.* This hypotheses further reinforces the needs for a modular interface and framework in order to develop a widely-applicable navigation simulation. The use of Model-Based Systems Engineering tools, specifically the use of SysML Block Definition Diagrams and specific Instance Diagrams, feeds directly into an object-oriented architecture. The capability to load software from multiple sources for use within the simulation is best supported through examination of the input deck definition given in Section 5.5.1. The input specification is built to allow for various library sources,

and the SimLink software interface as described in Section 5.5.2 has an embedded capability to load functions and software objects from multiple source files and analysis libraries. This is further supported by the implementation of the framework using the Python programming language. The development of the DEAPLink library, as described in Section 5.5.5, also demonstrates the capability of the object-oriented framework to perform external analysis, such as navigation packet optimization, due to its object-oriented nature, allowing for ease of integration and expansion.

***(H6)**. Navigation framework modeling will enable definition of input and output interfaces, to provide a common data definition for the implemented simulation environment.* In addition to the above hypotheses, this also addresses the use of Model-Based Systems Engineering processes to capture the interface between various systems, functions, and objects. This is particularly important in enabling efficient conversion of the framework from SysML models to executable simulation. By capturing these interfaces in the models, additional information is captured about individual object inputs, outputs, and scope. This provides the prototypes for the software development. Specifically for this analysis, the Internal Block Diagrams, such as Figure 22 in Section 5.3, captured the inputs of each block and providing a starting point for the software implementation, which is executed throughout Chapters 6 and 7. This data definition is extensively used through the interfaces developed in Sections 5.5.1 and 5.5.2.

***(H7)**. The execution of navigation framework implementation will captures the performance of various measurement types, enabling design space exploration and analysis.* Similar in scope to the above hypotheses, this focuses on the capability of the framework to model individual measurement and packet scenarios and then be able to perform optimization to ascertain the ideal transmission frequencies, data content, or delays. An overview of potential measurement and packet options is given in Section 4.5 and 4.7. The ability to provide navigation analysis for a wide range of scenarios and integration with Monte Carlo as well optimization tools allows for a wide range of functionality and applicability. Multiple measurement types are traded in Section 6.4, providing a basis for additional work. The

analyses demonstrated were performed using Monte Carlo analysis to capture the uncertainty in the design space, which is due to the stochastic nature of the measurement and state estimation process. The utilization of these external design tools enables a wide range of analysis capabilities, providing a thorough evaluation of navigation architectures.

***(H8)**. The incorporation of ABM techniques in the simulation implementation will enable optimization of state estimation processes and algorithms through variation of onboard spacecraft behaviors.* As Section 3.7.2 and 5.4 described, the framework modeling and implementation takes an ABM approach to the simulation design. This was chosen to address the needs described in Chapter 3, particularly in the capture of autonomous behavior of the spacecraft assets. In order to provide a flexible environment that can be expanded into hardware-in-the-loop simulation as well as distributed computing, the behaviors of the individual spacecraft must be captured independently.

The main aspects of vehicle operation under consideration include the processing of measurements into the state estimation algorithms. These are enabled and implemented by the AgentBody class as described in Section 5.4.5. Utilizing this development, it was possible to optimize both spacecraft behavior and state estimation parameters. The DEAP genetic algorithm library was implemented to support such analysis, as described in Section 5.5.5. The modular interfaces were utilized to allow for a straightforward integration of this external tool interface, and enabled robust selection of optimization variables, ranges, and generation of a the performance evaluation criteria. The use of ABM algorithms enabled this optimization through its capture of spacecraft behavior and capability.

From this discussion, the SNAPE framework is demonstrated to capture the analysis requirements of the navigation system. In order to apply the develop simulation environment to NNAV, the analysis requirements and needs described in Tables 6 - 10 were evaluated against the implemented capabilities. The summary of these requirements and references describing their implementation are reiterated in Table 28. This analysis functionality enabled and drove the navigation performance studies described in Chapters 6 and 7, specifically Section 6.5 and 7.7. Each of the defined software elements addresses the SNAPE framework functionality requirements as defined in Section 3.2. Each block has a

unique purpose in the simulation and either individually, or linked together, address the high level capabilities required of the analysis.

These functions of the SNAPE implementation address the research questions specified in Section 3.2. Development of the simulation environment addressed three primary research questions: ***(RQ1)***. *How can the NNAV architecture be analyzed to capture its navigation capability?,* ***(RQ2)***. *How do the NNAV onboard algorithms analyze the embedded navigation packets into a measurement that can be processed by onboard state estimation filters?, and* ***(RQ3)***. *How are NNAV packets and traditional measurements integrated into a common framework, both operationally and algorithmically?* In order to analyze navigation performance, the error-based performance metrics are described in Section 3.2, capturing the accuracy and error of the onboard state estimation algorithms. These measurements capture the performance due to the measurement accuracy and the state estimation algorithms. The analysis algorithms are described in the SysML models, such as Figure 27, and the operations of the simulation coordinator, defined in Section 5.4.2. By utilizing the Monte Carlo interface, with the input mission deck, SNAPE was used to capture the performance of deep space navigation systems.

8.5 Verification of SNAPE Implementation

In order to provide confidence in the SNAPE implementation and verify module functionality, specific analysis cases were executed in order to compare the generated outputs with other standard tools. To answer **(RQ10)**, several standard analysis packages were selected to verify SNAPE's functional performance. Comparison with STK results provided verification of the state propagation capability. ODTBX was used to capture state estimation performance of a sequential estimation algorithm. The documented results of both verification analyses provide evidence to support the capability and valid implementation of the analytical backend. These results support **(H8)** and **(H9)** by providing verification capability.

Chapter 6 presented verification and validation of the capability and functionality of the SNAPE framework. These validation test cases are summarized below in Table 29. The

Table 28: Implementation of SNAPE Framework Analysis Requirements

Framework Requirement	SNAPE Implementing Element(s)
1.1 Initialize body to input data	Simulation Coordinator
1.2 Compare estimated to reference/truth data	Simulation Coordinator, Data Collector
1.3 Calculate non-inertial forces on body	Force Model
1.4 Capture inertial forces on body	Force Model
1.5 Integrate body's onboard state	State Propagator
1.6 Modular measurement interface	SimLink, Coordinator, Measurements
1.7 Modular state estimation interface	SimLink, Coordinator, State Estimator
1.8 Calculate State Errors as a function of time	Simulation Coordinator, Data Collector
2.1 Perform Deep Space Link Analysis	Simulation Coordinator, Comm System, Agent Body, Truth Body
2.2 Autonomous Packet Generation	Agent Body, Packets, Coordinator
2.3 Capture Transmission Delays	Coordinator
2.4 Autonomous Reception and Processing of Packet	Agent Body, Coordinator
2.5 Integration of Packet with State Estimator	Agent Body, State Estimator
2.6 Onboard estimation of other SC states	Agent Body, Estimated Body
3.1 Interface to external measurement models	SimLink, Simulation Coordinator, Agent Body
3.2 Model Autonomous or Scheduled Measurements	Coordinator, Agent Body
3.3 Process measurement into state estimator	Agent Body, State Estimator, Simulation Coordinator
4.1 Model and vary packet content and measurements	SimLink, DEAPLink
4.2 Integration with Monte Carlo tools	SimLink, DEAPLink, Simulation Coordinator
4.3 Capability to perform packet optimization	DEAPLink
4.4 Modular interface to external design tools	DEAPLink, SimLink, Simulation Coordinator
5.1 Support a range of analysis scenarios	All Elements
5.2 Modular input and interface to test cases	User Interface, SimLink, Simulation Coordinator
5.3 Robust framework to variety of studies	All Elements

verification runs can be found in Table 25 in Chapter 6. Each analysis focused on a particular aspect of the analysis, and was used to demonstrate SNAPE's capability and integration with external design tools. These cases support the framework-specific hypotheses outlined above to demonstrate the functionality and capability of the implementation.

8.6 Evaluation of NNAV

A primary aspect of NNAV focuses on the algorithms that determine how the received time-tagged packets and navigation measurements are processed and integrated with the state estimation filter. This was performed using the modular definitions and interfaces of the measurement software prototypes. The integration of these measurements with the developed Extended Kalman Filter, from Section 4.4, is described in Sections 4.7, 5.4.7, 5.4.8, and 5.4.9. The input functions for each measurement are defined to capture the observed value (based on the truth state and defined sensor errors), the estimated value, as well as the derivative of the measured value with respect to the vehicle's state. Using this definition and these functional interfaces, this data is formatted to allow for integration into a wide variety of state estimation algorithms.

As mentioned, the capability of NNAV is based on the accuracy and stability of the onboard clock. It is also driven by the specific message content, as called out in **(RQ3)**. This research question was addressed in Section 7.5, analyzing various data included within the navigation header. Specifically, this analysis focused on the inclusion of a transmitter's best-known current state at the time of transmission. This data heavily influences the accuracy of the modeled measurements. By increasing the accuracy of the spacecraft's knowledge of the transmitting asset's state, external errors in the measurement predictions can be reduced. As shown in the results of this study, the inclusion of accurate data allowed for greater accuracy state estimation and highlighted the effect of the measurement prediction models and required data. Additional research captured the effect of packet update parameters and optimized the measurement timing statistics to determine the best frequency of updates.

With the SNAPE framework verified and demonstrated, it was shown to meet the

Table 29: Summary of SNAPE Validation Cases

Section	6.4.1 Effect of Initial Error	6.4.2 Effect of Measurement Error	6.4.3 Timing Behavior		6.4.4 Timing Sensitivity	6.4.5 Integrated Performance	6.4.6 Measurement Content			6.4.7 Parameter Sensitivity	6.5 Measurement Optimization		
Case	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
Mission/Reference	MSL Mars Cruise Trajectory												
Simulation Duration	25 days				50 days	25 days							
Primary Agent	MSL												
Onboard filter	EKF - 6 State		EKF - 8 State										
State Propagator	Dopri 5												
Other Agents	Earth-DSN												
Nav Packet Source	N/A												
Nav Packet Frequency													
Analysis Approach	Monte Carlo - Vary Initial Error, Time Between State Updates	Monte Carlo - Vary Position Error, Frequency	Monte Carlo - Initial Error		Monte Carlo - Clock Stability, Time Between State Updates	Monte Carlo - Position Measurement Accuracy and Frequency	Monte Carlo - Initial Error			Monte Carlo - Time between Measurements, and Time to First	Genetic Algorithm Optimization of Position Meas. Error and Frequency		
Packet Content	N/A												
Measurements	State - DSN	Position - DSN	Weekly State, Daily Position - DSN	Weekly State, Daily Position, Weekly Time - DSN	Weekly State, Variable Timing - DSN	Weekly State, Weekly Timing, Variable Position - DSN	Weekly or Daily Position - DSN	Weekly or Daily Range - DSN	Weekly or Daily State - DSN	Variable State, Position, Time - DSN	Weekly State, Variable Position DSN		Weekly Time, Weekly State, Variable Position - DSN

requirements for the navigation analysis functionality needed. SNAPE was then used to evaluate NNAV. This allows for addressing **(H1)-(H3)**, relating to the performance of the navigation architecture. Each of these will be discussed in terms of how they are addressed as well as their significance. The specific NNAV evaluation uses cases are given in Table 30.

(H1). *NNAV, which utilizes embedded navigation packets to enable state updates simultaneously with communication, is a viable method of deep space autonomous navigation.* As described in this thesis, the need for a modular, expandable, and robust navigation analysis tool drove the development the simulation framework and implementation. Chapter 7 captures the analysis scenario and input parameters utilized to analyze this concept. The feasibility of the approach is demonstrated in Section 31. Through the utilization of a packet include transmission state and time, the packet-derived measurements of range and range-rate can be used to navigation and track the state of the vehicle.

(H2). *Augmenting traditional navigation state update techniques with NNAV will provide improved onboard state estimation capability for deep space missions, especially with a limited network implementation.* In addition to providing navigation state updates on their own, this architecture can also be utilized with current deep space navigation system to provide for increased accuracy and frequency of state updates. This scenario is captured by looking at the integration of traditional DSN state updates with daily communications containing navigation packets. This test case is described and presented in Sections 7.1 and 7.8.3. From the analysis results, the inclusion of a high accuracy ground-based estimate helps to further reduce the onboard navigation capability, reducing errors and increasing the capability of the navigation packets to bound state estimation errors.

(H3). *NNAV will reduce the reliance on ground-based state updates and limits the growth of navigation errors between updates.* Through the utilization of NNAV, the spacecraft navigation system is able to operate with increased accuracy over times between ground-based updates. This test case is captured in Section 7.8.3. For this analysis, the time between ground-based updates was dispersed to capture the relationship between state update latency and state estimation errors. The analysis compared two approaches to navigation,

Table 30: Summary of NNAV Evaluation Cases

Section	7.5 Packet Content Study		7.6 Measurement Sensitivity	7.7 Packet Optimization	7.8 Comparison to Current Methods			
Case	N1	N2	N3	N4	N5	N6	N7	N8
Mission/Reference	MSL Mars Cruise Published Trajectory							
Simulation Duration	20 days							
Primary Agent	MSL							
Onboard filter	EKF-8 State							
Other Agents	Earth-DSN		MRO, Earth-DSN	MRO, Earth-DSN	Earth-DSN	MRO, Earth-DSN	MRO, Earth-DSN	MRO, Earth-DSN
Nav Packet Source	Earth-DSN		MRO	Earth-DSN	None	Earth-DSN, MRO	None	Earth-DSN
Nav Packet Frequency	Daily		Daily	Optimization Variable	N/A	Daily	N/A	Daily
Analysis Approach	Monte Carlo - Initial Error		Monte Carlo - Measurement Error	Genetic Algorithm	Monte Carlo - Initial Error	Monte Carlo - Initial Error	Monte Carlo - Time Between State Upates	
Packet Content	Transmission Time	Trans. Time and Location	Trans. Time and Location	Trans. Time and Location	N/A	Trans. Time and Location	N/A	Trans. Time and Location
Measurements	Weekly State - DSN		None	Weekly State- DSN	Weekly State- DSN	None	State Update - DSN	State Update - DSN

with and without the navigation packets. From the results displayed in Figures 90, 91, and 92, the benefit of utilizing navigation-based packets is clearly demonstrated through the reduction of navigation error due to inclusion of navigation packets.

For frequent ground-based state updates, the two options produce similar navigation results. This is due to the limited time of error propagation between state updates between the high accuracy ground-based measurements. As the time between these measurements increases, the improved capability of NNAV is increasingly apparent. This provides a direct demonstration of the reduction in reliance on Earth-based navigation support, though the architecture does have a large dependence on initial errors, which can be minimized through ground-based orbit determination over the mission. And while the navigation packets do provide improved accuracy over long delays between Earth updates, the error will still grow over time without the updates from Earth for this limited network of simply DSN and MRO. This is a limitation of the modeled navigation network. As the number of participating spacecraft increases and the algorithms improve to allow for simultaneous application of observations from multiple sources the navigation capability of the navigation system will also improve.

8.7 Hypotheses Overview

The preceding sections summarize how the analysis process was implemented in order to develop and implement SNAPE and its application to the NNAV architecture, specifically addressing the high-level hypotheses. Each hypothesis, with references to the sections of this work that address each is given in Table 31.

The steps of the analysis approach and their relationship to the declared hypotheses are summarized in Figure 97. This demonstrates the application of the analysis steps and their status in verifying the high level navigation requirements. The linkages between the functional analysis steps and the navigation framework demonstrate the capabilities of the approach and its relationship to the core performance requirements. These allowed for an analysis and modeling approach with a foundation in the required capture of the navigation

Table 31: Addressing of Hypotheses within Research

Hypothesis	Focused Sections
1. NNAV, which utilizes embedded navigation packets to enable state updates simultaneously with communication, is a viable method of deep space autonomous navigation.	7.1, 7.6,
2. Augmenting traditional navigation state update techniques with NNAV will provide improved onboard state estimation capability for deep space missions, especially with a limited network implementation.	7.1, 31, 7.8.3
3. NNAV will reduce the reliance on ground-based state updates and limits the growth of navigation errors between updates.	7.1, 7.8.3
4. The integration of MBSE and ABM analysis approaches into a unified navigation framework will capture analysis of multiple independent measurements, packets, and spacecraft.	5.3, 5.5.1, 6, 7, 7.8.3
5. The navigation analysis framework will require implementation in an object-oriented simulation environment, to allow for expansions and inclusion of a range of external measurement, state estimation, and analysis libraries.	5.5.1, 5.5.2, 5.5.5
6. Navigation framework modeling will enable definition of input and output interfaces, to provide a common data definition for the implemented simulation environment.	5.5.1, 5.5.2, 5.3, 6, 7
7. The execution of navigation framework implementation will captures the performance of various measurement types, enabling design space exploration and analysis.	4.5, 4.7, 6.4
8. The incorporation of ABM techniques in the simulation implementation will enable optimization of state estimation processes and algorithms through variation of onboard spacecraft behaviors.	3.7.2, 5.4, 5.4.5, 5.5.5

system under study (NNAV), by continually linking the developed tools to the identified needs.

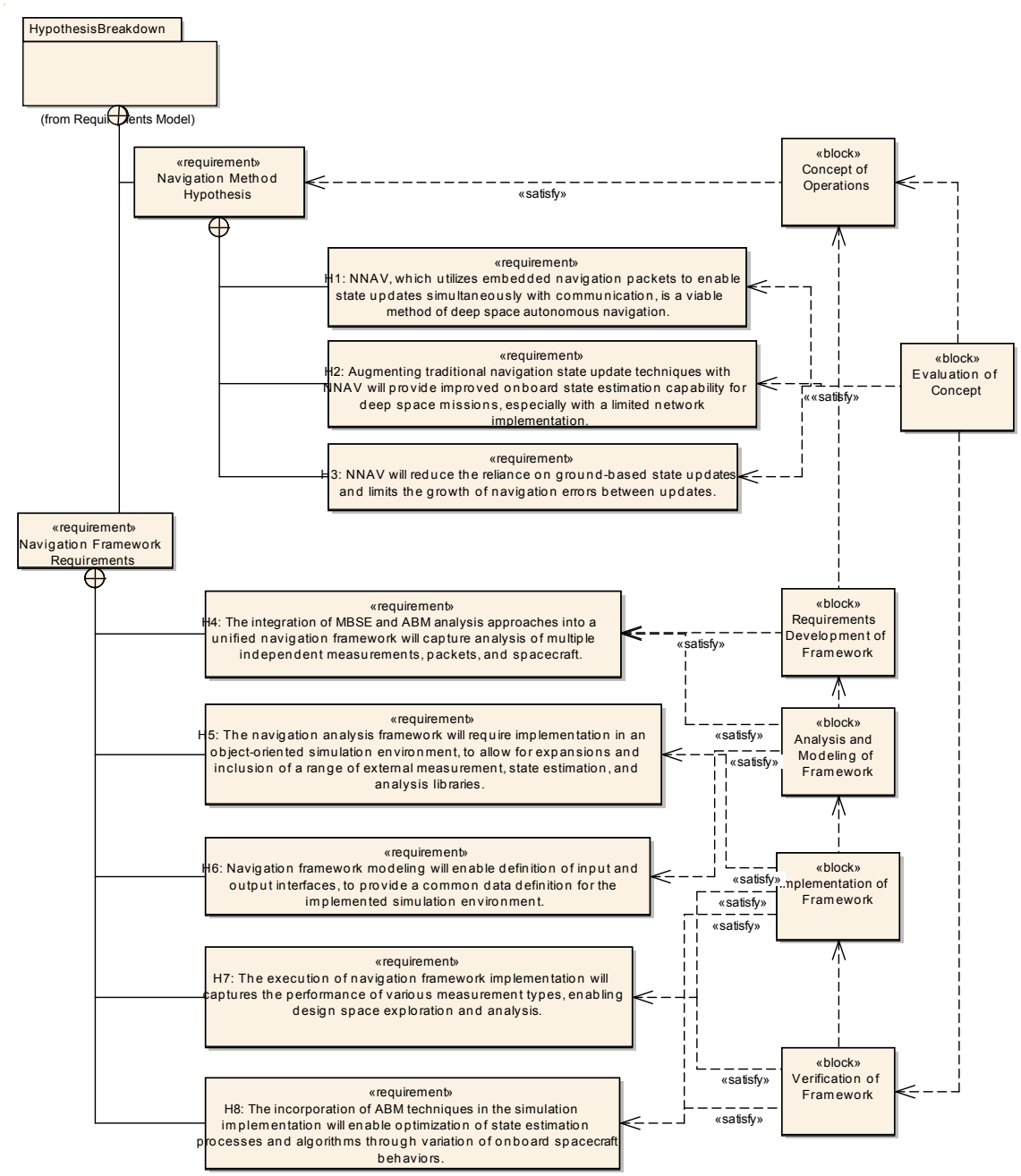


Figure 97: Linking Analysis Process to NNAV Hypotheses (SysML Requirements Diagram)

8.8 Summary of Contributions

The overarching contributions of this dissertation are the development of the SNAPE framework and the NNAV architecture. Each is described in detail below, both to summarize the contribution and the significance of each. The future application of each will also be discussed.

8.8.1 The SNAPE Framework

Validation of NNAV is enabled by the SNAPE framework development and simulation implementation. This provides a method for integrating formal modeling techniques with the implementation of a modular simulation environment to allow for the analysis of a wide range of space navigation system, both at a conceptual and analytical level. Although the current development is focused on analysis of a packet-based architecture with a Martian transfer use-case, SNAPE can be applied to a wide range of scenarios. Due to the modularity of the simulation architecture and generalized nature of the implemented analysis functions, developing an alternate test case can be done by loading additional trajectory data, and building spacecraft agents to reference the additional data. The current measurement models and packet simulation tools were developed to operate independent of the analysis case, allowing applicability to a range of mission scenarios.

The modeling approach captures the requirements of a generic navigation system through the integration of Model-Based Systems Engineering and Agent-Based Modeling techniques. This is achieved through the use of SysML in description, analysis, and implementation models. This provided a formal framework for the capture of the navigation system requirements, operations, and structure. SNAPE provided a direct linkage of the needs of spacecraft navigation to the needs of the simulation environment. Figure 98 demonstrates the connections between the navigation system design and the framework design. This traceability allowed for efficient capture of the navigation system, aiding in simulation implementation. The use of these tools facilitated efficient software development and provided forward development of psuedo-code, object definitions, and overall simulation architecture.

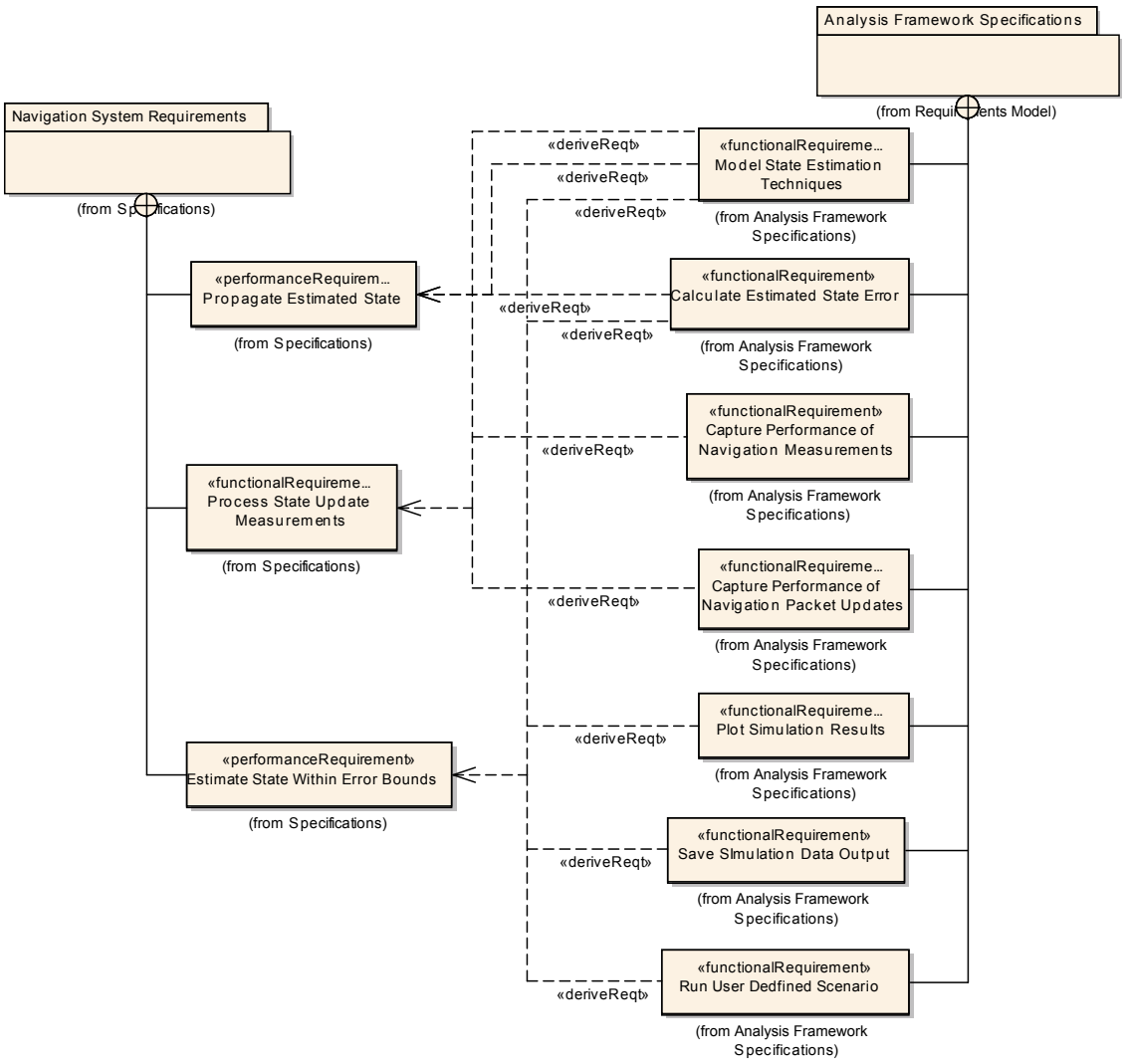


Figure 98: Navigation System Linkage to Framework Requirements (SysML Requirements Diagram)

This approach is in stark contrast to traditional navigation design and analysis, in which the focus is purely on the underlying physics and capturing the capability of a specific state estimation algorithm, measurement technique, and trajectory. Typical analyses focus on the problem at hand and take advantage of organization or mission internal tools. This limits collaboration and the development of a standard deep space navigation toolset to be used in mission navigation design. By following a formal approach to deep space navigation design, this thesis demonstrates a widely applicable navigation framework by means of a library

of standard models that capture both the navigation architecture and simulation design. These models help to share assumptions and the underlying design of the navigation system. Their use allows for increased sharing of knowledge, collaboration, and understanding of the analysis. Using SNAPE, it is simple to load a new trajectory or mission and compare and contrast the performance of multiple navigation systems. By providing libraries of deep space navigation filters and measurements models, it is possible to perform requirements analysis studies to capture the navigation needs of a particular mission. The results can be used to drive vehicle and mission design early in conceptual analysis.

An example test case of this is described in the development of a deep space mission to Jupiter. By performing trajectory design and optimization, a design ephemeris can input to the SNAPE implementation. Various navigation architectures can then be analyzed by selecting vehicle navigation capability and measurement sources. The user may then begin to identify the capabilities of a navigation filter and its estimation performance in terms of capturing the true dynamic state of the vehicle (inertial position and velocity for example) to a range of inputs. Additional studies may then be performed to trade navigation measurements during specific legs of the mission comparing state versus range and range-rate observations for example. Furthermore, it is possible to include optical navigation models and be able to capture the state estimation capability over the trajectory. These studies may also push the spacecraft towards having increased computational capability to handle advanced filtering techniques. This allows for modeling and analysis of the navigation performance over the course of the entire mission and provides insight into the various options available. This feeds directly into the mission architecture and development, allowing for detailed trades of navigation capability early in the design process.

8.8.2 Network-Based Navigation (NNAV)

From inspirations in radiometric ranging and range-rate measurements, combined with the continued growth and development of deep space communication networks, NNAV provides an integrated approach to deep space navigation. The integration of navigation packets within the communication protocol provides a new way to enable deep space navigation.

This allows for autonomous navigation onboard the spacecraft, in that it does not require ground assets to provide a computed navigation state. Incorporating these packets into the space communication architecture allows for increasing performance due to the additional navigation hosts. This development also allows for increased navigation accuracy out into the solar system to counteract the decreased accuracy of Earth-based measurements at large distances from Earth.

This thesis demonstrates the capability of NNAV, specifically analyzing the capability of using the navigation packets to augment traditional spacecraft state updates. This test case also captures the implementation path of these protocols. Initial missions would use these in addition to current Earth-based orbit determination methods. As the algorithms are proven and advanced, the navigation performance will be demonstrated by reducing updates from Earth-based assets and analyzing the spacecraft's navigation errors. It can also be used as a proof of concept of the capability to implement these protocols by means of a software update. This analysis case therefore directly maps to an initial implementation and flight testing of NNAV.

The analysis results demonstrate the feasibility of the packet-based navigation architecture and provide a basis for continued studies of design sensitivities to both advanced state estimation techniques and additional stochastic dynamic modeling effects. The performance of this navigation architecture improves with increases in the number and distribution of assets and spacecraft in the network. As the in-space communication infrastructure continues to grow with the development of multi-purpose relay satellites, such as MRO, the frequency of inter-spacecraft communication also increases. This growing network allows for more frequent, broader navigation measurements, improving performance, capability, and range.

Though the initial evaluation is limited to small number of assets, the research can be expanded to include a growing number of assets and spacecraft. As the number of elements participating in the network increases, individuals will have more both more opportunities for state measurement and increased accuracy due to the dispersion and number of external navigation packet sources. Additionally, NNAV is robust to hardware implementation,

independent of transmission medium. As long as the communication architecture allows for measurements of time of flight and the embedding of navigation packets within the digital data, NNAV can continue to be used. This provides a robust and expandable navigation architecture to both support and eventually supplant traditional navigation sources.

8.9 Future Work

This research has developed SNAPE for the design and analysis of deep space navigation systems. It provides implementations of the functionality required to propagate trajectories, calculate state measurements, and model the transmission of packets between assets. Additionally, the framework has been implemented with modular well-defined interfaces to allow for the integration of additional force models, telemetry data, measurement models for specific sensors, and most importantly navigation estimation algorithms. The implementation of these interfaces allows for the integration and analysis for a wide range of analysis case.

By implementing additional physics-based observation and measurement models, such as optical or X-Ray navigation, this framework can increase in capability and be used to optimize measurement types for a given mission. Improvements in the dynamics models, such as implementing higher order gravity models and relativistic effects, will further improve the capability to utilize this framework on trajectories under design. With the inclusion of these models, additional optimization methods may be integrated to allow for integrated mission analysis and design of the navigation algorithms.

To provide additional capability of the navigation system under a stochastic measurement environment, improvements can additionally be made to the analysis process to capture the variability of the performance, using one of two main methods. One approach is to adapt the LINCOV[49] analysis approach to this communication-centric navigation design, and compare its estimates to simulation results. Another method of analysis is to integrate Markov Chain Monte Carlo techniques into the framework. This allows for capture of the estimated performance at each step of the simulation.

With the demonstrated capability of NNAV for state estimation augmentation, the research effort shifts to focus on the actual implementation of the method. This involves developing interfaces to the actual communication modules, to directly capture and model delays between packet arrival and time detection. This will allow for increased fidelity analysis of the navigation capability. With this added capability, the analysis can begin to move towards a hardware-in-the-loop simulation technique with the replacement of internal software agents with external hardware agents. This allows for direct analogy to an actual space implementation and allows for continued development of integration algorithms and communications modeling. Improved modeling begins to integrate functionality of the actual protocols, whether CCSDS or LTP, similar to that of the space protocol emulators. Additionally, these experiments can begin to use space-like oscillators and timing sources to continue to increase the fidelity of the communications and clock modeling.

With the continued increases of the simulation environment's fidelity and functionality, similar improvements are required in the onboard spacecraft's estimated dynamics. This includes higher fidelity dynamic and gravity models, such as general relativistic corrections[78] and improved state propagation techniques such as the Runge-Kutta-Fehlberg method[36]. Additional improvements in the onboard state estimation filter can be achieved by implementing advanced Particle Filtering [23] or information filters[50] [108].

An additional factor not considered by SNAPE is consideration of the costs involved, both in terms of technology development and implementation. It is envisioned that NNAV involves minimal cost in comparison to the other identified methods, due to its basis as a software component of the communication. The primary costs involved are in development of flight algorithms, the software update procedure, and certification of the library for use on orbit. Additional work must be done as to the onboard processing capability required for these state estimation algorithms to ensure their potential implementation on legacy spacecraft. Further research into cost estimation techniques and their specific application to navigation systems development supports complete analysis and trade-offs between cost and performance. Integration of these elements allows the systems architect to trade performance, development, and implementation costs in one unified framework. This enables

identification and estimation of these costs at an early stage of the lifecycle, providing additional insight into the navigation system of interest. Capturing the total investment required in order to develop and implement a navigation solution is very important to minimize costs and is an increasingly important parameter in mission selection and implementation.

These represent many potential paths forward for the development of NNAV and SNAPE, and demonstrate that work across a variety of disciplines could be incorporated to improve their capability and functionality. The improvements in both hardware and software simulation and analysis can be directly applied to continue development. NNAV can also be applied to a variety of mission scenarios, with the nearest term application to Lunar and Martian missions. These scenarios provide an early showcase for demonstration missions due to existing communication infrastructure (Mars) and close locale (Moon). Its implementation across multiple spacecraft can provide an augmentation capability to traditional navigation methods. This can additionally be important for missions requiring low latency state updates for approaches to asteroids, or planetary entry, providing for a one-way similar ranging capability through the use of advanced onboard navigation filters.

The SNAPE framework implementation serves as an initial validation of the concept. As advanced modeling and simulation functionality is implemented, improving the fidelity of the analysis, NNAV can continually be analyzed to capture its performance and sensitivity to those improved aspects. These advanced dynamic models can also be used to drive hardware-in-the-loop simulations that begin by implementing hardware to capture specific instruments, such as space qualified timing sources and oscillators, to further improve the fidelity of the measurement system. With the shift to hardware-based testing, it is also possible to take advantage of space protocol emulators and communication processing software to begin to develop the actual implementation of the navigation packet into current standard CCSDS protocols. Upon verification of this method at a hardware level, it will be ready for flight testing. Dual avenues exist for demonstration of this concept. The first is to use current space-based software defined radios such as the SCAN payload on the International Space Station [1] to act as the autonomous vehicle, and compare the onboard

[1] http://spaceflightsystems.grc.nasa.gov/SOPO/SCO/SCaNTestbed/

estimated state to that calculated by GPS or other methods. This will additionally require a ground station to act as the navigation host, as well as the inclusion of accurate atmospheric models to capture ground-centric transmission delays. Alternate implementation routes involve integration with an alternate mission as a science payload. This will provide a better estimate of true flight performance due to the deep space trajectory, and allow direct comparison to current methods. With the correct software links, this NNAV hardware implementation can demonstrate the capability for navigation augmentation and reduce spacecraft ground operations support costs by augmenting and reducing the need for frequent ground-based orbit determination analysis.

8.10 Closing Comments

In conclusion, this dissertation presents the SNAPE framework for the analysis of deep space navigation architectures. Through the combination of MBSE and ABM principles, the SNAPE framework was designed, modeled, and implemented. This was developed to enable analysis of a range of deep space navigation mission scenarios, measurements sources, and state estimation algorithms. An example optimization toolset was also integrated into the framework to supplement built-in Monte Carlo analysis capability to allow for a range of design exploration and trade studies. The interfaces to these tools were integrated into a graphical front-end whose design followed the steps of the analysis method to allow for scenario definition, data collection, simulation control, and data processing. The SNAPE implementation in Python was verified against standard tools and validated using generic measurement models to capture the performance of the implemented estimator to variable state observations and uncertainty.

This dissertation also develops and presents the Network-Based Navigation (NNAV) concept. NNAV is evaluated against current navigation methods through the SNAPE framework. The development demonstrated the capability of SNAPE to feed requirements from coneptual navigation system design and structure to implementation. Additionally, the evaluation of NNAV using SNAPE demonstrates its functional and analytical capability. Through this framework, the NNAV packet structure and operations were optimized and

compared to traditional state measurements to show the performance gains in state estimation achieved. This was shown in the ability to maintain state estimation accuracy with a reduction in ground navigation support for a limited network. This analysis provides a strong foundation for further navigation studies and detailed hardware-in-the-loop simulation. With its modular design, the SNAPE framework is capable of analyzing a wide range of NNAV mission scenarios, which will continue to demonstrate NNAV's increasing capability and performance across an expanding interplanetary communication infrastructure.

REFERENCES

[1] "Umbra modeling and simulation framework." http://umbra.sandia.gov/pdfs/umbra.pdf.

[2] Acton, C., "Ancillary data services of nasa's navigation and ancillary information facility," *Planetary and Space Science*, vol. 44, no. 1, pp. 65–70, 1996.

[3] Allan, D., "Time and frequency (time-domain) characterization, estimation, and prediction of precision clocks and oscillators," *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, vol. 34, pp. 647 –654, Nov. 1987.

[4] Bajaj, M., Scott, A., Deming, D., Wickstrom, G., DeSpain, M., Zwemer, D., and Peak, R., "Maestro a model-based systems engineering environment for complex electronic systems.," in *22nd Annual INCOSE International Symposium*, (Rome, Italy), INCOSE, July 2012.

[5] Bajaj, M., Zwemer, D., Peak, R., Phung, A., Scott, A., and Miyako, W., "Satellites to supply chains, energy to finance slim for model-based systems engineering, part 1: Motivation and concept of slim.," in *21st Annual INCOSE International Symposium*, (Denver, CO), INCOSE, June 2011.

[6] Bajaj, M., Zwemer, D., Peak, R., Phung, A., Scott, A., and Miyako, W., "Satellites to supply chains, energy to finance slim for model-based systems engineering, part 2: Applications of slim.," in *21st Annual INCOSE International Symposium*, (Denver, CO), INCOSE, June 2011.

[7] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*. Dover Publications, 1971.

[8] Bechtold, K. E., Bucior, S. E., Sepan, R. L., and Matsumoto, S., "Operations challenges for missions with significant round-trip light times," in *SpaceOps 2010 Conference*, (Huntsville, AL), April 2010.

[9] Bhasin, K. B., Barritt, B., Matthews, S., and Eddy, W., "Integrated approach to architecting, modeling, and simulation of complex space communication networks," in *SpaceOps 2010 Conference*, (Huntsville, Al), April 2010.

[10] Bhaskaran, S., Riedel, J., Kennedy, B., and Wang, T., "Navigation of the deep space 1 spacecraft at borrelly," in *AIAA AAS Astrodynamics Specialist Conference and Exhibit*, (Monterey, CA), AAS/AIAA, August 2002.

[11] Bhaskaran, S., Riedel, J., Synnott, S., and Wang, T., "The deep space 1 autonomous navigation system: A post-flight analysis," in *Astrodynamics Specialist Conference*, AIAA. AIAA-2000-3935.

[12] Biswas, A. and Piazzolla, S., "Ipn progress report 42-154: Deep-space optical communications downlink budget from mars: System parameters," tech. rep., NASA/Jet Propulsion Laboratory, August 2003.

[13] Brown, R. and P.Y.C., H., *Introduction to Random Signals and Applied Kalman Filtering*. New York: Wiley, 2nd ed., 1992.

[14] Burleigh, S., Cerf, V., Durst, R., Fall, K., Hooke, A., Scott, K., and Weiss, H., "The interplanetary internet: A communications infrastructure for mars exploration," in *53rd International Astronautical Congress*, (Houston, TX), International Astronautical Federation, October 2002.

[15] Burleigh, S., Cerf, V., Durst, R., Fall, K., Hooke, A., Scott, K., Torgerson, L., and Weiss, H., "Interplanetary internet." presentation, www.ipnsig.org/reports/IPN-04Mar03-IPNSIG.pdf, March 2003.

[16] C., D., E., F., Lam, D., and Lee, C., "Model based document and report generation for systems engineering," in *2013 IEEE Aerospace Conference*, March 2013.

[17] Carpenter, R. and Lee, T., "A stable clock error model using coupled first and second order gauss-markov processes," in *2008 AAS/AIAA Space Flight Mechanics Meeting*, 2008.

[18] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and Weiss, H., "Delay-tolerant network architecture: The evolving interplanetary internet," tech. rep., Interplanetary Network Research Group, Internet Research Task Force, August 2002.

[19] Chory, M. A., Hoffman, D. P., Major, C. S., and Spector, V. A., "Autonomous navigation-where we are in 1984," *AIAA*, 1984.

[20] Christian, J. A. and Lightsey, E. G., "Review of options for autonomous cislunar navigation," *Journal of Spacecraft and Rockets*, vol. 46, September-October 2009.

[21] Collier, N., "Repast: An extensible framework for agent simulation," *The University of Chicagos Social Science Research*, vol. 36, 2003.

[22] Corporation, T., "Flames framework architecture." http://www.ternion.com/framework-architecture/, 2013.

[23] Crassidis, J. and Junkins, J., *Optimal Estimation of Dynamic Systems*. Chapman & Hall/Crc Applied Mathematics and Nonlinear Science Series, Taylor & Francis Group, 2011.

[24] Curkendall, D. W., "Navigation system design for the mariner jupiter/saturn mission," in *AIAA Guidance and Control Conference*, (Key Biscayne, FL), AIAA, August 1973.

[25] DeBolt, R., Duven, D., and Haskins, C., "A regenerative psuedonoise range tracking system for the new horizons spacecraft," in *Institute of Navigation 61st Annual Conference*, (Cambridge, MA), pp. 487–497, June 2005.

[26] DeKoning, H. P., "Incose model based systems engineering grand challenge. proposed contribution of the space systems working group," May 2007.

[27] Dieter, G., *Engineering Design: A Materials and Processing Approach*. McGraw-Hill Series in Mechanical Engineering, McGraw-Hill, 2000.

[28] Dunham, J., Long, A., and Sielski, H., "Onboard orbit estimation with tracking and data relay satellite system data," *Journal of Guidance*, vol. 6, pp. 292–301, July-August 1983.

[29] Dunham, J. and Teles, J., "One-way return-link doppler navigation with the tracking and data relay satellite system (tdrss): The ultra-stable oscillato (uso) experiment on the cosmic background explorer," *AIAA*, 1990.

[30] Emadzadeh, A. A. and Speyer, J., *Navigation in Space by X-ray Pulsars*. Springer, 2011.

[31] Endres, S., Griffith, M., and Behnam, M., "Space based internet network emulation for deep space mission applications," *AIAA*, 20XX.

[32] Estefan, J., "Survey of model-based systems engineering methodologies." INCOSE MBSE Focus Group Document, May 2007.

[33] Estefan, J. A., Pollmeier, V. M., and Thurman, S. W., "Precision x-band doppler and ranging navigation for current and future mars exploration missions," *Advances in the Astronautical Sciences*, 1993.

[34] Fang, B., "Satellite-to-satellite tracking orbit determination," in *AIAA, Aerospace Sciences Meeting*, vol. 1, 1978.

[35] Farrel, S. and Cahill, V., *Delay- and Disruption-Tolerant Networking*. Artech House, 2006.

[36] Fehlberg, E., "Classical fifth-, sixth-, seventh-, and eighth-order runge-kutta formulas with stepsize control," tech. rep., NASA TR-287, 1968.

[37] Fisher, J., "Model-based systems engineering: A new paradigm," *INCOSE Insight*, vol. 1, no. 3, 1998.

[38] Flanders, J. H., Fraser, D. C., and Lawson, J. R., "Technology for guidance and navigation of unmanned deep space missions in the 1970's," in *AIAA Annual Meeting and Technical Display*, (Philadelphia, PA), AIAA, October 1968.

[39] Folkner, W. M., Williams, J. G., and Boggs, D. H., "The planetary and lunar ephemeris de 421," tech. rep., Jet Propulsion Laboratory, 2008. JPL Memorandum IOM 343R-08-003 31.

[40] for Space Data Systems, C. C., "Ccsds report concerning the proximity-1 space link protocol," August 2007.

[41] for Space Data Systems, C. C., "Ccsds report: Overview of space communication protocols," December 2007.

[42] for Space Data Systems, C. C., "Ccsds recommendation for space packet protocol," September 2012.

[43] Fortin, F.-A., Rainville, F.-M. D., Gardner, M.-A., Parizeau, M., and Gagné, C., "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.

[44] Franklin, S. F., John P. Slonski, J., Kerridge, S., Noreen, G., Riedel, J. E., Komarek, T., Stosic, D., Racho, C., Edwards, B., and Boroson, D., "The 2009 mars telecom orbiter mission," *IEEE AC*, October 2004.

[45] Frauenholz, R. B., Bhat, R. S., Chesley, S. R., Mastrodemos, N., Jr., W. M. O., and Ryne, M. S., "Deep impact navigation system performance," *Journal of Spacecraft and Rockets*, vol. 45, January-February 2008.

[46] Friedenthal, S., Moore, A., and Steiner, R., *A Practical Guide to SysML: The Systems Modeling Language*. The MK/OMG Press, Elsevier Science, 2011.

[47] Garcia, H. A. and Owen, W. J., "Design and analysis of a space sextant for high altitude navigation," *Journal of Spacecraft*, vol. 13, December 1976.

[48] Gelb, A., *Applied Optimal Estimation*. Mit Press, 1974.

[49] Geller, D. K., "Linear covariance techniques for orbital rendezvous analysis and autonomous onboard mission planning," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 6, pp. 1404–1414, 2006.

[50] Gibbs, B., *Advanced Kalman Filtering, Least-Squares and Modeling: A Practical Handbook*. Wiley, 2011.

[51] Gifford, K. K., Jenkins, A., and Kuzminksy, S., "Dtn experiments and activities onboard the international space station." CCSDS Spring Meetings presentation, May 2010.

[52] Gramling, C., Hornstein, R., Long, A., and Samii, M., "Tdrss onboard navigation systems(tons) experiment for the explorer platform(ep)," *AIAA*, 1990.

[53] Grammier, R. S., "A look inside the juno mission to jupiter," *IEEE*, 2009.

[54] Graven, P., Collins, J., Sheikh, S., Hanson, J., Ray, P., and Wood, K., "Xnav for deep space navigation," in *31st Annual AAS Guidance and Control Conference*, American Astronautical Society, February 2008.

[55] Groves, P. D., *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. Artech House, 2008.

[56] Hairer, E., Norsett, S., and Wanner, G., *Solving Ordinary Differential Equations i. Nonstiff Problems*. Springer Series in Computational Mathematics, Springer-Verlag, 2nd ed., 1993.

[57] Hanson, J. E., *Principles of X-ray Navigation*. PhD thesis, Stanford University, March 1996.

[58] Hemmati, H., ed., *Deep Space Optical Communications (JPL Deep-Space Communications and Navigation Series*. Wiley-Interscience, 2006.

[59] Hirt, C., Claessens, S., Kuhn, M., and Featherstone, W., "Kilometer-resolution gravity field of mars:mgm2011." http://geodesy.curtin.edu.au/research/models/mgm2011/, 2011.

[60] Hoffman-Wellenhof, B., Legat, K., Wieser, M., and Lichtenegger, H., *Navigation: Principles of Positioning and Guidance*. Springer, October 2003.

[61] Hunter, J. D., "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[62] Im, E., Thomson, M., Pearson, J. C., and Lin, J., "Prospects of large deployable reflector antennas for a new generation of geostationary doppler weather radio satellites," in *AIAA Space 2007 Conference and Exposition*, (Long Beach, CA), AIAA, September 2007.

[63] INCOSE, "Systems engineering vision 2020," tech. rep., INCOSE, September 2007. TP-2004-004-02.

[64] Ivancic, W., "Experience with delay-tolerant networking from orbit," in *Advanced Satellite Mobile Systems Conference*, 2008.

[65] Jones, D., "Spacecraft navigation with large arrays of small antennas," in *American Astronomical Society Meeting Abstracts #208*, vol. 38 of *Bulletin of the American Astronomical Society*, p. 137, June 2006.

[66] Jones, E., Oliphant, T., Peterson, P., and others, "Scipy: Open source scientific tools for python."

[67] Jones, R. M., "Deep space networking experiments on the epoxi spacecraft," in *Infotech@Aerospace 2011*, March 2011.

[68] Kapurch, S., ed., *NASA Systems Engineering Handbook*. National Aeronautics and Space Administration, 2007.

[69] Karban, R., Zamparelli, M., Bauvir, B., Koehler, B., Noethe, L., and Balestra, A., "Exploring model based engineering for large telescopes- getting started with descriptive models," 2008.

[70] Kordon, M. and Wall, S., "Model-based engineering design pilots at jpl," in *2007 IEEE Aerospace Conference*, March 2007.

[71] Kordon, M. and Wood, E., "Multi-mission space vehicle subsystem analysis tools," in *IEEE Aerospace Conference Proceedings*, March 2003.

[72] Kremer, A. S., "Linear covariance analysis trade study of autonomous navigation schemes for cislunar missions," Master's thesis, Massachusetts Institute of Technology, June 2007.

[73] Larson, W. J. and Wertz, J. R., eds., *Space Mission Analysis and Design*. Microcosm, 3 ed., October 1999.

[74] Lightsey, E. G., Mogensen, A. E., Burkhart, P. D., Ely, T. A., and Duncan, C., "Real-time navigation for mars missions using the mars network," *Journal of Spacecraft and Rockets*, vol. 45, May-June 2008.

[75] Makovsky, A., Ilott, P., and Taylor, J., *Mars Science Laboratory Telecommunications System Design.* Deep Space Communications and Navigation Systems Center of Excellence Design and Performance Summary Series, Jet Propulsion Laboratory, California Institute of Technology, November 2009.

[76] McCarthy, T., Campbell, T., and Moseley, P., "A generic common simulation framework based starting point for missile 6dof simulations," in *Proc. AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2005.

[77] Miller, J., Stanbridge, D., and Williams, B., "New horizons pluto approach navigation," *Advances in the Astronautical Sciences*, vol. 119, no. 1, 2004.

[78] Moyer, T. E., *Formulation for Observed and Computed Values of Deep Space Network Data Types for Navigation.* Jet Propulsion Laboratory, California Institute of Technology, October 2000.

[79] NAIF, N. J., "Time routines in cspice." User's Guide, http://naif.jpl.nasa.gov/pub/naif/toolkit_docs/C/req/time.html, April 2009.

[80] Nelson, R. A., *Key Issues for Navigation and Time Dissemination in NASA's Space Exploration Program.* NASA, 2006.

[81] Nichols, K., Holbrook, M., Pitts, R. L., Gifford, K. K., Jenkins, A., and Kuzminsky, S., "Dtn implementation and utilization options on the international space station," in *SpaceOps 2010 Conference*, April 2010.

[82] North, M. J. and Macal, C. M., *Managing Business Complexity : Discovering Strategic Solutions with Agent-Based Modeling and Simulation: Discovering Strategic Solutions with Agent-Based Modeling and Simulation.* Oxford University Press, USA, 2007.

[83] O'Connor, C., Mehiel, E., and Butler, B., "Horizon 2.1: A space system simulation framework," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2008.

[84] Oliphant, T. E., "Guide to numpy." http://www.tramy.us/numpybook.pdf, Dec. 2006.

[85] OMG, "Omg mof 2 xmi mapping specification," Tech. Rep. formal/2011-08-09, Object Management Group, August 2011. http://www.omg.org/spec/XMI/Current/.

[86] OMG, "Omg systems modeling language version 1.3," Tech. Rep. formal/2012-06-01, Object Management Group, June 2012. http://www.omg.org/spec/SysML/Current/.

[87] Parkinson, B. and Spiker, J., *The Global Positioning System: Theory and Application.* No. v. 1 in Progress in Astronautics and Aeronautics Series, Amer Inst of Aeronautics &, 1996.

[88] Parsons, D., Rashid, A., Speck, A., and Telea, A., "A framework for object oriented frameworks design," in *Technology of Object-Oriented Languages and Systems, 1999. Proceedings of*, pp. 141–151, 1999.

[89] Pavlis, N., Holmes, S., Kenyon, S., and Factor, J., "An earth gravitational model to degree 2160: Egm2008," in *2008 General Assembly of the European Geosciences Union*, (Vienna, Austria), April 2008.

[90] Pearce, P. and Hause, M., "Iso-15288, oosem and model-based submarine design," in *Proceedings from the SESA, ITEA and INCOSE Region VI Systems Engineering Test and Evaluation Conference, in conjunction with the 6th Asia Pacific Conference on Systems Engineering*, May 2012.

[91] Pirondini, F. and Fernandez, A. J., "A new approach to the design of navigation constellations around mars: The marco polo evolutionary system," in *International Astronautical Congress 2006*, (Valencia, Spain), International Astronautical Federation, October 2006.

[92] Prestage, J. D., "Next generation space atomic clock," tech. rep., Stanford 2011 PNT Challenges and Opportunities, 2011. Nov.

[93] Psiaki, M. L., "Absolute orbit and gravity determination using relative position measurements between two satellites," *Journal of Guidance, Control and Dynamics*, vol. 34, no. 5, pp. 1285–1297, 2011.

[94] Rappin, N. and Dunn, R., *WxPython in Action*. Manning Pubs Co Series, Manning Publications, 2006.

[95] Riechle, D., *Framework Design: A Role Modeling Approach*. PhD thesis, ETH Zurich, 2000.

[96] Riedel, J., Bhaskaran, S., Eldred, D. B., Gaskell, R. A., Grasso, C. A., Kennedy, B., Kubitscheck, D., Mastrodemos, N., Synnott, S. P., Vaughan, A., and Werner, R. A., "Autonav mark3: Engineering the next generation of autonomous onboard navigation and guidance," 2006.

[97] Riedel, J. E., "Autonomous optical navigation (autonav) ds1 technology validation report," tech. rep., Jet Propulsion Laboratory, California Institute of TEchnology, 2001.

[98] Riedel, J. E., Wang, T.-C., and Werner, R., "Configuring the deep impact autonav system for lunar, comet and mars landing," in *AIAA AAS Astrodynamics Specialist Conference and Exhibit*, (Honolulu, HI), August 2008.

[99] Rush, J., Isreal, D., and Ramos, C., *DRAFT Communication and Navigation Systems Roadmap, Technology Area 05*. National Aeronautics and Space Administration, November 2010.

[100] Saltelli, A., Annoni, P., Azzini, I., Campolongo, F., Ratto, M., and Tarantola, S., "Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index," *Computer Physics Communications*, vol. 181, no. 2, pp. 259 – 270, 2010.

[101] Saltelli, A., Annoni, P., Azzini, I., Campolongo, F., Ratto, M., and Tarantola, S., "Variance-based sensitivity analysis of model output. design and estimator for the total sensitivity index," *Computer Physics Communications*, vol. 181, no. 2, pp. 259–270, 2010.

[102] Saltelli, A., Tarantola, S., and Chan, K.-S., "A quantitative model-independent method for global sensitivity analysis of model output," *Technometrics*, vol. 41, no. 1, pp. 39–56, 1999.

[103] Schier, J. S., Rush, J. J., Williams, W. D., and Vrotsos, P., "Space communication architecture supporting exploration and science: Plans and studies for 2010-2030," in *1st Space Exploration Conference*, (Orlando, FL), January 2005.

[104] Schoolcraft, J. B., "The deep impact network experiment - concept, motivation, and results," in *SpaceOps 2010 Conference*, (Huntsville, AL), April 2010.

[105] Sheikh, S. I., Pines, D. J., Ray, P. S., Wood, K. S., Lovellette, M. N., and Wolff, M. T., "Spacecraft navigation using x-ray pulsars," *Journal of Guidance, Control, and Dynamics*, vol. 29, January-February 2006.

[106] Sheikh, S. I., *The Use of Variable Celestial X-Ray Sources for Spacecraft Navigation*. PhD thesis, University of Maryland, 2005.

[107] Siegfried, R., Lehmann, A., El Abdouni Khayari, R., and Kiesling, T., "A reference model for agent-based modeling and simulation," in *Proceedings of the 2009 Spring Simulation Multiconference*, SpringSim '09, (San Diego, CA, USA), pp. 23:1–23:8, Society for Computer Simulation International, 2009.

[108] Simon, D., *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley, 2006.

[109] Sinha, R., Paredis, C., Liang, V.-C., and Khosla, P. K., "Modeling and simulation methods for design of engineering systems.," *Journal of Computing and Information Science in Engineering(Transactions of the ASME)*, vol. 123, no. 1, pp. 84–91, 2001.

[110] Sobol, I., "On the distribution of points in a cube and the approximate evaluation of integrals," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, no. 4, pp. 86 – 112, 1967.

[111] Spangelo, S. and Team, I. S. S. C., "Model based systems engineering (mbse) applied to radio aurora explorer (rax) cubesat mission operational scenarios," in *2013 IEEE Aerospace Conference*, (Big Sky, Montana), March 2013.

[112] Spangelo, S., Kaslow, D., Delp, C., Cole, B., Anderson, L., Fosse, E., Gilbert, B., Hartman, L., Kahn, T., and Cutler, J., "Applying model based systems engineering (mbse) to a standard cubesat," in *2012 IEEE Aerospace Conference*, (Big Sky, Montana), March 2012.

[113] Stadter, P., Chacos, A., Heins, R., Moore, G., Olsen, E., and Asher, M., "Confluence of navigation, communication, and control in distributed spacecraft systems," in *Aerospace Conference, 2001, IEEE Proceedings.*, vol. 2, pp. 2/563–2/578 vol.2.

[114] Standish, E. M., "Time scales in the JPL and CfA ephemerides," *Astronomy and Astroyphysics*, vol. 336, pp. 381–384, Aug. 1998.

[115] Stastny, N. B. and Geller, D. K., "Autonomous optical navigation at jupiter: A linear covariance analysis," *Journal of Spacecraft and Rockets*, vol. 45, no. 2, pp. 290–298, 2008.

[116] Steitz, D. E., "Communications, navigation and in-space propulsion technologies selected for nasa flight demonstration." http://www.nasa.gov/home/hqnews/2011/aug/HQ_11–272_TDM_Selections.html, August 2011.

[117] Tapley, B. D., Bettadpur, S., Watkins, M., and Reigber, C., "The gravity recovery and climate experiment: Mission overview and early results," *Geophys. Res. Lett*, vol. 31, no. 9, p. L09607, 2004.

[118] Taylor, J., Butman, S., Edwards, C., Ilott, P., Kornfeld, R., Lee, D., Shaffer, S., and Signori, G., *Phoenix Telecommunications.* Deep Space Communications and Navigation Systems Center of Excellence Design and Performance Summary Series, Jet Propulsion Laboratory, California Institute of Technology, August 2010.

[119] Taylor, J., Lee, D. K., and Shambayati, S., *Mars Reconnaissance Orbiter Telecommunications.* Deep Space Communications and Navigation Systems Center of Excellence Design and Performance Summary Series, Jet Propulsion Laboratory, California Institute of Technology, September 2006.

[120] Terrile, R., "Automated design of spacecraft telecommunications using evolutionary computational techniques," in *IEEE Aerospace Conference Proceedings*, March 2007.

[121] Thornton, C. L. and Border, J. S., "Radiometric tracking techniques for deep space navigation," tech. rep., Jet Propulsion Laboratory, California Institute of Technology, October 2000.

[122] Training, A. L., *Apollo Training: Guidance and Control Systems, Block II S/C 101.* NASA, Septermber 1967.

[123] Turner, A., "An open-source, extensible, spacecraft simulation and modeling environment framework," Master's thesis, Virginia Polytechnic Institute and State University, 2003.

[124] Vallado, D., *Fundamentals of Astrodynamics and Applications.* Springer, 3 ed., May 2007.

[125] Van Dierendonck, A. and McGraw, J., "Relationship between allan variances and kalman filter parameters," in *Proceedigns of the 16th Annual Precise Time and Time Interval Systems and Applications*, pp. 273–293, NASA Goddard Space Flight Center, November 1984.

[126] Vanderperren, Y. and Dehaene, W., "Sysml and systems engineering applied to uml-based soc design," 2005.

[127] Weilkiens, T., *Systems Engineering with SysML/UML: Modeling, Analysis, Design.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[128] Weiss, G., *Mutiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Intelligent Robotics and Autonomous Agents Series, Mit Press, 1999.

[129] Wertz, J. R., "Autonomous navigation and autonomous orbit control in planetary orbits as a means of reducing operations cost," in *5th International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations*, (Pasadena, CA), July 8-11,2003.

[130] William M. Owen, J. and Bhaskaran, S., "Interplanetary optical navigation 101." Presentation, http://hdl.handle.net/2014/38410, July 2003.

[131] Winternitz, L., Moreau, M., and Jr., G. J. B., "Navigator gps receiver for fast acquisition and weak signal space applications," in *ION GNSS Meeting*, (Long Beach, CA), September 2004.

[132] Yunck, T. P. and Wu, S.-C., "Tracking geosynchronous satellites by very-long-baseline interferometry," *Journal of Guidance*, vol. 6, pp. 382–386, Sept.-Oct. 1983.

[133] Zeigler, B. and Sarjoughian, H., *Guide to Modeling and Simulation of Systems of Systems*. Springer, 2012.

[134] ZEIGLER, B., Prahofer, H., and Kim, T., *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Acad. Press, 2000.