ENABLING COLLABORATIVE BEHAVIORS AMONG CUBESATS

A Thesis
Presented to
The Academic Faculty

by

Daniel C. Browne

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Computational Science and Engineering

Georgia Institute of Technology
August 2011

ENABLING COLLABORATIVE BEHAVIORS AMONG CUBESATS

Approved by:

Dr. Ryan P. Russell, Committee Chair
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Carlee Bishop
Georgia Tech Research Institute
Electronic Systems Laboratory
Georgia Institute of Technology

Dr. Richard Vuduc
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Michael E. West
Georgia Tech Research Institute
Electronic Systems Laboratory
Georgia Institute of Technology

Date Approved: June 10^{th}, 2011

I dedicate this thesis to my Dad for his constant love and support, not only in school but all of life's endeavours.

ACKNOWLEDGEMENTS

I would like to begin by acknowledging all of the people that assisted me completing my degree and specifically this body and work.

First, I want to thank Dr. Carlee Bishop for assisting me in finding work at the Georgia Tech Research Institute (GTRI) that corresponded well with my interests and graduate studies. Dr. Bishop provided great assistance in developing a plan of action for my research, readily provided guidance, and helped me work through drafts of this thesis. Furthermore, Dr. Bishop introduced Dr. Mick West and myself, resulting in an opportunity to conduct CubeSat-related research at GTRI. Dr. West mentors many graduate students and it was a privilege getting to work with him on this research as well as other related projects. He regularly guided my research and assisted me in overcoming challenges. Dr. Ryan Russell inspired me several years ago to become intrigued by astrodynamics, as it is a field that allows my interests of engineering and computing to combine. Dr. Russell continually ensured that my end product would be something to be proud of by asking the tough questions and challenging me to explore all possible avenues. Finally, Dr. Rich Vuduc is an excellent teacher who was able to take my programming capabilities from the "engineering world" and mold it into a skill set including parallel and high-performance programming. His interest in high-performance computing is contagious and I look forward to exploring this developing field more in the coming years. I still have much to learn from all my advisors and look forward to continuing my academic and professional careers with them.

I additionally want to thank Dr. Amy Pritchett for her patience and long hours she put in to ensure that all of her modeling and simulation students were successful. I learned much from her lectures and office hours that prepared me to conduct and be successful in the research embodied in this document.

I thank my Mom, Dad, and brothers, Matthew and Mike, for their continued encouragement throughout my schooling. They always help me stay focus on what was important while encouraging me toward success.

Finally, my wife Megan, who is currently bearing our first child, a baby boy. She is a constant source of love and encouragement, continually making sacrifices for me as we pushed through together. I love you, Megan; it is a privilege to be your husband and I look forward to the exciting journey of raising a family with you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS OR ABBREVIATIONS

A	Amp (or Ampere); unit for electric current.
a	Keplerian orbit semi-major axis; see Figure 1.
AES	Advanced Encryption Standard; a U.S. government-adopted symmetric-key cryptographic standard with keys of size 128, 192, or 256.
AFRL	Air Force Research Lab.
CA	California.
cm	metric unit; centimeter.
CMP	chip multiprocessor; microprocessor which contains multiple computing cores on a single die, allowing for parallel processing.
COTS	Commercial off-the-shelf.
CSV	Comma-separated values; a standard text document for storing tabulate data.
dB	decibel; logarithmic unit of power often used for RF signals.
deg	unit; degrees.
DLR	German Aerospace Center.
DSN	Distributed Sensor Network; a decentralized network of inexpensive sesors (generally low-performing and low-reliability to result in low cost) that as a team can provide a higher-performing and higher-reliability system than a conventional single-node sensor.
e	Keplerian orbit eccentricity; see Figure 1.
EKF	Extended Kalman Filter; extends the standard Kalman Filter (KF) by allowing nonlinear models and mappings from measurements to states.
GPS	Global Positiong System; network of 32 half-geosynchronous satellites that provide the means to calculate highly accurate position and velocity data of an object fitted with a receiver.
GRACE	Gravity Recovery and Climate Experiment mission implemented in a collaborative effort by NASA's JPL, DLR, and University of Texas' Center for Space Research.
i	Keplerian orbit inclination; see Figure 1.
ISL	Inter-Satellite Link.
J	Joule; unit for energy or work equal to the product of a W and second.
JPL	Jet Propulsion Laboratory; a NASA lab located in Pasadena, CA.

KF Kalman Filter; a process in which an estimate of state is determined by utilizing measurement(s) from one or more sensors while adjacently integrating the state forward in time using a linear model of the expected dynamics.

km metric unit; kilometer.

LEO Low Earth Orbit; used to refer to orbits with altitudes of 200 to $1,000$ km.

LPE Lagrange Planetary Equation; equations that describe the change in orbital elements over time due to perturbations.

M Keplerian orbit mean anomaly; related to ν through eccentricity and the eccentric anomaly such that $\dot{M}$ is constant.

MIPS Million instructions per second; a unit of measure describing the number of atomic operations a digital process can complete in a given period of time.

mJ milliJoule; one-thousandth of of a J.

MPI Message Passing Interface; parallel programming framework for distributed memory systems which uses message passing between cores for the sharing of data.

μ**A** micro-Amp; one-thousandth of an A.

NASA National Air and Space Administration, United State of America.

ν Keplerian orbit true anomaly; see Figure 1.

ODE Ordinary differential equation.

ω Keplerian orbit argument of periapse; see Figure 1.

Ω Keplerian orbit longitude of ascending node; see Figure 1.

P-POD Poly-Picosatellite Orbiter Deployer; standard interface developed by CalPoly and Stanford for integrating CubeSats to a launch vehicle which provides an ejection mechanism.

RF Radio Frequency.

RSSI Received Signal Strength Indicator; value available by some transceivers that indicates the stength, in dB, of the last received signal.

SENSE Space Environment Nanosatellite Experiment; mission concept by Air Force Space Command to utilize two 3U CubeSats for the collection of space weather science data.

SRP Solar Radiation Pressure; a pertubation force exerted on spacecraft due to solar flux.

STM State Transition Matrix; the STM is propagated alongside a reference trajectory and allows for the calculation of nearby spacecraft at a later time if an initial offset is known.

TRL Technology Readiness Level; a scale of 1 to 9 where 1 indicates a technology is in the early conception stages and 9 indicates a technology has proven flight experiences.

UKF Unscented Kalman Filter; unlike the EKF which linearizes about a point of interest, the UKF uses deterministic sampling to calculate the Jacobian and Gain Matrix.

V Volt; unit for electric potential.

VBA Visual Basic for Applications; the language used within Microsoft Excel to provide enhanced functionality through coding Macros.

W Watt; unit for power consumption, product of V and A.

WISE Wide-field Infrared Survey Explorer; a JPL mission launched on December 14, 2009 with a primary mission of surveying the sky.

LIST OF LISTINGS

SUMMARY

Future LEO missions will require teams of collaborating satellites to achieve the performance, robustness, and cost effectiveness desired for the next generation of spacecraft systems. Various research labs across the globe have spent significant effort over the past few decades, and continue still, developing software and hardware components to tackle the challenging tasks of enabling collaboration among small satellites.

This research effort focuses on the need for teams of small spacecrafts, specifically targeting CubeSat applications, to maintain accurate knowledge of an entire satellite formation. By providing a lightweight, power-efficient means for each spacecraft to track the motion in both an inertial and relative sense of the entire CubeSat team, inter-satellite communication can be limited while still enabling collaborative behaviors. CubeSats provide an inexpensive and almost completely commercial off-the-shelf (COTS) means to reach LEO (low Earth orbit); however, their mere size results in some rather tight restrictions. Arguably the most limiting factor when using the CubeSat framework is the available power. For a 3U CubeSat a total surface area of 0.14 m^2 is available with an optimistic Sun-facing area of 0.06 m^2 at a 45° angle [5]. Using the most efficient COTS solar panels available, the maximum generated power is about 15 W [8, 66]; this tight restriction requires that power savings be realized at every subsystem down to each component of the functional architecture.

Through the development of an efficient means to conduct on-board orbit determination, team propagation, and limiting inter-satellite communication, power savings can be realized. In order to achieve an efficient module for enabling collaborative behaviors, a software development and hardware selection plan is developed. The first step in the research is to explore the propagation of small spacecraft LEOs. Various propagation techniques, force models, and integration routines are examined concluding with details on the propagation implementation that balances the trade-off between accuracy and cost (cost being computational expense which requires power). With a well-defined model, an on-board orbit determination algorithm using a GPS sensor and an extended Kalman filter

(EKF) is implemented to provide meter and centimeter per second position and velocity accuracy, respectively. Inter-satellite communication is rare in small satellites, so each piece of an ISL (inter-satellite link) framework is discussed including message structures, protocols and frequency bands, and final hardware selection.

With the three primary pieces of the module completed or defined, integration of the components to conduct simulations and analysis is completed. Simulation results prove the advantage of using the EKF over a naïve approach such as broadcasting the GPS solution directly. Furthermore, by selecting a transceiver with a variable transmit power setting, additional power savings can be gained even when broadcasting. Hardware-in-the-loop simulations demonstrate the use of actual CubeSat hardware with the developed software pieces.

The typical microcontroller is powered using 3.3 V and consumes as little as 0.1 μA during standby mode and generally between 200 and 250 μA per MIPS (million instructions per second) [15]. While an S-Band transceiver requires only slightly more power in standby mode, since it is powered with 5 V and can pull up to 1 A, it consumes far more power when receiving or transmitting. During transmission more than 4 W of power are consumed when the transmit power is set to 1 W and when receiving slightly more than 1 W is required [47]. From these estimate values it quickly becomes evident that if we can remove just 1 second of S-Band transmit time that more than 20 minutes of microcontroller computation time operating at 4 MIPS can be completed before using the same amount of energy (about 4 J).

In summary, the collaborative module discussed in this thesis provides a CubeSat-focused software and hardware solution to assist in the enabling of collaborative behaviors among satellites. This module provides a means to acquire and maintain team state knowledge for planning of data collection, optimal formation reorganization, and collision avoidance. Furthermore, with an ISL (Inter-satellite link) framework in place, sharing of sensor data for on-board processing is transparent to the other subsystems. With the introduction of the collaborative module, other spacecraft subsystems and software components can build upon its functionality to develop higher performing small satellite formations.

CHAPTER I

INTRODUCTION

Popularized separately over the past decade, combining technology advances in both **CubeSats** and **formation flying** among small satellites will assist in realizing the next generation of spacecraft systems. Universities and amateurs have and are taking advantage of the CubeSat and P-POD (Poly-Picosatellite Orbiter Deployer) specifications to reach space, but all complete mission successes have been single satellite systems [64]. Recent endeavours including NASA's Afternoon-Train (more commonly referred to as the A-Train) have begun pursuing formation flight for larger spacecraft [43]. The A-Train is currently composed of four satellites in a 13:30 sun-synchronous orbit that all cross the equator within several minutes of each other. Two more satellites are planned to join the formation in the next few years. (These two scheduled missions do not include the GLORY satellite recently lost due to a failure to reach its intended orbit on March 4, 2011 [26, 27].)

One effort, coined QB50, is organizing launches for 50 2U CubeSats into a low altitude constellation. The constellation is designed to collect data on the ionosphere. Although the CubeSat data will later be correlated there is no plan for this mission to demonstrate formation flying [65].

Recently the Swedish Space Corporation has been demonstrating autonomous formation flying between two nanosatellites on their PRISMA mission; formation flying on the order of meters has already been demonstrated to date. Leading up to the mission extensive work was conducted, some in conjunction with DLR (German Aerospace Center), in developing the control laws and necessary autonomy [29, 30, 33, 34, 52, 53, 56]. PRISMA is still under way with plans to demonstrate additional autonomous capabilities.

From just these few examples, with more provided in the following sections, the popularity of both CubeSat missions and formation flying demonstrations is clear. Efforts have been made to explore CubeSat formations, but additional technology development is necessary. With the collaborative module explained throughout this thesis, successful CubeSat formations could be realized

using simpler, lower-power subsystem components. Pairing this technology with other advancements including low-power micro-controllers, star trackers, nano arc thrusters, and high efficiency solar arrays will provide the pieces necessary for highly-capable CubeSat formations.

1.1 Motivation and Application

Spacecraft formations are not an entirely new idea, but additional effort is being put forth in this arena since it is hypothesized that the next generation of LEO missions will require teams of collaborating spacecraft in order to deliver the desired performance. Decreasing cost while increasing robustness is further pushing the ideas of fractionated spacecraft. Although concepts for formations or fractionated spacecraft date back to at least 1984 [49], movement in the last five years has pushed these ideas to the forefront once again.

A second motivation comes from the growing CubeSat community. CubeSats missions have primarily been executed by universities, but since the original conception in 1999, these small cubes in the sky have demonstrated surprising capabilities. The CubeSat specification allows for systems with a base of 10 cm x 10 cm and heights of 5, 10, 15, 20, or 30 cm [5]. Defense-orientated companies such as Aerospace Corporation and Boeing have executed their own CubeSat systems as well, but only on a small scale for demonstration purposes. Last year, however, Air Force Space Command released a solicitation for the procurement of two 3U CubeSats for a space weather experiment (Space Environment Nanosatellite Experiment, SENSE) [6]. One of the primary objectives was to measure scintillations due to the ionosphere by monitoring the highly calibrated GPS L1 band of 1.575 GHz. In general, however, the Air Force was interested in the design of a versatile and robust CubeSat bus capable of supporting various payload suites. Although not clearly mentioned in the solicitation, the use of two systems with complementary payloads would provide an opportunity for demonstrating collaborative behavior through complementary payload selections.

Although this research has a focus toward CubeSat application, the overarching concepts are not limited to that small platform. The software and algorithms for maintaining team state knowledge are independent of platform and even sensor selection for the purposes of obtaining GPS measurements and attitude quaternions. Additionally, the communications protocol to be discussed, though proprietary and designed for terrestrial applications, could be easily modified or replaced with a

government-rated protocol to enable encryption or other features currently lacking. The following section provides more background on large spacecraft formation efforts; it is these efforts that are both part of the motivation and future application of our collaborative capability.

1.2 Satellite Formations and the Future of LEO Missions

In 2007 DARPA first announced its Future, Fast, Flexible, Fractionated, Free-Flying Spacecraft United by Information Exchange effort (more commonly referred to as System F6) [48, 61]. The goal of System F6 is to develop the various technologies necessary to realize fractionated spacecraft. Fractionated spacecraft can be classified as a type of spacecraft formation, where what would have traditionally been considered subsystems, are now individual space vehicles in and of themselves. For example, rather than sending up a single "monolith" with a large command and data handling system for image processing, a pair of high resolution cameras, and a one meter dish for high data rate downlink, separate those subsystems. One spacecraft could be devoted to command and data handling and have very simple support systems while the primary payload is a high performance processor suite and data storage. Each high resolution camera could be set in orbit on an individual craft, possibly increasing the value from obtaining images at different angles simultaneously. Lastly, a communications vehicle would be fitted with a high power amplifier and high gain antenna, with little else, providing a link for the rest of the team to the ground. Although each spacecraft would need its own set of simple subsystems, being able to devote a single craft to one task allows for small, lighter, cheaper modules that can likely be developed more rapidly and require smaller launch vehicles. Fractionated systems are also considered more versatile and robust; in the event that the command and data handling module suffers a massive single event upset knocking out all of its non-volatile storage, the entire mission is not a loss. Temporarily, the remaining spacecraft can use their simple command and data handling systems to supplement and a new module could be sent up; this is far less expensive and time consuming than developing an entire new spacecraft. Versatility can be realized in the same way. Take the previous example with the high resolution cameras: if five years after the mission launches new technology allows for twice the resolution, the current process would be to consider building an entire new spacecraft. What, if instead, new camera modules were launched into orbit and the old cameras decommissioned? Complete reuse of the command and

data handling and communications spacecrafts could greatly reduce the cost of what is basically an entirely new mission. Late in the first decade of the 2000s System F6 fell off the radar, but during the third quarter of 2010 a new Broad Agency Announcement was released revitalizing the effort [48, 7].

System F6 is not alone. As mentioned previously, NASA's "A-Train" is currently composed of four complementary science satellites collecting data to enhance climate forecasts [4]. Although the fractionated approach has not been considered, by placing the quartet of satellites in nearly the same orbit, sensor data obtained simultaneously can be compared and correlated for superior science value. Fifteen instruments are distributed across the four satellites and span the electromagnetic spectrum. Measurements on water vapor, the vertical greenhouse gas column, 3D cloud imagery, and more have been and are being collected; this mission data is the best available to date, assisting climatologists in understanding our changing planet.

A final example mission worthy of mention is the Air Force Research Lab's TechSat-21 project [69]. Although cancelled in 2003 due to budget overruns, AFRL conducted significant work in the area of formation flying and distributed sensor data fusion required to realize TechSat-21. The overall mission concept was to place three satellites in a reconfigurable cluster formation to execute a "sparse aperture antenna system" [69]. Synthetic aperture radar, the primary technology to be demonstrated by TechSat-21, allows for a network of distributed sensors, such as RF antennas, to simultaneously collect or emit RF radiation over a distributed region. Through this collaboration, a "virtual" antenna is created with performance nearly equivalent to a single dish as large as the area covered by the separate antennas. Although TechSat-21 was not completely realized, much was learned in the areas of relative motion, sensor data fusion, and the design of relatively small yet robust satellites; each of these technology advances have continued in separate avenues bringing us closer to the successful execution of a missions such as TechSat-21.

1.3 Satellite Relative Motion

When considering satellite formations, generally the relative positioning of spacecraft with respect to each other rather than the absolute positions is of more importance. The field of satellite relative motion dates back to the middle of the last century when the earliest model was developed: the Hill's

or Clohessy-Wiltshire model provides closed form solutions which relate motion between satellites in near-circular orbits with only two-body forces considered [25]. Although a simple model, the Hill's equations being closed form removes the need to integrate equations of motion to calculate the relative position at any point in time. This model has been improved upon by removing the circular orbit and force model constraints, although sometimes losing the closed-form solutions.

Although the Clohessy-Wiltshire equations provide a nice, simple close-form solution, expanding the linearization process to include higher order terms provides significant improvement; Karlgarrd used the method of multiple scales to develop a second-order analytical solution [39]. His results displayed a significant decrease in error, provided terms to capture the coupling of in-plane and out-of-plane motion between two satellites, and for orbits with different periods improved the drift rate estimate.

Pluym and Damaren developed a relative motion model which included differential gravity terms to the third-order and J_2 perturbations up to the second-order [54]. This model displayed significant improvement for the prediction of relative position over ten or more orbits while simpler models begin losing accuracy after as little as half an orbit.

Other interesting work in the area of relative motion has come out of AFRL over the past decade. Some of this work defines a set of relative orbital elements; like regular orbital elements, this set of six parameters allows for simple visualization of the interaction between two spacecraft in similar orbits [46]. Using this developed framework, improved relative motion models have been developed as well as new algorithms for optimal formation maintenance and reconfiguration.

1.4 Distributed Sensor Networks

With research roots in the area of defense, distributed sensor networks (or DSNs) are viewed as the future replacement of conventional single-point, high-performing, and highly reliable sensors; to some degree this transition is already taking place for terrestrial applications [36]. Since the early 1990s research has been conducted in the field of decentralized or distributed sensor networks. A primary argument behind DSNs is that by using low cost sensors that are decentralized and distributed over a reference area, this system of low-performing and lower reliability sensors can provide a complete solution that is more capable than a single node and is far more robust.

In a DSN the loss of a single sensor has little to no impact on a system's capability; comparatively, in a conventional single-node solution, loss of that node or even just some component on that node would result in a system's capability being greatly affected, possibly to the point of having no value. The robustness of a single-node solution is equivalent to the product of the component reliabilities that make up that system. For a distributed system, the overall robustness can be improved by simply increasing the number of nodes with similar capabilities. Since a sensor or node should be low cost and relatively simple, adding nodes is not prohibitive.

Two primary capabilities required by a DSN that are likely exempt from single-node solutions are the need for inter-node communication and sensor fusion; the latter may still be implemented for a single-node if data from multiple sensors can be utilized to determine some other factor not measured directly. For the most robust DSN, it is desirable for inter-node communication to be decentralized. A simple communications solution is the standard hub-and-spoke setup, but having a hub introduces a single point of failure vulnerability. Decentralizing communication provides robustness by allowing a single-node failure without eliminating communicative capabilities between other sensors.

Identified as one of, if not the, main technology needed to realize distributed sensor networks was data fusion. Data fusion is the process by which measurements collected from multiple sensors, whether the same data or complementary, are corellated in order to provide enhanced information. Although the concept itself is not new and is readily seen in nature, computing advances in the last 20 years have provided the resources necessary to realize data fusion algorithms. Fusion algorithms for both equivalent and complementary sensor data are required for the best performance from a DSN. Through data fusion many of the advantages of DSNs are enabled. For example, improved estimates of measured values are possible when using identical or complementary sensors. Furthermore, overall observability of a domain is improved through the relative positioning of static or dynamic sensors [45].

It is programs such as System F6 described previously that desire to take advantage of the technologies developed for distributed sensor networks. Understanding the distinction between a constellation and a satellite formation become important as this point since it is this concept of decentralized nodal systems that have encouraged satellite formations. Constellations such as Iridium

and GPS distribute identical, or nearly identical, satellites into separate orbits to provide instantaneous global coverage. Although a constellation may provide robustness through removing the single-node failure, it is not realizing the goal of modular low cost systems. A satellite formation aims at having multiple, likely differently equipped, satellites in close proximity orbits. An example is AFRL's TechSat-21 mission design in which three different satellites fitted with receivers would have created a synthetic aperture radar. Equipping each spacecraft with a different sensor suite (as in the case in the "A-Train") or specializing each spacecraft to be a module (such as the desire of System F6) provides the desired robustness and value increase in the data obtained. Through developing a collaborative module for the CubeSat platform, this thesis asserts that distributed sensor networks in LEO have the capability of being realized on nano- or picosatellite platforms in the near future.

1.5 ODE Integration

A primary underlying component of the collaborative module is ODE integration. Although the simplest integration method is the first-order Euler [55], probably the most well-known family of integrators is the Runge-Kutta methods. This family contains algorithms ranging from the commonly used fourth-order up to orders in the teens; additionally fixed versus variable time steps (also known as adaptive time step) can be considered. In short, Runge-Kutta methods use a derivative function, often referred to as a model, to calculate the slopes of n states; the derivatives are solely a function of the state vector and time. Depending on the order of the Runge-Kutta method and whether a fixed or adaptive time step algorithm is utilized, intermediate derivatives are calculated and are finally combined through a weighted sum. Determining appropriate constants for the weight requires solving an extensive over-constrained system of equations. For conciseness, derivations and additional insight into the theory are absent from this summary; the basic theory and algorithms utilized in this thesis work are readily available in listed references [31, 55].

Among the orbital mechanics community a popular integration algorithm is Gauss-Jackson [21]. The Gauss-Jackson method provides a numerical solution to second-order differential equations. The method utilizes an additional function denoted as $\delta^{-2} f$ whose second differences are equivalent to f. The summation to determine δ^{-2} can be calculated exactly, with rounding errors only occurring

one time [37]. In the field of astrodynamics, rounding errors create significant frustration since the orders of magnitude between distance and forces are often great, unless careful attention is given to selected units of measure. By solving the second-order ODE directly and removing rounding errors, improved numerical solutions are feasible.

Details on the ODE integration method selection process are provided in Chapter 2.

1.6 Kalman Filtering

One portion of the collaborative module described in this thesis involves on-board orbit determination. The primary sensor selected for the module implementation is a GPS receiver designed for the LEO environment. No perfect receiver exists, therefore filtering of noise is necessary for sensor output data to be useful. Additionally, by implementing on-board orbit determination, spacecraft and force model constants that may not be known exactly can be estimated to improve the overall model. As discussed, one of the goals of the collaborative module software is to provide sufficiently accurate state propagation such that inter-satellite communication is limited. Therefore, actively improving the model through parameter estimation assists in realizing this goal.

Use of a Kalman filter, specifically an extended Kalman filter (EKF), provides the means to estimate model parameters through filtering sensor data. In the case of the collaborative module, a GPS receiver provides direct measures of position or position and velocity for the orbit propagation model. If extended to include attitude propagation, a star tracker or magnetometer provides the necessary attitude quaternion. A Kalman filter combines sensor data with a state estimation calculated through integration of a representative force model while considering noise (error) in both the sensor data and model [24]. Applications for Kalman filters cover the entire spectrum of sensors and improve data accuracy by fusing measurements from suites of sensors. For the orbit determination problem use of an EKF algorithm, also referred to as the Sequential Estimation Algorithm, is well understood and multiple implementations have been published [17, 22, 63, 67]. Use of these example implementations assisted in the development of the EKF on-board orbit determination algorithm for the collaborative module. See reference [24, 63] for derivation of the Kalman filter algorithms; details on the implementation are provided in Chapter 3.

1.7 Research Objectives

At the onset of this research the primary objective was to develop a *module*, fit to the CubeSat form factor, that could provide the means to enable satellite autonomous collaboration. Current CubeSat commercial off-the-shelf (COTS) components are modular per spacecraft subsystem; modules exist for command and data handling, communications, structure, and power management [13, 8]. The end-goal of this research is to provide the design for a plug-and-play solution, slightly customizable for a given mission scenario, that would provide the on-board software and hardware for collaborative behavior. In order to enable collaborative behavior three primary requirements were identified (in no particular order of importance):

- Shared state knowledge (position, velocity, and model parameters with framework for extension to include attitude and attitude rates) of all spacecraft team members
- Framework/Protocol for sharing of state and sensor data
- Inter-satellite link with sufficient bandwidth to satisfy the communications required by the previous two items

These three goals map to four primary research branches with a fifth branch identified as future work:

- Orbit propagation
  - Force model selection
  - Propagation technique (Cartesian, orbital elements, relative motion, STM)
  - Integration method (Runge-Kutta, of various orders and fixed- vs. variable-step)
- On-board orbit and attitude determination
  - Hardware component selection (GPS receiver, star tracker, magnetometer)
  - Extended Kalman filter
- Algorithm/Process for sharing state knowledge
- Inter-satellite Link (ISL)

  - Band selection (UHF, S-Band)
  - Protocol selection
- Consideration of a final CubeSat bus design and mission concept limitations, with the primary design process being set aside for future work

Software development is the primary deliverable associated with the first four branches listed above; the fifth branch guides some of the software design decision to ensure it can be realized in the long-term. This first part of the research effort will culminate in demonstrating all software capabilities with some CubeSat COTS hardware in-the-loop. For example, simulations with generic PCs representing individual spacecraft communicate over S-Band to demonstrate not only the ISL but also the orbit propagation, on-board determination, and state sharing capabilities. Additionally, certain software components run on a micro-controller to demonstrate that a simpler processor is capable of executing the necessary software. This thesis provides details on each software component implementation, hardware selection decision, and analysis of simulation results.

1.8 Outline

The remainder of this thesis is organized as follows. Chapter 2 provides a review of propagation techniques, force models, and integration routines explored; details of the final on-board implementation are provided. Chapter 3 presents the on-board orbit determination functionality. This includes discussion of the extended Kalman filter utilized for estimating a spacecraft's own state vector, which includes "constants" within the propagation model. Chapter 4 provides details on the inter-satellite communications including protocol, message structure, and frequency band selections. The pieces from Chapters 2, 3, and 4 are combined into one algorithm for ensuring accurate state knowledge among the team. Chapter 5 first discusses the algorithm and then details the various simulation setups (including hardware-in-the-loop) and analyzes the results. Chapter 6 concludes the thesis with a summary discussion of completed work plus suggestions on future work related to this research effort.

CHAPTER II

ORBIT PROPAGATION

Propagation of a spacecraft in LEO is the first requirement of the collaborative module proposed by this research. Three sets of trade studies encompass this portion of the research which include the selection of a propagation technique, force model, and integration routine.

Force models range from the most simplistic being two-body motion up to including a high fidelity gravitational model of the Earth represented by ten thousand estimated constants, third-body perturbations from other celestial bodies, drag effects, solar radiation pressure, and more [18]. For an on-board implementation a balance between model fidelity and computational cost is necessary in order to realize a gain in formation performance. In other words, one could solve the problem of shared accurate state knowledge among satellite teammates by simply broadcasting the data from a space-rated GPS receiver at a high frequency close to that of the sensor. However, the cost with regards to power and data handling would in essence make the primary mission of every CubeSat be the tracking of the formation. At the opposite end of the spectrum resides the near-perfect model scenario which requires immense computing power but hardly ever needs to broadcast an ephemeris update; as in the scenario with It is the need for this balance which drove the first branch of this thesis research, exploring a range of fidelity force models. The model selection process chose four main types of perturbation effects for consideration: non-spherical gravitational potential, drag, solar radiation pressure, and N-body.

Besides the selection of a specific model fidelity, a second arena of comparison exists. Orbit propagation in standard Cartesian coordinates is a common technique, but often times it is more appropriate to propagate the Lagrange Planetary Equations of Motions (LPEs). The LPEs are often referred to as Lagrangian VOP since these equations are developed through the variation of parameters [18, 40, 66]. In short, the Lagrangian VOP are equations of motion that capture the osculating and secular changes in the six orbital elements. Equation (1) summarizes the Lagrangian VOP where $\vec{c}$ is a six dimensional vector containing the orbital elements a, e, i, ω, Ω, and M. Figure

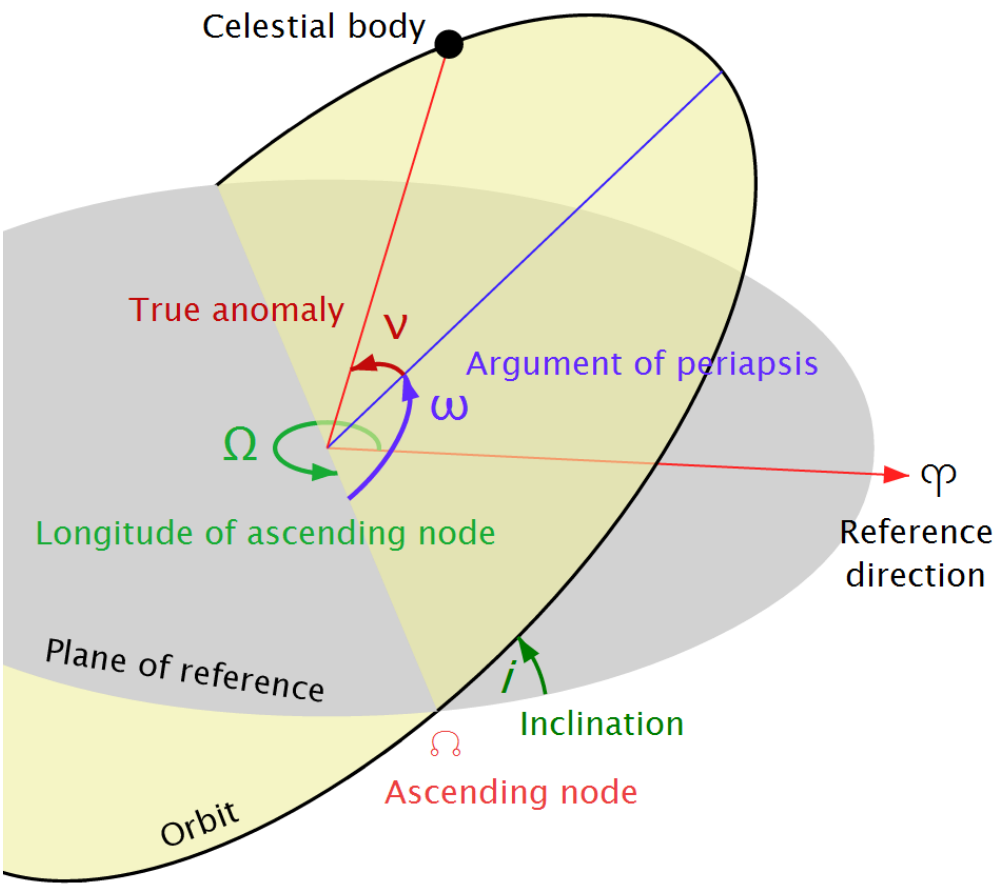

Figure 1: Visualization of the six common orbital elements (a, e, i, ω, Ω, ν) that define the orbit shape and current location of the spacecraft on the orbit [3].

1 visualizes these orbital elements, except replaces M with ν. A more complete description of the Lagrangian VOP equations of motion is provided in a following section.

$$\frac{d\vec{c}}{dt} = f(\vec{c}, t) \tag{1}$$

A third method of orbit propagation, when considering multiple satellites, is that of relative states. For certain applications it may be more important to know the relative position and velocity rather than the absolute state of each spacecraft. The simplest relative motion equations are generally referred to as the Clohessy-Wiltshire (CW) or Hill's Equations [25]. Clohessy and Wiltshire's equations are the result of linearizing the relative motion equations with the assumptions of two-body motion and a circular orbit. The greatest benefit from this linearization is that the final equations are closed-form, removing the need for integration altogether. Note that the equations

are expressed in the rotating Hill's frame, therefore in order to determine absolute position for each spacecraft a reference frame transformation is necessary.

The third study compares integration routines as well as fixed- versus variable-step solutions. Fourth through eighth order Runge-Kutta methods are implemented to determine the trade off between step size and amount of computation per step [31]. The following subsection provides details on each integration routine.

This chapter discusses the propagation techniques, models, and integration routines explored. Details including computational cost (both flops and memory footprint) and general implementation information is provided.

2.1 Orbit "Truth"

Throughout this chapter results are provided which compare the propagation of an orbit using a particular model with some defined "truth". It is important to provide information on how this truth model is ascertained and why comparing to this set of data is valid. For purposes of model testing two sets of data are used for comparison. The first is generated using **a. i. solution's** *FreeFlyer* software package. In this software package different perturbation effects can be turned on or off depending on the level of fidelity desired. To generate data representative of truth all options are enabled which included a spherical harmonics model with degree and order 100, third body effects from the Sun, Moon, and eight other planets (or dwarf planets), atmospheric drag and lift, and solar radiation pressure (SRP). A variable step Runge-Kutta eighth-order integrator is the highest-order integrator type provided by the software and is utilized for the simulations. A six-dimensional state vector of position and velocity in the Mean of J2000 Earth Ecliptic reference frame is output for purposes of comparison to the various force models. Since the focus of this research is on CubeSat applications, a Sun-synchronous orbit is the best fit for a reference orbit; Table 1 lists the initial orbital elements. Note that these orbital elements are expressed in the Mean of J2000 Earth Ecliptic reference frame. A visualization of the reference orbit is provided in Figure 2.

To briefly define, a Sun-synchronous orbit is one in which the line of nodes is fixed with respect to the Sun. The orbital elements are selected such that the J_2 secular effects alter the longitude of ascending node (Ω) at the same rate as the Earth's orbit revolution rate (approximately $\frac{1^\circ}{day}$. These

Table 1: Initial Orbital Elements for Generating "Truth" Data.

a (km)	e	i (deg)	Ω (deg)	ω (deg)	ν (deg)
7083.1364	0.0100	98.2067	280.4003	0.0000	0.0000

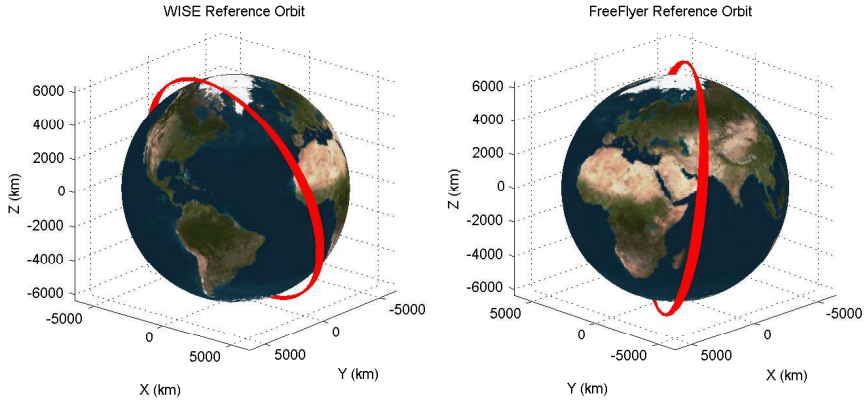

Figure 2: View of WISE orbit (left) and *FreeFlyer* reference orbits over Jan 01, 2011 through Jan 04, 2011. Earth displayed just for visualization of inclinations.

orbits are popular for mission scenarios requiring solar panel alignment or, more commonly, consistent lighting conditions on the Earth from one orbit to the next [18, 66]. Sun-synchronous orbits are near-polar and for most sensors can provide complete global coverage. Global coverage simplifies ground communications, allowing university and private CubeSats to take advantage of the international amateur radio community for purposes of tracking and data downlink. Furthermore, using a near-polar orbit for simulations ensures that the CubeSat will experience the full range of non-spherical perturbation effects. Examining unofficial lists of CubeSat missions further indicates that Sun-synchronous orbits are the most common among target mission orbits for CubeSats [1, 64].

The second set of data used for comparison is acquired from JPL's HORIZONS on-line ephemeris system. The HORIZONS web service is accessible in multiple manners, but for obtaining the WISE data the web-interface (provided at [12]) is utilized to export a plain text document; this document is parsed with VBA to generate a CSV. JPL's WISE spacecraft was launched on December 14, 2009 into a Sun-synchronous orbit with an altitude of 525 km. The primary mission was a sky survey including identifying the most luminous galaxies in the universe and finding the stars closest to our own Sun. A spacecraft with a Sun-synchronous orbit is selected for the same reasons as stated above. Using HORIZONS, ephemeris data with one-minute increments during the first few days of 2011 is obtained for comparison. A visualization of the WISE spacecraft orbit is provided in Figure 2.

In general the HORIZONS data could be considered more accurate than a high-fidelity propagation since it is determined using real measurements from on-board sensors and tracking ground stations. As a means to determine the accuracy of *FreeFlyer*'s orbit propagation, the initial conditions for the WISE spacecraft are used within *FreeFlyer*. The same full force model as described above is selected. Figure 3 displays the difference for each state element over the course of ten orbits. Note the discontinuity just before the completion of one orbit; this is likely due to some correction included in the HORIZONS data from ground tracking or other methods, not captured by *FreeFlyer*. The discontinuity may be the result of a maneuver conducted by WISE during the one-minute interval. A second possibility is that HORIZONS interpolation has discrete boundaries which is captured at this instant in time. Even over the course of ten orbits, however, the 2-*norm* of the position error is on the order of hundreds of meters and the 2-*norm* of the velocity error is on the

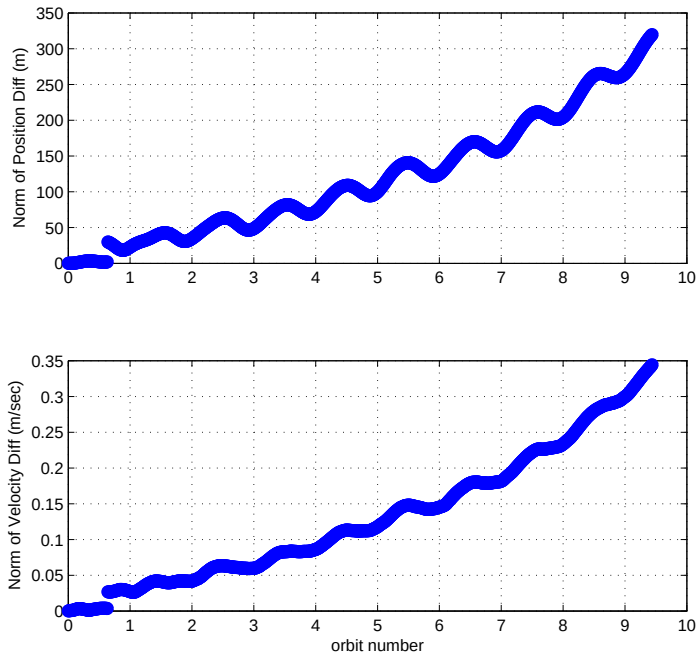

Figure 3: Comparison of WISE Orbit from HORIZONS versus propagated with *FreeFlyer* over ten orbits.

order of decimeters per second; examining just the first half-orbit (prior to the discontinuity), the difference is only on the order of meters and centimeters per second. For clarity Figure 4 displays just this portion of the original Figure 3.

2.2 Propagation Techniques

Prior to investigating the level of force model fidelity required, a trade study on propagation techniques is provided. Three primary techniques are considered: relative motion, variation of parameters, and classic Cartesian. The following sections discuss the advantages and disadvantages of each technique and provide an argument for the technique selected.

2.2.1 Relative Motion

In Chapter 1 details on relative motion models for LEO applications are provided. Although for a collaborative application the relative position of spacecraft team members may be the desired frame of reference, there are also disadvantages to propagating in the relative frame. Chapter 3 discusses the on-board orbit determination algorithm implemented, which utilizes an extended Kalman filter

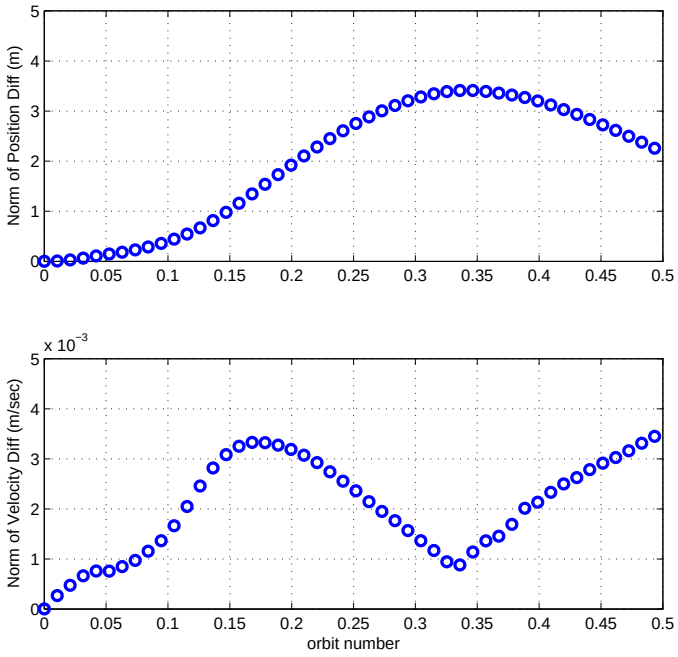

Figure 4: Comparison of WISE Orbit from HORIZONS versus propagated with *FreeFlyer* over half an orbit.

(EKF). In short, the EKF provides a means to combine noisy sensor data and state predictions through propagation in order to create a more accurate estimate of the actual state. In order to accomplish this a mapping between the sensor output and desired state parameters is necessary. Although this is possible, the use of high TRL (Technology Readiness Level) COTS LEO GPS receivers is a far simpler solution.

2.2.2 Variation of Parameters - Lagrange Planetary Equations

Rather than integrating the spacecraft's Cartesian state, the Lagrange Planetary Equations provide equations for the rates of change for the six common orbital elements over time. Like any standard integration scheme, the LPEs can include or exclude perturbation effects in order to simplify calculations or improve accuracy. LPEs help with visualizing what is actually happening to an orbit over time since five of the six orbital elements define the shape of the orbit while the sixth provides the current location on the orbit. For example, seeing changes in the semi-major axis, eccentricity, or inclination is far simpler than picturing the same effect when examining Cartesian coordinates with

respect to time.

As with the relative motion technique, one issue when taking the EKF into consideration is that orbital elements cannot be directly measured. Mapping between a Cartesian output from a GPS receiver to orbital element space is rather straight forward, but not as simplistic as a one-to-one mapping available if in Cartesian space to begin with. Furthermore, although easy to visualize, an autonomous cluster of picosatellites will likely require either relative or Cartesian position information to plan data collection.

2.2.3 Cartesian

Integration in a Cartesian frame, specifically a frame like Mean of J2000 Earth Ecliptic, is probably the most straight forward technique. Some of the simplicity comes from only needing to calculate three derivatives (the accelerations or rather the derivatives of the velocities) since the other three are provided in the current state. Another advantage that will be more obvious in Chapter 3's discussion of the on-board orbit determination is the direct mapping of GPS measurements to the state; the COTS receivers designed for LEO applications ouput state in the same Mean of J2000 Earth Ecliptic reference frame being used for propagation.

Lastly, calculating relative positions of satellites based on two individual Cartesian coordinate is more straight-forward than comparing two sets of orbital elements. In order to compare two orbital element vectors, first transforming the orbital element space to Cartesian is necessary. Propagation in the Cartesian system is selected as the primary technique.

2.3 Force Models

2.3.1 Non-Spherical Effects

Several missions over the past decade have carried payloads for geodetic goals, geodesy being a branch of science concerned with measuring the exact size and shape of the Earth's surface and its gravitational field. The data collected from these missions is used for the monitoring of ocean levels, crustal movement, and generating high fidelity gravitational force models for satellite motion [23, 38]. One mission of note is the Gravity Recovery and Climate Experiment more well-known as GRACE. The GRACE mission is a collaborative effort between NASA's Jet Propulsion Laboratory (JPL) in Pasadena, CA, the German Aerospace Center (DLR), and the University of Texas' Center

for Space Research (CSR)[62]. Launched in 2002 with a primary 5-year mission, GRACE has generated a spherical harmonics model of the Earth's gravitational field an order of magnitude more accurate than any preceding model. Models of degree and order 120 or greater are available to the science community [32]. These models are utilized to incorporate perturbation effects due to a non-spherical Earth in the orbit force model.

In order to generate the varying fidelity of models desired, a single set of data for a 200 x 200 field is obtained from JPL. Although for the highest level of accuracy at each degree and order new coefficients should be calculated, it is a common practice to truncate the matrices to the desired size. The disturbing effects for the higher-order terms are of small enough significance that improvement in a gravitational model is still achieved through truncation rather than recalculating new model coefficients from raw data. For model accuracy analysis four different levels of fidelity are explored with degree 2 (this is just the Earth oblateness described by J_2) and then order and degree 4 and 20. A brief discussion of spherical harmonics immediately follows, complemented by the results of the four force models aforementioned.

2.3.1.1 Spherical Harmonics

For planetary bodies, all which can be closely approximated with a sphere, it is convenient to express the variations of the gravitational potential in a spherical coordinate system [40, 66]. Spherical harmonics are divided into three broad categories: zonal, sectorial, and tesseral. Zonal harmonics occur when the order (m) is 0 and can be viewed as bands that follow lines of latitude. For any given degree (l), there are $l+1$ bands. For example, the J_2 zonal harmonic has been previously mentioned. When this term is included for the Earth's gravitational potential, it is as if the the Earth is wearing a belt of mass around the equator. The bands that include the north and south poles are of lesser mass while the band that includes the equator is of greater mass. Sectorial harmonics occur when the degree and order are the same ($l = m$) and can be thought of as vertical bands that follow lines of longitude. There are $2 * l$ bands for each sectorial harmonic. Finally, when $m > 0$ and $l \neq m$ a tesseral harmonic occurs. Tesseral harmonics appear as checkerboard patterns with $(l + 1) * (2 * m)$ boxes [66]. Figures 5, 6, and 7 provide examples of each type just described.

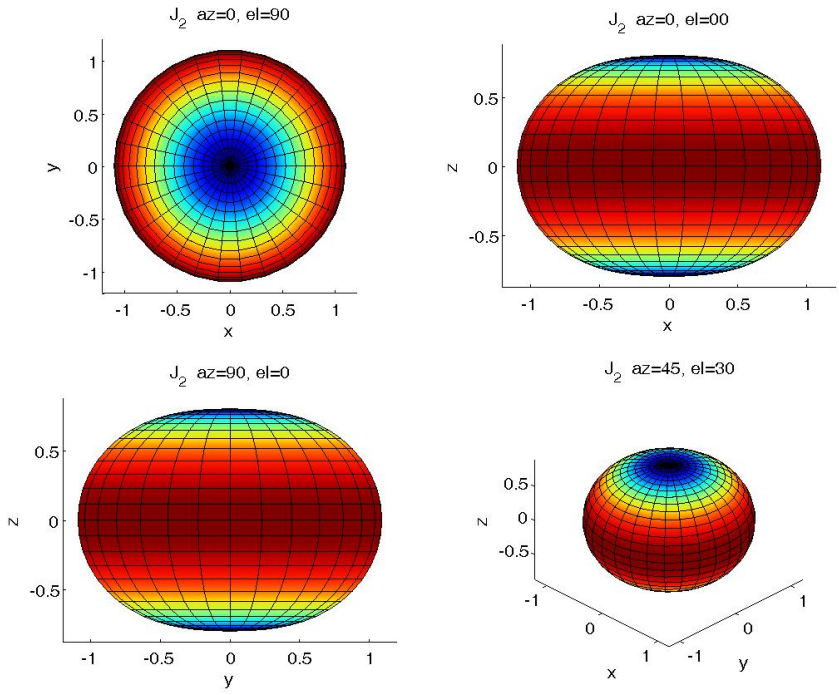

Figure 5: Exaggerated example of spherical harmonic zonal term. [57]

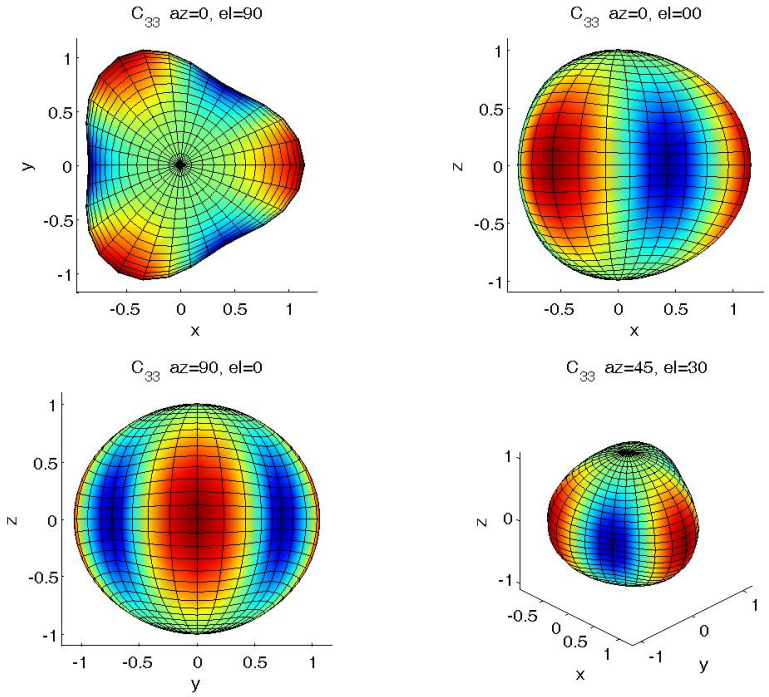

Figure 6: Exaggerated example of spherical harmonic sectorial term. [57]

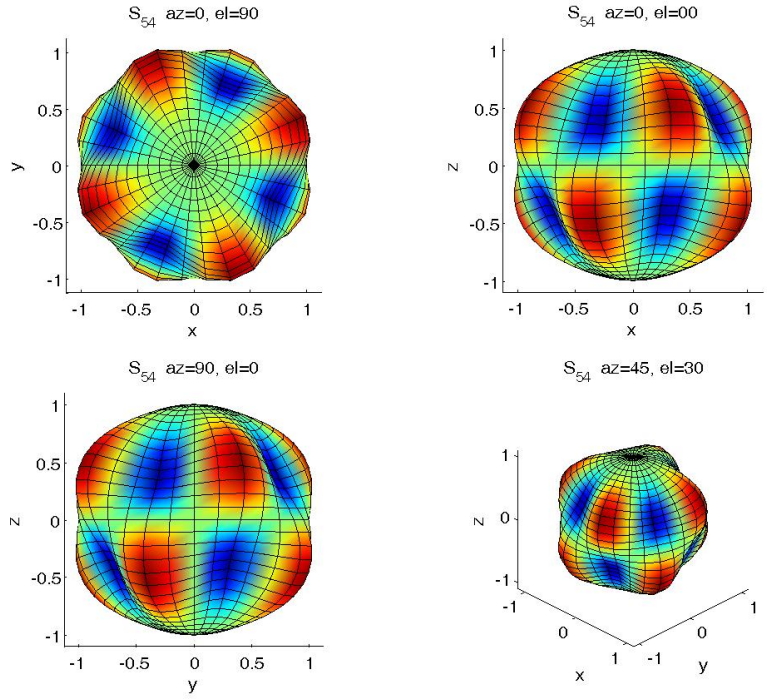

Figure 7: Exaggerated example of spherical harmonic tesseral term. [57]

Combining these three types of harmonics results in a high-fidelity model of the Earth's gravitational field. Once the $l * m$ coefficients are determined, calculating the potential is relatively easy using Equation (2).

$$U = \frac{\mu}{r}\sum_{l=0} P_l\left[\sin(\phi_{gc_{sat}})\right]\left(\frac{R_\oplus}{r}\right)^l C_{l,0} + \frac{\mu}{r}\sum_{l=1}^{\infty}\sum_{m=1}^{l} P_{l,m}\left[\sin(\phi_{gc_{sat}})\right]\left(\frac{R_\oplus}{r}\right)^l \{C_{l,m}\cos(m\lambda_{sat}) + S_{l,m}\sin(m\lambda_{sat})\} \quad (2)$$

For clarification the parameters for Equation (2) are defined in Table 2.

$$J_l = -C_{l,0} \quad (3)$$

Examining Equation (2) shows that the first summation is simply the zonal terms since m is always equal to 0; therefore, it is common to refer to that set of coefficients by vector J rather than the first column of the matrix C. Note, however, that J is equivalent to the negative of $C_{l,0}$ as displayed by Equation(3) [66]. The second double sum provides the gravitational potential for both

Table 2: Parameter Definitions for the Spherical Harmonic Potential in Equation (2).

Parameter	Description
μ	Gravitational parameter of body equal to the body mass times the gravitational constant, G
r	Radius from center of body to spacecraft
$\phi_{gc_{sat}}$	Geocentric latitude position of spacecraft
λ_{sat}	Longitude position of spacecraft
$R_{\oplus}$	Reference radius of body
$P_l\left[\sin(\phi_{gc_{sat}})\right]$	Legendre Polynomial with argument $\sin(\phi_{gc_{sat}})$
$P_{l,m}\left[\sin(\phi_{gc_{sat}})\right]$	Associated Legendre Polynomial with argument $\sin(\phi_{gc_{sat}})$
$C_{l,m}$	Lower-triangular matrix of gravitational coefficients
$S_{l,m}$	Lower-triangular matrix of gravitational coefficients

the sectorial and tesseral terms. To reiterate an important clarification of the potential equations, the $P_l\left[\sin(\phi_{gc_{sat}})\right]$ and $P_{l,m}\left[\sin(\phi_{gc_{sat}})\right]$ are Legendre Polynomials where $\sin(\phi_{gc_{sat}})$ is the input argument for the polynomial. Legendre Polynomials can be calculated explicitly from Equations (4) and (5) (by replacing x with the appropriate argument), calculated recursively from Equations (6) through (11), or are readily available from a number of sources [20].

$$P_l\left[x\right] = \frac{1}{2^l l!}\frac{d^l}{dx^l}\left(x^2-1\right)^l \tag{4}$$

$$P_{l,m}\left[x\right] = \frac{1}{2^l l!}\left(1-x^2\right)^{m/2}\frac{d^{l+m}}{dx^{l+m}}\left(x^2-1\right)^l \tag{5}$$

$$P_l\left[\sin(\phi)\right] = \frac{(2n-1)\sin(\phi)P_{l-1}\left[\sin(\phi)\right] - (n-1)\,P_{n-2}\left[\sin(\phi)\right]}{n} \tag{6}$$

$$P_{0,0}\left[\sin(\phi)\right] = 1 \tag{7}$$

$$P_{1,0}\left[\sin(\phi)\right] = \sin(\phi) \tag{8}$$

$$P_{l,l}\left[\sin(\phi)\right] = (2n-1)\cos(\phi)P_{l-1,l-1}\left[\cos(\phi)\right] \tag{9}$$

$$P_{1,1}\left[\sin(\phi)\right] = \cos(\phi) \tag{10}$$

$$P_{l,m}\left[\sin(\phi)\right] = \left[\frac{1}{l-m}\right]\left[(2n-1)\sin(\phi)P_{l-1,m}[\sin(\phi)] + (l+m-1)P_{l-2,m}[\sin(\phi)]\right] \tag{11}$$

Once the gravitational potential is calculated, there is still the need to calculate the partial derivatives in spherical coordinates in order to find the forces. Although the derivatives are taken with

respect to spherical coordinates, the accelerations can be defined in Cartesian coordinates by using the chain rule. Note that the accelerations are first calculated in the **body-fixed** Cartesian frame and are then rotated to the inertial frame used for numerical integration [66]. The acceleration in body-fixed Cartesian coordinates can be found using Equation (12).

$$\vec{a} = \frac{\partial U}{\partial r}\left(\frac{\partial r}{\partial \vec{r}}\right)^T + \frac{\partial U}{\partial \phi_{gc_{sat}}}\left(\frac{\partial \phi_{gc_{sat}}}{\partial \vec{r}}\right)^T + \frac{\partial U}{\partial \lambda_{sat}}\left(\frac{\partial \lambda_{sat}}{\partial \vec{r}}\right)^T \tag{12}$$

For the purposes of generating the necessary equations quickly, **Wolfram**'s *Mathematica* software package provides a convenient method. With this package, the eight lines of code in Listing (2.1) can be used to generate the body-fixed Cartesian acceleration equations for any order and degree (where $l = m$). In the list the "4" in the third line indicates the degree and order; modifying this to any number will generate output for a different degree and order. Note that with the output from Listing (2.1) the C and S matrices are still required to calculate the acceleration at a given geocentric latitude (ϕ_{gc}), longitude (λ), and radius (r) in Earth-centered Earth-fixed coordinates.

Listing 2.1: The following code can be used in **Wolfram**'s *Mathematica* to generate body-fixed Cartesian gravitational accelerations due to the non-sphericity of a central body

```
Usph = Sum[((Rref/r)^1)*((-1)^m)*LegendreP[1, m, Sin[phi]]
      * (C[1, m]*Cos[m*lambda]
            + S[1, m]*Sin[m*lambda]), {1, 2, 4}, {m, 0, 1}];
U = (MU/r)*(1 + Usph);
Ur = D[U, r];
Uphi = D[U, phi];
Ulambda = D[U, lambda];
aI = ((((1/r)*Ur) - ((z/(r^2*Sqrt[x^2 + y^2]))*Uphi))*x)
   - ((Ulambda/(x^2 + y^2))*y);
aJ = ((((1/r)*Ur) - ((z/(r^2*Sqrt[x^2 + y^2]))*Uphi))*y)
   - ((Ulambda/(x^2 + y^2))*x);
aK = ((z/r)*Ur) + (Uphi*Sqrt[x^2 + y^2]/r^2);
```

2.3.1.2 J_2 (*Degree* 2, *Order* 0)

The simplest non-spherical gravitational model of the Earth to consider is represented by a single coefficient, most commonly referred to as J_2. An estimate of this value is provided in Equation (13). This value corresponds to the oblateness of the Earth, mostly caused by the rotation of Earth about its axis. J_2 effects result in sinusoidal motion for all six primary orbital elements discussed earlier, but additionally causes secular drift for the argument of periapse (ω) and the longitude of ascending node (Ω). Equations (14), (15), and (16) provide the set of three equations for calculating the force on a satellite while include Earth oblateness. Since the zonal harmonics are axi-symmetric, only the spacecraft latitude is required which is simply a function of position, not time. For the sectorial and tesseral harmonics represented in the higher order models longitude is additionally required; longitude is a function of time and the Earth's angular position at the initial epoch.

$$J_2 = 0.0010826269 = -C_{2,0} \tag{13}$$

$$\ddot{x} = -\frac{\mu}{r^3}x + \frac{3J_2\mu R_{ref}^2}{r^5}\left(\frac{1.5}{r^2}z^2 - 0.5\right)x + \frac{3J_2\mu R_{ref}^2 z^2}{r^7}x \tag{14}$$

$$\ddot{y} = -\frac{\mu}{r^3}y + \frac{3J_2\mu R_{ref}^2}{r^5}\left(\frac{1.5}{r^2}z^2 - 0.5\right)y + \frac{3J_2\mu R_{ref}^2 z^2}{r^7}y \tag{15}$$

$$\ddot{z} = -\frac{\mu}{r^3}z + \frac{3J_2\mu R_{ref}^2}{r^5}\left(\frac{1.5}{r^2}z^2 - 0.5\right)z + \frac{3J_2\mu R_{ref}^2}{r^3}\left(\frac{z}{r^2} - \frac{z^3}{r^4}\right) \tag{16}$$

For compactness all error analysis is provided in the last sub section of this section.

2.3.1.3 *Degree and Order* 4 *and* 20

Common subsets of the large non-spherical potential are a 4x4 or 20x20 grid. Using the *Mathematica* code explained above, the acceleration equations are calculated for each force model. Unlike the J_2 model, these equations are far more extensive and are not included here. Spherical harmonic coefficients for order and degree 20 are listed in Tables 10, 11, and 17 located in Appendix A. Recall that for lower order models, such as the 4x4, these vectors/matrices are simply truncated to the size required.

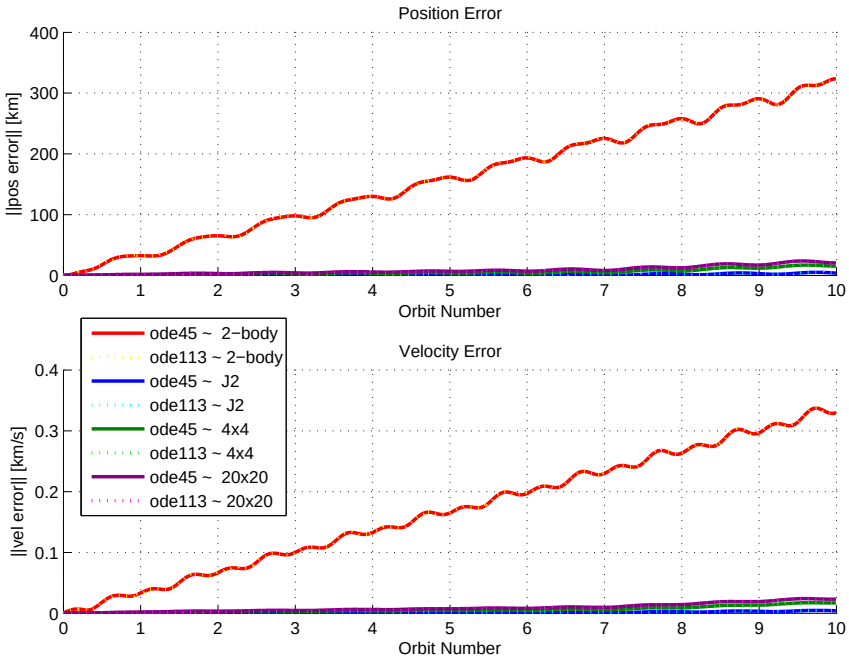

Figure 8: Comparison of Error from the Two-Body, J_2, 4x4, and 20x20 Force Models compared to *FreeFlyer* Reference.

Just from examining the set of coefficients, the Earth's oblateness is shown to be at least three orders of magnitude larger than the remaining values. Furthermore, as the degree and order increase the non-normalized values used in the previously discussed equations approach machine precision resulting in loss due to rounding. As will be further discussed in the following analysis section, these results encourage the use of a simple gravitational model that only includes the two-body and J_2 terms.

2.3.1.4 Error Analysis and Summary

With the output from *Mathematica* the three different non-spherical potential models are implemented for experimentation. Figures (8) and (10) display the 2-*norm* of the errors for position and velocity compared to the *FreeFlyer* and WISE reference orbits, respectively. Notice that both the `ode45` and `ode113` *MatLab* integrators are utilized to verify any error is a result of the model as opposed to selected integration method; nearly identical results are found with both integrators when using an error tolerance of 1×10^{-13}.

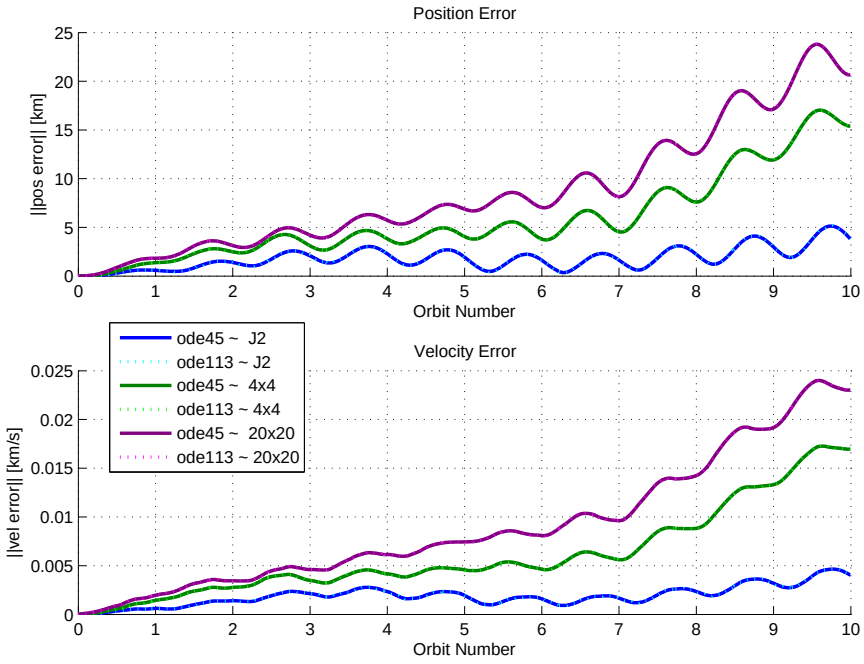

Figure 9: Comparison of Error from the J_2, 4x4, and 20x20 Force Models compared to *FreeFlyer* Reference.

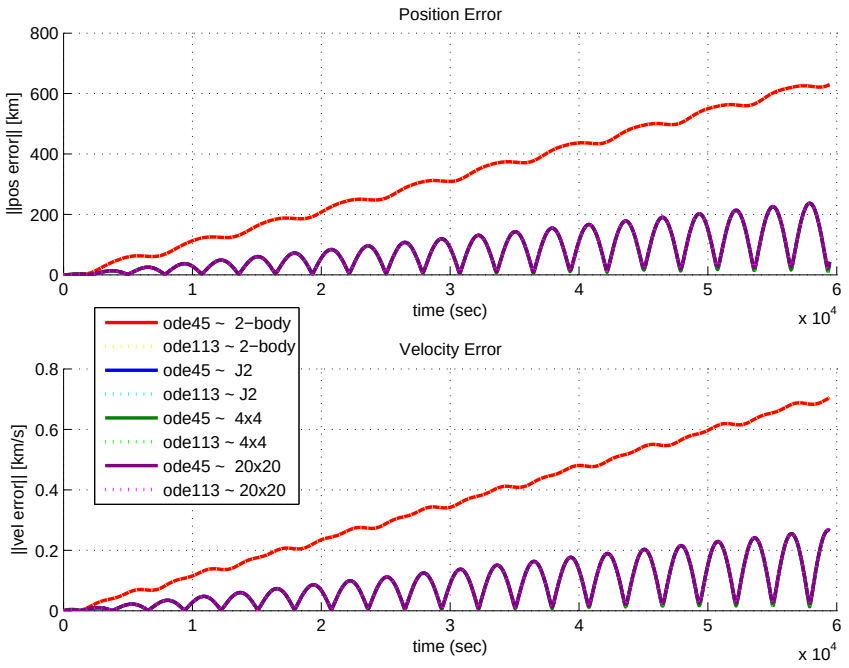

Figure 10: Comparison of Error from the Two-Body, J_2, 4x4, and 20x20 Force Models compared to WISE Reference.

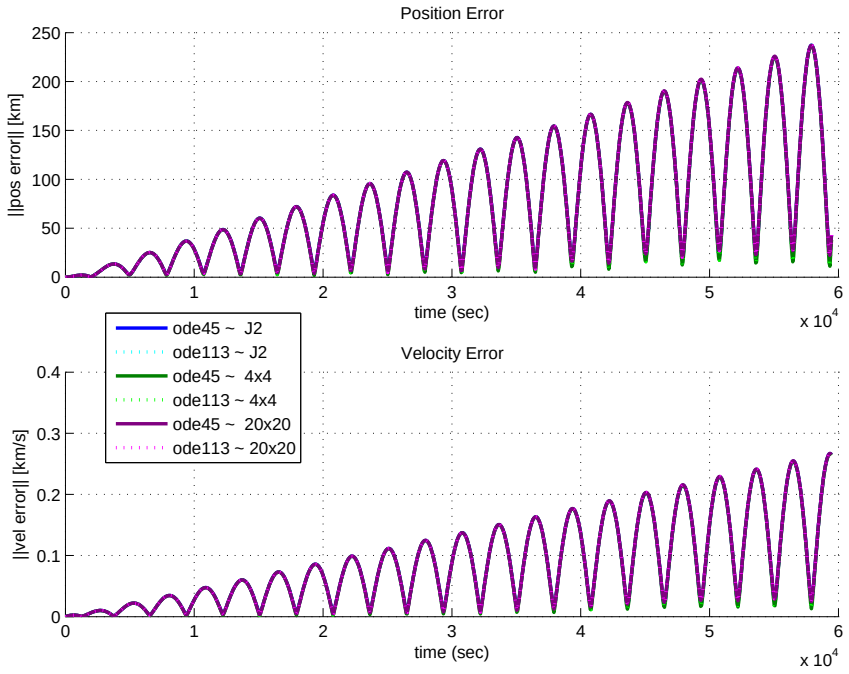

Figure 11: Comparison of Error from the J_2, 4x4, and 20x20 Force Models compared to WISE Reference.

As expected for all cases, the two-body force model gives the worst accuracy. The three non-spherical force models, however, all display relatively similar performance with the most accurate model being dependent on the reference orbit. To provide a little more clarity Figures (9) and (11) show the same data as Figures (8) and (10), respectively, but with the two-body force model data removed. For the *FreeFlyer* reference, the three models show similar performance with the simplest, J_2 only, displaying the least amount of error over the longer term. When compared to the WISE reference orbit, little variation can be seen. Examining the plots very closely shows that the 4x4 model provides the least amount comparative error followed by the 20x20 and finally J_2. The fact that the lower order 4x4 model appears slightly more accurate than the 20x20 may be the result of the model used by the HORIZONS ephemeris service for interpolation. Additionally, the higher order models demonstrate reduced accuracy since the other pertubations are not being included, causing the error to drift more rapidly. Since the WISE orbit is of lower altitude, 525 km compared to 700 km, the fact that the Earth oblateness term is at least three orders of magnitude greater than the other non-spherical terms is magnified, likely the cause for the errors to be relatively the same.

It is important to note that the error with respect to the WISE reference is significantly greater than the *FreeFlyer*; this again can be contributed to the altitude; at a lower altitude the effects of drag are greater. Since the WISE data is from a flight mission there is also the possibility that the spacecraft conducted small propulsive maneuvers for orbit maintenance over the course of the approximate ten orbits. It is still worth comparing since it is known no large maneuvers were conducted (as this would be noticeable in the obtained data), but such data will be more valuable when testing the orbit determination algorithms in Chapter 3. The osculating error seen in the WISE comparison is also worth noting; since the error returns to near zero every 45 minutes (about half an orbit) some periodic force is absent from our model. While the drag could explain the secular increase in amplitude, the return to reduced error may be explained by gravitational potential perturbations due to tidal changes, third-body perturbation, or solar radiation pressure. This will be further examined in the following sections.

In short, the complexity and cost of using a higher-order potential beyond that of the simple J_2 model is deemed unnecessary for this application. Reduction in the cost of higher-order fields can be achieved, however, through the use of a recursive approach. In the implementation described above, a non-recursive approach is taken due to the simplicity of generating equations of motion from *Mathematica*. This allowed for analyzing the accuracy of the higher fidelity models rapidly. If better accuracy is deemed necessary, a recursive implementation (as discussed in section 2.3.1.1 with Equations (6) through (11)) would be employed which has been shown as better performing [66]. Returning to the need for balanced software solution, the computational costs (using the non-recursive equations) as determined by flop count is provided in Figure 12. Note the logarithmic ordinate scale. The flop counts have been determined in two different ways: first each operation, whether a trigonometric or arithmetic call, is considered a single flop. A slightly more accurate estimate is developed by estimating the cost of each operation through timing the execution of a large number of complex operations and averaging. Table 3 provides the estimates for each "atomic" operation; although the operations listed are not all technically atomic, they are considered as such for the purposes of calculating a flop count estimate. Figure 12 clearly shows linear growth on the logarithmic scale, indicating exponential growth in flop count. Note that the two-body and J_2 force model computational costs are on the same order of magnitude while the difference between

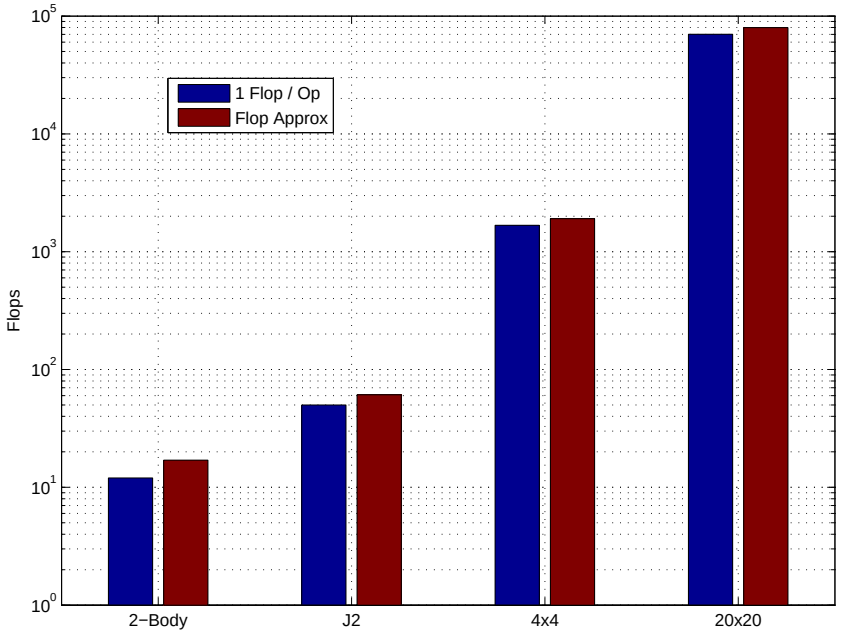

Figure 12: Estimate of Flop Counts per Derivative Call for each Force Model.

the J_2 and 4x4 force model is greater than an order of magnitude. Considering this exponential growth, while recalling the respective differences in error, it is clear that by selecting the simplest non-spherical model the greatest return on computational invest is realized. Visually, imagine computational cost is the abscissa axis and accuracy the ordinate axis. The curve relating these two parameters would have an elbow indicating the diminishing "rate of return" in accuracy considering "computational investment".

In summary, the decision to select the simplest non-spherical J_2 only model is clear. Since the error over the course of ten orbits (a period of time greater than required since on-board orbit determination will be utilized) is still on the order of kilometers and meters per second, there is room for model improvement. For this reason drag, solar radiation pressure, and third-body effects are discussed in the following sections to supplement the primary gravitational force model.

2.3.2 Drag

In LEO another strong and relatively consistent perturbation, for near-circular orbits, is drag. Calculating the force due to drag requires knowledge of a spacecraft's coefficient of drag (c_D), current

Table 3: Flop Count Estimate for "Atomic" Operations.

Operation	**Approx. Flops**
$x_1 + x_2$	1
$x_1 - x_2$	1
$x_1 * x_2$	1
$\frac{x_1}{x_2}$	2
$\frac{x}{2^y}$ †	1
$\sqrt{x}$	5
$\sin x$	5.5
$\cos x$	5.5
y^x	$5 - 20x^{\ddagger}$
†: division by some power of 2	
‡: required 5 to 20 times longer than x multiplications	

mass (m), cross-sectional area perpendicular to the velocity vector (A), and velocity relative to the atmosphere ($\vec{v}_{rel}$). Additionally, the atmospheric density (ρ) at the spacecraft's current position is required. The relationship between these variables is provided in Equation (17).

$$\vec{a}_{drag} = -\frac{1}{2}\frac{c_D A}{m}\rho v_{rel}^2 \frac{\vec{v}_{rel}}{|\vec{v}_{rel}|} \tag{17}$$

For the problem at hand some of these variables can be considered constant, since there is sufficient variability in others (such as density) that high precision for only portions of this calculation are moot. Before describing these constants, make note that once on-board orbit determination is brought into consideration, these values no longer remain "constant". The cross-sectional area of a 3U spacecraft can be assumed to be either 100 cm^2 or 300 cm^2 depending on the mission attitude requirements. If the CubeSat is utilizing a passive gravity-gradient stabilization system, with the long axis generally aligned with nadir pointing, then the 300 cm^2 area is fitting; likewise for a spacecraft using a passive magnet system such that the long axis is aligned with the magnetic field, during most of the orbit a 100 cm^2 area will face the ram direction. More precision could be obtained by using the state's attitude quaternion, however error in the variables described shortly indicate this is not necessary. Mass can safely be assumed constant for a CubeSat which has no propulsion system.

Unless a mission requires the departure of some component from the spacecraft, collecting an accurate mass measurement prior to launch is sufficient. One final variable that could be considered constant is the drag coefficient. For a flat plat model in LEO atmosphere a c_D of about 2.2 is used in practice [42, 66]. A spherical spacecraft may have a slightly lower coefficient between 2.0 and 2.1.

As suggested in the previous paragraph, although these values are constant for a given propagation period, on-board orbit determination provides a means to better estimate these values. A good practice is to take the c_D, A, and m variables and create a single constant: $\frac{c_D A}{m}$. You may notice that this relationship is the inverse of the common ballistic coefficient [66]. Chapter 3 provides details on the adding of this inverse ballistic coefficient to the state vector and updating its estimate through the use of an extended Kalman filter.

Previously the $\vec{v}_{rel}$ term was defined as the velocity relative to the atmosphere. This is an important clarification as it is not the same velocity maintained in the state vector. Due to boundary conditions, the atmosphere's velocity, even at LEO altitudes, is close to the velocity of the surface of the Earth. Many models exist for calculating an average atmosphere velocity for a given latitude, longitude, and altitude. Since there is significant variation from wind, however, it is common to ignore the use of such models and simply assume the velocity of the atmosphere is equal to the Earth's spin-rate [66]. In order to calculate $\vec{v}_{rel}$ from the state, Equation (18) is utilized [66]. Long term motion, over the course of one or multiple orbits, is the primary goal for LEO simulations. Variations in high atmospheric wind will, in general, not generate significant secular variations over the course of an orbit, allowing this simplification with regards to velocity to be considered.

$$\vec{v}_{rel} = \frac{d\vec{r}}{dt} - \vec{\omega}_{\oplus} \times \vec{r} \tag{18}$$

Atmospheric density is the final variable yet to be discussed. Like atmospheric wind there are many different models for calculating an approximate density. Model complexity varies from simply considering altitude to also including the effects of current solar flux and the Earth's magnetic field. Although including solar flux or mean changes in the Earth's magnetic field could assist in improved long-term models (on the order of weeks and months, not orbits), it is the unpredictable short period variations which would cause improvement in accuracy for propagation on the order of orbits. With such data being unavailable, a standard piece-wise continuous exponential model is

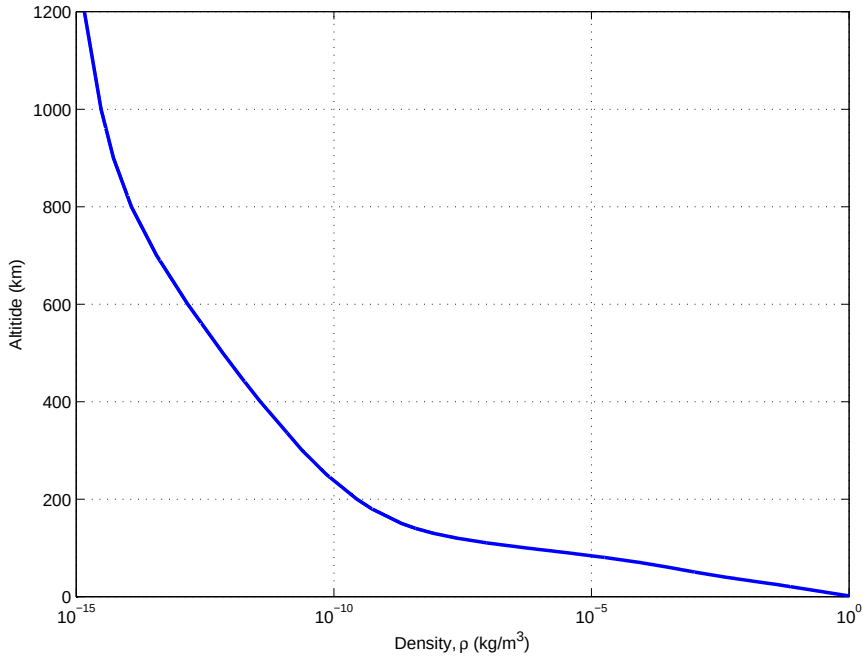

Figure 13: Piece-wise continuous curve used to determine atmospheric density soley as a function of altitude.

selected for determining local atmospheric density. Using Equation (19), Table 4, and the current altitude determined from the state vector and estimate of Earth's radius, a reasonable estimate can be procured for the final drag calculation [66]. Figure 13 provides the curve generated for altitudes ranging from 0 to 1, 200 km.

$$\rho = \rho_0 e^{-\frac{h_{ellp}-h_0}{H}} \tag{19}$$

The first task in testing this model is to ensure that the semi-major axis of the orbit decreases with time. This is an easy check since a force model composed of only the two-body gravity potential and drag, the total energy of the spacecraft will decrease at each time step and therefore lose altitude. Instantaneously, the spacecraft is losing velocity due to the friction from drag. However, since the altitude (and therefore the semi-major axis) is decreasing, the overall velocity of the spacecraft increases: this is commonly known as the "drag paradox" [66]. Figure 14 shows that over the course of one year the semi-major axis decreases by about 15 km. For this simulation a mass of 4 kg and c_D equal to 2.2 is utilized; the initial state is obtained from the WISE reference data. The

Table 4: Piece-Wise Exponential Atmospheric Model used in Equation 19 ([66]).

Altitude, h_{ellp} (km)	**Base Alt.,** h_0 (km)	**Nominal Density,** ρ_0 (kg/m^3)	**Scale Heigh,** H (km)
0 – 25	0	1.225	7.249
25 – 30	25	3.899×10^{-2}	6.349
30 – 40	30	1.774×10^{-2}	6.682
40 – 50	40	3.972×10^{-3}	7.554
50 – 60	50	1.057×10^{-3}	8.382
60 – 70	60	3.206×10^{-4}	7.714
70 – 80	70	8.770×10^{-5}	6.549
80 – 90	80	1.905×10^{-5}	5.799
90 – 100	90	3.396×10^{-6}	5.382
100 – 110	100	5.297×10^{-7}	5.877
110 – 120	110	9.661×10^{-8}	7.263
120 – 130	120	2.438×10^{-8}	9.473
130 – 140	130	8.383×10^{-9}	12.636
140 – 150	140	3.845×10^{-9}	16.149
150 – 180	150	2.070×10^{-9}	22.523
180 – 200	180	5.464×10^{-10}	29.740
200 – 250	200	2.789×10^{-10}	37.105
250 – 300	250	7.248×10^{-11}	45.546
300 – 350	300	2.418×10^{-11}	53.628
350 – 400	350	9.518×10^{-12}	53.298
400 – 450	400	3.725×10^{-12}	58.515
450 – 500	450	1.585×10^{-12}	60.828
500 – 600	500	6.967×10^{-13}	63.822
600 – 700	600	1.454×10^{-13}	71.835
700 – 800	700	3.614×10^{-14}	88.667
800 – 900	800	1.170×10^{-14}	124.64
900 – 1000	900	5.245×10^{-15}	181.05
1000–	1000	3.019×10^{-15}	268.00

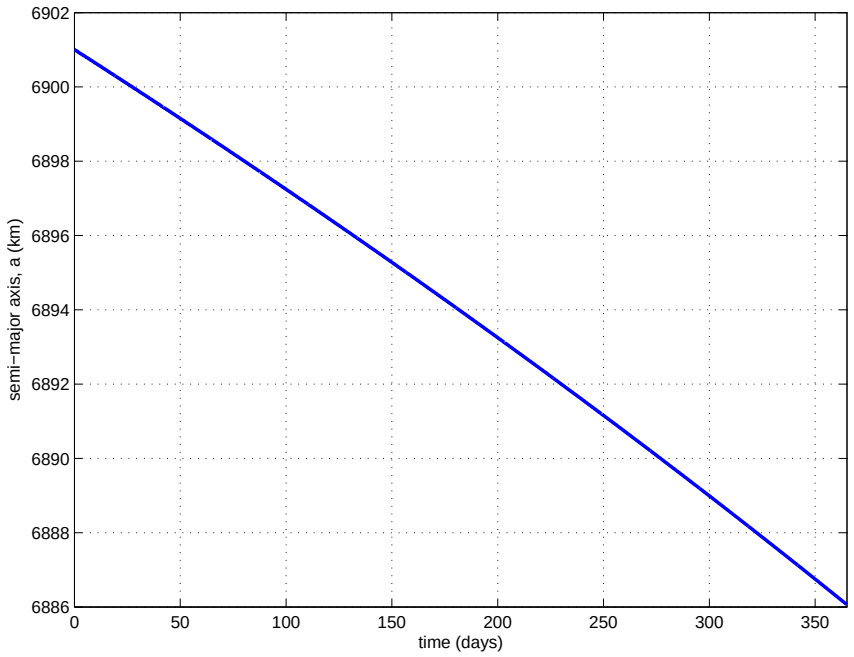

Figure 14: A decreasing semi-major axis validates the implemented drag force model since it is consistently reducing the CubeSat's total energy.

shape of the curve also fits intuition: rather than decreasing in a linear fashion, the negative slope gradually increases. As the altitude drops, the atmosphere becomes more dense, further increasing the drag force. If this is propagated long enough, the curve would become very steep and the spacecraft would crash into the Earth. Before providing additional error analysis results, two additional perturbation forces need to be added to our model: solar radiation pressure and third-bodies. In short, however, adding drag to the current J_2 model did improve the *FreeFlyer* error: at the end of ten orbits the position error is improved by about 350 m and the velocity error by about 0.35 $^{m}/_{s}$.

2.3.3 Solar Radiation Pressure

Energy from the Sun while in LEO is not obtained without some inconvenience. That same solar flux (also referred to as intensity or irradiance) that is beneficial to be as large as possible for power generation also applies a small perturbing force to the spacecraft. Although the solar irradiance fluctuates in both short- and long-period manners, such as sudden flares or the 11-year cycle respectively, it averages 1367 $^{W}/_{m^2}$. For power subsystem estimation the product of this solar flux value, solar panel area, cosine of the incidence angle, and solar panel efficiency provides an estimate on

the power generated when in sunlight. Likewise, the solar radiation force can be calculated from the product of the solar pressure (p_{SR}), spacecraft reflectivity (c_R), and spacecraft area exposed to the sunlight ($A_\odot$). First an estimate of solar pressure can be calculated assuming a constant solar flux and c, the speed of light; Equation (22) provides the solar pressure value at LEO [66].

$$p_{SR} = \frac{SolarFlux}{c} = \frac{1367\ W/m^2}{3 \times 10^{+8}\ m/s} \tag{20}$$

$$= 4.559821181358738 \times 10^{-6}\ \frac{W \cdot s}{m^3} \tag{21}$$

$$= 4.559821181358738 \times 10^{-6}\ \frac{N}{m^2} \tag{22}$$

Like in the case of calculating drag, it may be acceptable to assume a constant area. However, by utilizing the state vector quaternion a more exact calculation could be determined. In the reference case of a 3U CubeSat in a 13:30 Sun-synchronous orbit, setting the area constant to be 3×10^{-2} is reasonable. When the CubeSat is crossing the equator, this estimate is too large. However, the error in the area calculation can be captured by allowing estimation of the reflectivity constant c_R, or rather, a parameter like that used for the drag calculation: $\frac{p_{SR} c_R A_\odot}{m}$. Again, details on this estimation process are provided in Chapter 3.

The final piece needed is the unit vector that points from the Sun to the CubeSat. The force will also be in this direction since the flux will always follow such a path and provide a pressure in that direction. That means that as the spacecraft is leaving eclipse and entering sunlight it will experience a resistive force similar to drag; as the spacecraft is leaving sunlight and entering eclipse, however, the pressure will be in the same direction as the velocity and will accelerate the CubeSat. Validating a simple model can be done by ensuring the semi-major axis oscillates over the course of an orbit due to this cyclical force. For this validation simulation the initial conditions from the WISE orbit are selected. The mass is set to 4 kg and solar pressure is defined as in Equation (22). For simplicity, the reflectivity coefficient is set to 1.4; this is generally considered a reasonable first order approximation [42]. Equation (23) summarizes the solar radiation pressure acceleration calculation; Figure 15 demonstrates the expected semi-major axis oscillations for the simple SRP model.

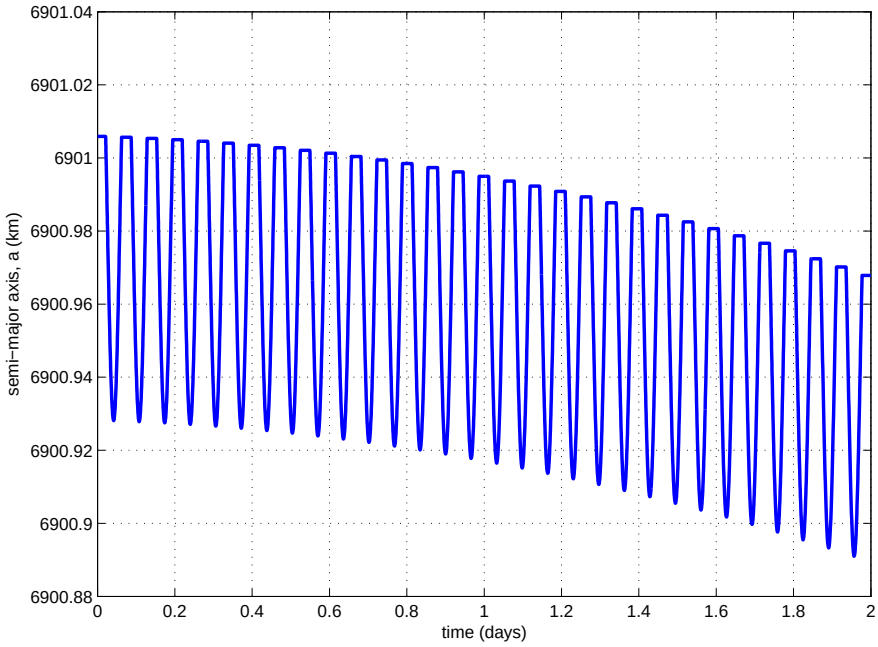

Figure 15: Check of solar radiation pressure model. An oscillating semi-major axis indicates the solar radiation force is appropriately decelerating or accelerating the spacecraft depending on whether it is leaving or entering eclipse.

$$\vec{a}_{SRP} = -\frac{p_{SR} c_R A_\odot}{m} \frac{\vec{r}_{sat\odot}}{r_\odot} \tag{23}$$

It is necessary to note that unlike the drag force which a spacecraft "feels" at all times, solar radiation pressure is only experienced when in sunlight. Therefore, in order to determine whether the force should or should not be considered it must first be determined if the CubeSat is in eclipse or not. Since the Mean of J2000 Earth Ecliptic reference frame was selected for all integrations and both reference orbits are initialized at midnight on January 1, the problem is slightly simplified. Although it cannot be assumed that the Earth-fixed reference frame is identically aligned with the J2000 frame every January 1, the error would be less than one degree and therefore sufficiently accurate given the remaining uncertainties in this calculation. To determine if the spacecraft is in eclipse first the spacecraft position is rotated about the z-axis equal to the angle that the Earth has carved out with respect to the Sun since January 1. Rotating the state aligns the x-axis so that it points directly at the Sun. With the vector rotated, if the spacecraft has a positive x-position, then it is in front of the Earth and therefore in sunlight. If the CubeSat does not have a positive

x-component, than the 2-*norm* of the y- and z-components of the position is compared to the Earth's reference radius; if the 2-*norm* is greater than the CubeSat is still in sunlight although behind the Earth; if instead the 2-*norm* is less than Earth's radius it is determined to be in eclipse.

To first verify that that method is accurately determining if the CubeSat is in eclipse, Figure 15 may be examined again. During the peaks in the oscillations, short periods absent of the oscillations are noticable. This is a result of the solar radiation perturbation becoming non-existent, as expected, when the CubeSat is in eclipse. Such flat regions do not exist at the troughs since a trough occurs as the CubeSat is passing over the equator. Before leaving this figure, there is reason to note the secular decrease in the semi-major axis; this decrease is a result of the solar radiation pressure pushing the CubeSat toward the Earth while it is near the equator. Since a near-equal but opposite force does not occur on the dark side of the Earth, the net effect is a total drop in semi-major axis length and an increase in eccentricity.

Although the method of determining whether the CubeSat is in sunlight or eclipse is simplistic, a quick verification with the automated output from *FreeFlyer* shows that it is 87.5% accurate. Some of this error is the result of the reduced fidelity in the force model. Recall that only the two-body point mass gravitational potential and solar radiation force are included. By increasing the model fidelity, this accuracy may be increased. Nonetheless, recalling the already existent error in the radiation area pressure coefficient ($\frac{p_{SR} c_R A_\odot}{m}$), the error in calculating sunlight or eclipse can also be partially captured through the estimation of that parameter.

By introducing solar radiation pressure, error improvements of about 1.5 km and 2 $^{m}/_{s}$ are realized for the position and velocity, respectively, over the J_2-only model. With both drag and solar radiation pressure the error improvements are approximately summed (adding drag further improves the SRP model in a similar manner as it did the J_2 model). Analysis plots are provided following the inclusion of third-body effects.

2.3.4 Third-Bodies

Disturbances from third-body gravitational potentials are the final perturbation to examine. Although any celestial body could be considered, only two bodies are selected: the Sun and Moon. Compared to other third-body perturbations these are orders of magnitude greater [66]. A summary

Table 5: Summary of third-body approximate gravitational parameters and distances from Earth used to selected which bodies to include in force model [66].

Body	**Min. Dist. from ♁** (km)	**Max. Dist. from ♁** (km)	μ (km^3/s^2)	**Max. Acc.** (km/s^2)
Sun	$147,098,290$	$152,908,232$	$132,712,428,000$	6.133×10^{-6}
Mercury	$77,281,390$	$221,915,132$	$22,032$	7.042×10^{-11}
Venus	$38,156,181$	$261,040,341$	$325,700$	4.9749×10^{-10}
Moon	$362,570$	$405,410$	$4,902.799$	3.069×10^{-6}
Mars	$54,570,768$	$401,307,532$	$43,050$	1.483×10^{-10}
Jupiter	$588,475,368$	$968,619,032$	$126,800,000$	3.673×10^{-10}
Saturn	$1,201,474,724$	$1,665,424,015$	$37,940,000$	2.655×10^{-11}
Uranus	$2,596,840,229$	$3,156,517,936$	$5,794,000$	9.182×10^{-13}
Neptune	$4,300,842,601$	$4,706,044,722$	$6,809,000$	3.896×10^{-13}
Pluto	$4,284,901,768$	$7,463,098,232$	900	2.175×10^{-14}
Eris	$5,497,901,768$	$14,752,098,232$	$1,108$	1.322×10^{-14}

of estimate perturbation accelerations is provided in Table 5. Examine the exponents for the *Max. Acc.* column to see why only the Sun and Moon gravitational pertubations need to be included; each of these accelerations are at least 4 orders of magnitude greater than any acceleration due to another celestial body.

In order to track the position of the Sun and Moon several options are considered. The first obvious option is to further expand the state vector to include six states each for the Sun's and Moon's position and velocity. This would additionally require slightly modifying our force model to ensure that the derivatives are calculated. Further effort would be required to select an appropriate force model and reference frame for each celestial body. The computational cost is estimated to be too large for consideration of this method; memory, more than flops, is of concern since *each* time step would require an additional 96 bytes which quickly adds up. Therefore, the idea of expanding the state vector is shelved to pursue other methods.

Use of analytical expressions for determining the position of the Sun and Moon is also considered. By fitting a set of sine and cosine waves to each body's Cartesian positions with respect to the Mean of J2000 Earth Ecliptic coordinate frame, a position could be determined by simply provide

the time with respect to some defined epoch. Minimal exploration is completed with fitting sine curves as it is quickly determined that a trade study comparing analytical expression complexity with the time frame length for which the expression is valid would be necessary. Before conducting such a trade study, a third and final family of method is considered.

For the third method the same JPL HORIZONS database mentioned previously is utilized to acquire Sun and Moon state vectors. A single state vector for each day over the course of the year 2011 is selected as the primary dataset. With this dataset a few options are available in terms of finding each body's position at a given time. The simplest method would be to select the state vector whose associated time is closest to the desired time; in this method there is no interpolation, just a simple time comparison to find the minimum difference. Although this could work relatively well for the Sun's position, this does not prove to be sufficient for the Moon's. A quick calculation shows that over the course of one day the Moon covers an arc of almost a quarter radian; since multiple data points would be available to our on-board integration software a better estimate can be made. The next simplest method is linear interpolation. Finding the two closest data points, calculating a slope and determining the corresponding value given a time improves the state estimate quite well with very little overhead.

Beyond linear interpolation more complicated models or higher-order curve fits could be utilized. Since earlier exploration determined fitting of a sinusoid would require in depth trade studies and may not result in improved performance, only one slightly more complicated interpolation model is selected: two-body integration. Like the first two methods, the data point closest to actual time is first selected. Using this as an initial condition, a simple two-body integration is conducted; depending on whether the data point is in the past or future with respect to the "current" time determined whether the propagation is conducted forward or backward. Since the Sun and Moon have drastically different paths with respect to the Earth, the methods are tested on both bodies. Figures 16 and 17 compare the error between linear interpolation and the use of two-body integration for the Moon and Sun, respectively. The sinusoidal behavior is expected since it assumes that a perfect knowledge data point is available once a day. Additionally, the sharp peaks seen on the two-body error plot are expected since the integration is conducted forward and backward from the closest "true" data point and meet in the middle. Specifically note the different ordinate scales; a

quick comparison shows that error is reduced by about two orders of magnitude. This does not come without some cost, however; on average the linear interpolation only requires 45 μs while the two-body short integration requires 3 ms. With these average compute times, one can see that the two-body integration takes about 67 times longer to computer. Such an increase in computational cost, however, can be accepted when it is compared with the 100 fold decrease in average error. With this approximately even trade, the two-body interpolation is selected as the primary method of calculating third-body positions.

Before providing the final error analysis, a comparison between the immediately preceding model and the new model now with these two third-body effects. After approximately ten orbits, introducing Sun and Moon gravitational perturbations resulted in an approximate additional 700 m and 70 $^{cm}/_{s}$ accuracy in position and velocity respectively.

To summarize the incremental improvement to the model, Figure 18 provides the 2-*norms* of the position and velocity compared to the *FreeFlyer* reference orbit; Figure 19 provides an equivalent plot for the WISE reference orbit. Note that although the final model does not provide the best results when compared to the WISE orbit, this is caused by estimating the physical parameters of the spacecraft. Since these parameter estimates will improve through on-board orbit determination, it is preferable to leave the model with these force terms.

2.3.5 Final Force Model

The final force model, as described above, includes accelerations due to the following:

- Two-body gravitational potential
- Non-spherical gravitational potential due to Earth oblateness, J_2
- Drag
- Solar Radiation Pressure
- Third-body gravitational potential, specifically from Sun and Moon

Equations for the calculation of each of these acceleration terms have been provided throughout this chapter, but for completeness the final set of derivative model equations are provided here in

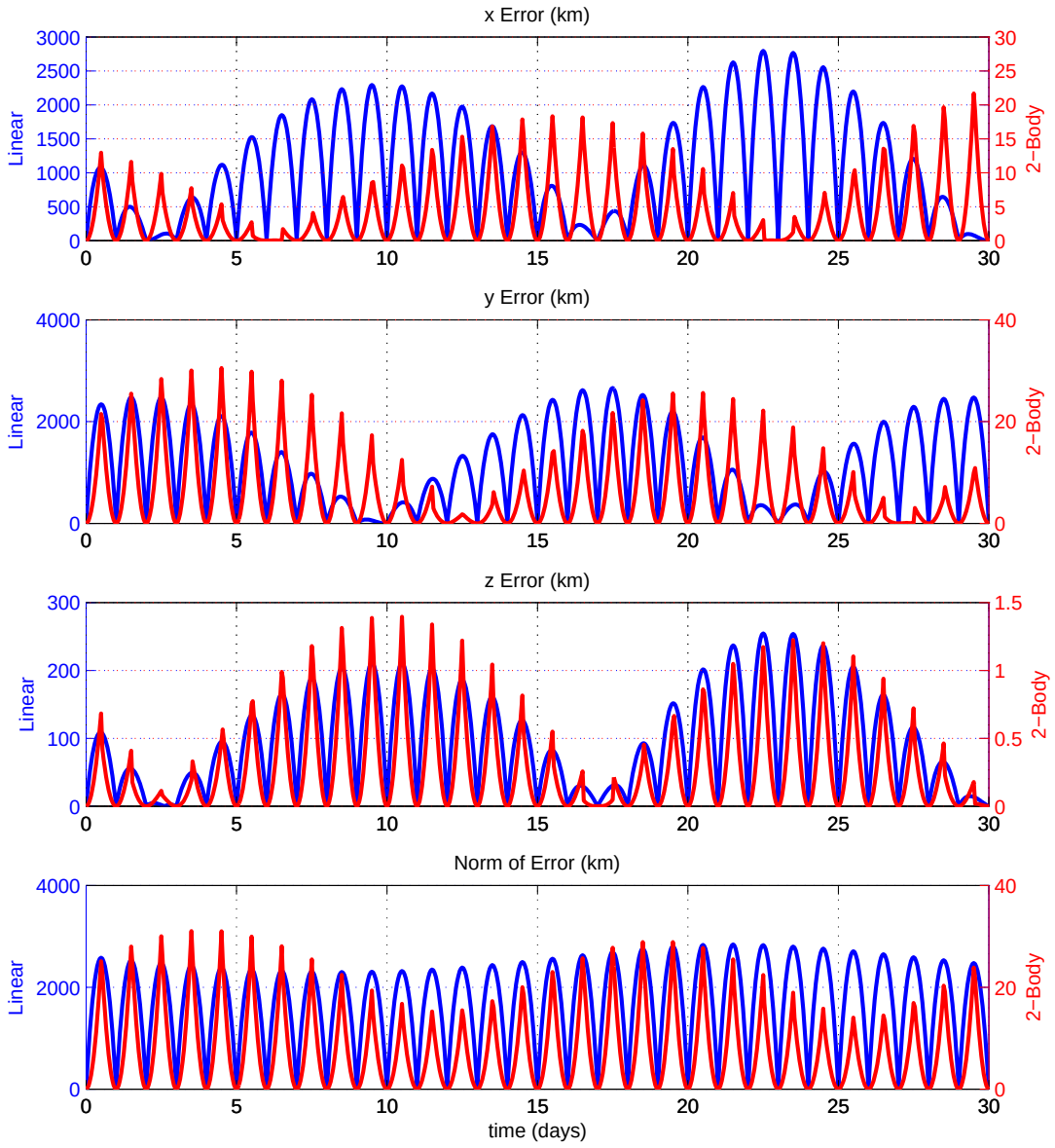

Figure 16: Comparison of the position error over time when using linear interpolation versus two-body forward/backward integration for determining the position of the Moon.

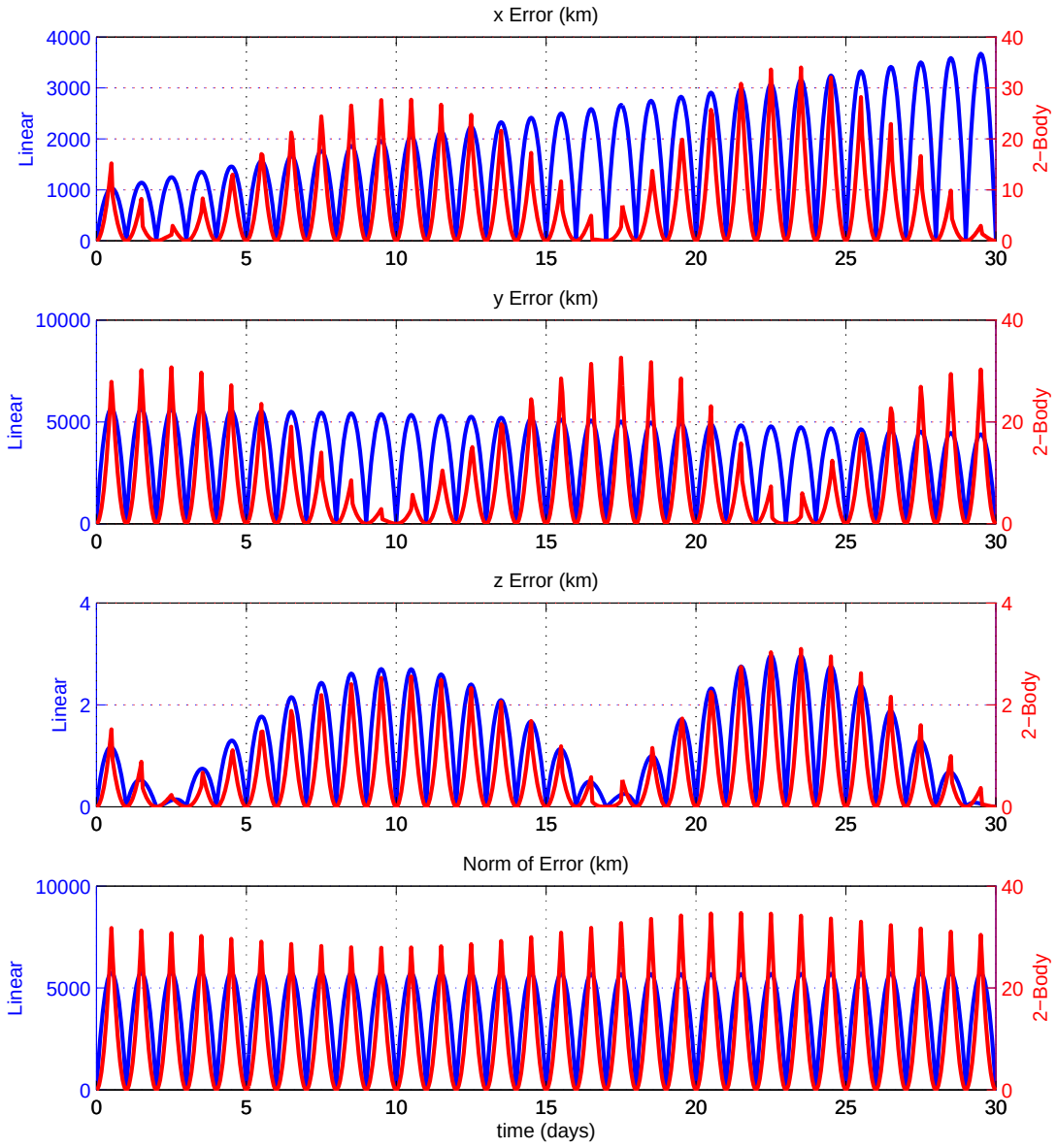

Figure 17: Comparison of the position error over time when using linear interpolation versus two-body forward/backward integration for determining the position of the Sun.

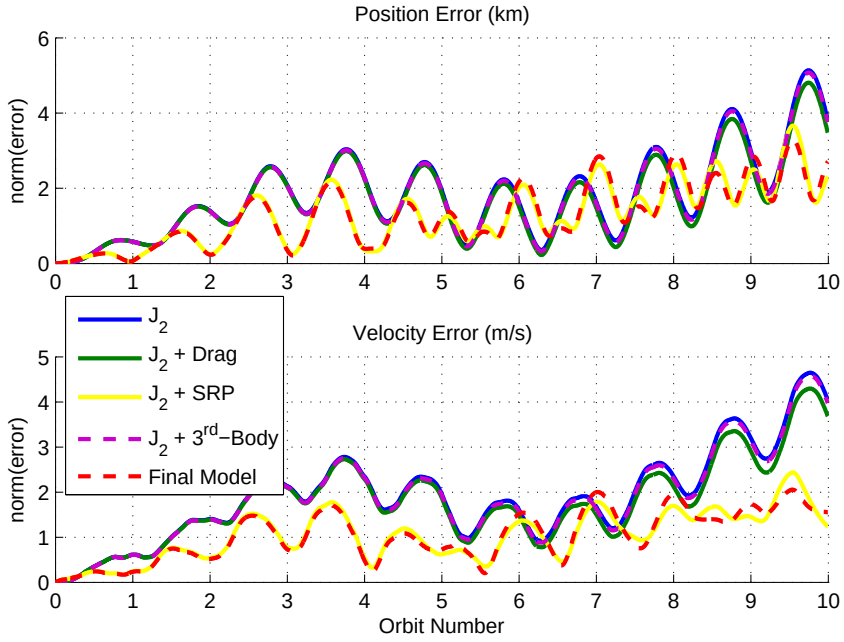

Figure 18: Incremental improvement in force model accuracy compared to the *FreeFlyer* reference orbit is achieved through the addition of J_2, drag, SRP, and third-bodies.

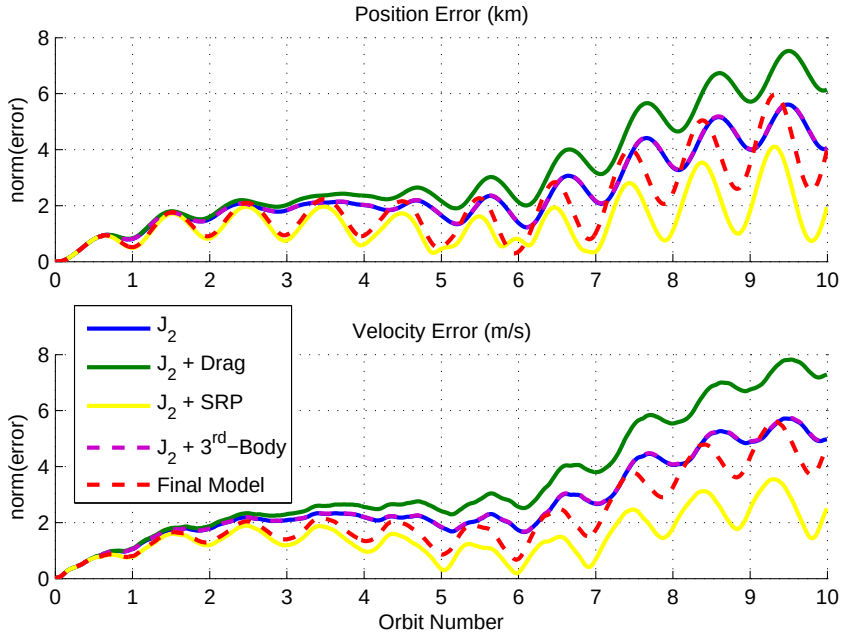

Figure 19: Incremental improvement in force model accuracy compared to the WISE reference orbit is achieved through the addition of J_2, drag, SRP, and third-bodies.

Equations (24), (25), and (26). These equations provide the accelerations in the Mean of J2000 Earth Ecliptic inertial frame. Note the fourth row in the $\ddot{x}$ and $\ddot{y}$ equations which provide the SRP acceleration. Given the inertial reference frame selection SRP is only considered to perturb the spacecraft in the *xy*-plane; the direction of perturbation is determined by the time of year, which is the reason for including the cosine and sine with the $\Omega_{\oplus wrt\odot}$ term (the revolution rate of the Earth with respect to the Sun) and time. Since the Earth is rotating around the Sun but the *x*-axis is inertially fixed, the vector pointing from the Sun to the spacecraft rotates as the year progresses. Furthermore, note that the equations do not include the check to determine if the CubeSat is in eclipse of sunlight; if the satellite is in eclipse then the SRP acceleration term is ignored in the derivative call.

$$\begin{aligned}
\ddot{\vec{x}} &= -\frac{\mu_{\oplus}}{\vec{r}}x \\
&+ \frac{3J_2\mu_{\oplus}R_{\oplus}^2\left(\frac{3}{2}\frac{z^2}{r^2}-\frac{1}{2}\right)}{r^5}x + \frac{3J_2\mu_{\oplus}R_{\oplus}^2z^2}{r^7}x \\
&- \frac{c_D\rho V_{rel}^2 A}{2m}\frac{u_{rel}}{v_{rel}} \\
&+ \frac{p_{SR}c_R A}{m}\cos\left(\Omega_{\oplus wrt\odot}\,t\right) \\
&+ \frac{\mu_{\odot}}{\|-\vec{r}+\vec{r}_{\oplus\odot}\|^3}\left(-x+x_{\oplus\odot}\right) - \frac{x_{\oplus\odot}}{\|\vec{r}_{\oplus\odot}\|^3} \\
&+ \frac{\mu_{\leftmoon}}{\|-\vec{r}+\vec{r}_{\oplus\leftmoon}\|^3}\left(-x+x_{\oplus\leftmoon}\right) - \frac{x_{\oplus\leftmoon}}{\|\vec{r}_{\oplus\leftmoon}\|^3}
\end{aligned} \tag{24}$$

$$\begin{aligned}
\ddot{\vec{y}} &= -\frac{\mu_{\oplus}}{\vec{r}}y \\
&+ \frac{3J_2\mu_{\oplus}R_{\oplus}^2\left(\frac{3}{2}\frac{z^2}{r^2}-\frac{1}{2}\right)}{r^5}y + \frac{3J_2\mu_{\oplus}R_{\oplus}^2z^2}{r^7}y \\
&- \frac{c_D\rho V_{rel}^2 A}{2m}\frac{v_{rel}}{V_{rel}} \\
&+ \frac{p_{SR}c_R A}{m}\sin\left(\Omega_{\oplus wrt\odot}\,t\right) \\
&+ \frac{\mu_{\odot}}{\|-\vec{r}+\vec{r}_{\oplus\odot}\|^3}\left(-y+y_{\oplus\odot}\right) - \frac{y_{\oplus\odot}}{\|\vec{r}_{\oplus\odot}\|^3} \\
&+ \frac{\mu_{\leftmoon}}{\|-\vec{r}+\vec{r}_{\oplus\leftmoon}\|^3}\left(-y+y_{\oplus\leftmoon}\right) - \frac{y_{\oplus\leftmoon}}{\|\vec{r}_{\oplus\leftmoon}\|^3}
\end{aligned} \tag{25}$$

$$
\begin{aligned}
\ddot{\vec{z}} &= -\frac{\mu_{\oplus}}{\vec{r}} z \\
&+ \frac{3J_2\mu_{\oplus}R_{\oplus}^2\left(\frac{3}{2}\frac{z^2}{r^2}-\frac{1}{2}\right)}{r^5} z - \frac{J_2\mu_{\oplus}R_{\oplus}^2\left(\frac{3z}{r^2}-\frac{3z^3}{r^4}\right)}{r^3} \\
&- \frac{c_D\rho V^2 A}{2m}\frac{w_{rel}}{V_{rel}} \\
&+ \frac{\mu_{\odot}}{\|-\vec{r}+\vec{r}_{\oplus\odot}\|^3}\left(-z+z_{\oplus\odot}\right) - \frac{z_{\oplus\odot}}{\|\vec{r}_{\oplus\odot}\|^3} \\
&+ \frac{\mu_{\leftmoon}}{\|-\vec{r}+\vec{r}_{\oplus\leftmoon}\|^3}\left(-z+z_{\oplus\leftmoon}\right) - \frac{z_{\oplus\leftmoon}}{\|\vec{r}_{\oplus\leftmoon}\|^3}
\end{aligned} \tag{26}
$$

2.4 Integration Routines

Selection of an integration routine, specifically the appropriate order, is necessary in order to ensure a balance between computational cost and accuracy. The popular Runge-Kutta routine is thoroughly examined with orders ranging from fourth through eighth. Integrator size, meaning code file size, is also taken into consideration during the decision process since the final implementation will be placed on a micro-controller in future work. The first step, however, is determining whether fixed or variable step methods should be considered.

2.4.1 Fixed Step vs. Variable Step

Before conducting simple simulations, it is hypothesized for the force models being examined that a fixed-step integrator will suffice. First, the orbits being examined are circular or near-circular, therefore we would expect the force experienced by the satellite is near equal from one time step to the next. For this research, that is further enforced by the lack of on-board propulsion; even with the addition of nano arc thrusters (the most likely thrust source for a CubeSat) the additional net force would not require variable-step integrators. Another advantage of using fixed-step is the ability to accurately estimate the computation time and memory footprint for a given simulation time. For a given force model and integrator method the exact number of flops can be calculated and then multiplied by the number of steps required. Having this knowledge during the CubeSat design phase will allow for reduced variance in estimating the required size of on-board storage and microprocessor clock speed.

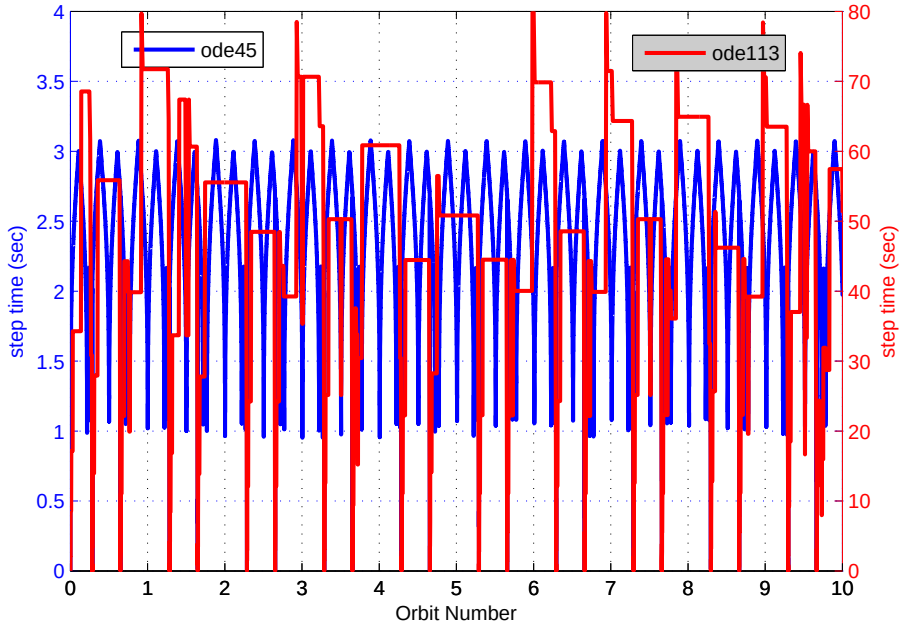

Figure 20: Using *MatLab*'s built-in variable time step integrators and our final force model results in cyclical behavior in the selected time step size.

To confirm this hypothesis the *FreeFlyer* reference orbit aforementioned is propagated using variable-step integrators built into *MatLab*; specifically `ode45`, a medium-order ordinary differential equation (ODE) solver, and `ode113`, a high/variable-order ODE solver, are examined. Using each method ten orbits are propagated using the final force model described in section 2.3.5 and error tolerances of 1×10^{-12}.

In Figures 20 and 21 the time step value for each integration step is plotted. As expected, both methods start with small values and increase over the first orbit period. `ode45` never reaches a steady-state, but does oscillate around a consistent mean value. The abscissa scale is orbit number in order to determine if there is a correlation between the oscillations and the orbital period. An almost identical plot is seen for a J_2-only potential force model. `ode113` also does not reach a steady-state time step. However, when run with a J_2-only model `ode113` does settle at a steady-state time step of about 50 seconds; see Figure 22. This indicates that one of the cyclical forces such as the third-bodies or solar radiation pressure are causing the integrator to greatly reduce the time step. Since there are two distinct sudden decreases in the time step size per orbit, it is hypothesized that these aligned with when the spacecraft is entering or leaving the eclipse. To verify this assumption, the

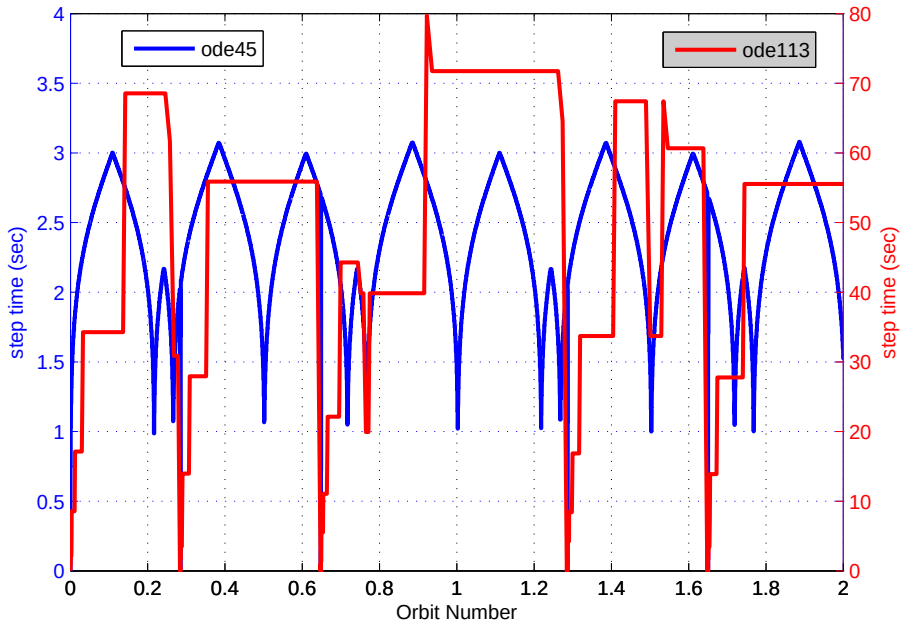

Figure 21: Zooming in along the abscissa from Figure 20 clearly displays the cyclical time step behavior with respect to the orbit.

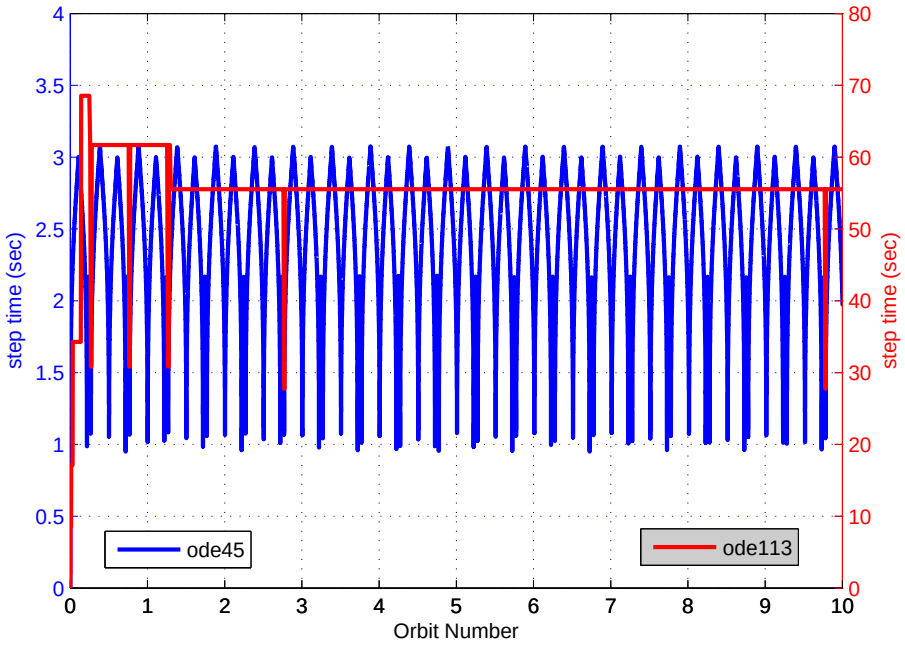

Figure 22: A simple force model (J_2-only) allows for the establishment of an *ode113* steady-state time step.

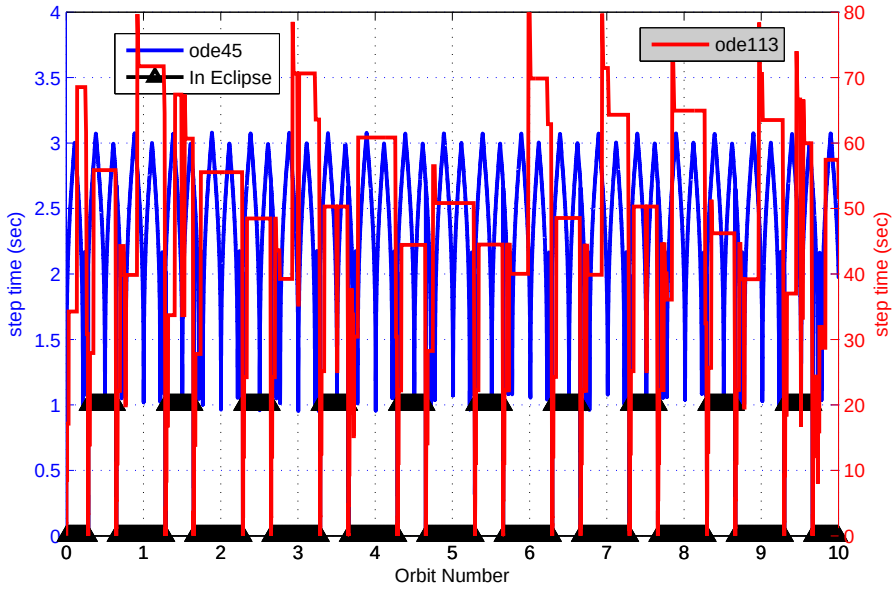

Figure 23: When a binary value for whether the spacecraft is in sunlight (0) or eclipse (1) is added to Figure 20, the spikes in *ode113*'s time step size clearly align with the spacecraft entering or leaving eclipse.

binary "in eclipse" value is also plotted, as seen in Figure 23. This binary value simply registers as 0 if the spacecraft is not in eclipse and 1 if it is in eclipse. Clearly, at the barrier between entering or leaving eclipse, the variable time step integrator has to reduce the size to near-zero in order to meet the desired error tolerance of 1×10^{-12}. A second check is to remove only the solar radiation pressure from the force model; doing so results in a plot identical to Figure 22. Since the solar radiation pressure acceleration is on the order of 5×10^{-8} $\mathrm{km/s^2}$, selecting a fixed time step may still be reasonable; section 2.4.2 goes into detail on the error calculated for each different Runge-Kutta integrator to determine whether a fixed-step integrator is sufficient.

2.4.2 Runge-Kutta

Due to Runge-Kutta methods being ubiquitous integrators, this family of methods is the primary set examined. Equations (27, 28, 29, 30) summarize Runge-Kutta methods in general [31]. For the system at hand a simple first order ODE is used as our derivative function, as seen in Equation (27); the right-hand side of this equation is determined by the selected force model. Equation (29) compactly shows how intermediate states are calculated, with Equation (28) being the initial

condition. Finally, the new state at time $t_0 + h$ (where h is the time step), is the weighted average determined by Equation (30). Note that the error term, $O(h^{order+1})$, is not shown in Equation (30) since these implementations are not estimating the error nor modifying the time step with such an estimate. Depending on the selected method order, the α, β, and c coefficients are different. Calculating coefficients for new methods is outside the scope of this effort, but details are available in [31] as to how the coefficients used in this research are obtained.

$$x' = f(t, x) \tag{27}$$

$$f_0 = f(t_0, x_0) \tag{28}$$

$$f_\kappa = f\left(t_0 + \alpha_\kappa h, x_0 + h \sum_{\lambda=0}^{\kappa-1} \beta_{\kappa\lambda} f_\lambda\right) \tag{29}$$

$$y = y_0 + h \sum_{\kappa=0}^{N} c_\kappa f_\kappa \tag{30}$$

Coefficients for fourth through eighth order are obtained and implemented [31]. For each method seventh-order of lower a *Butcher Tableau* is provided with the various coefficients utilized; due to the size of the eighth-order it is provided in a more basic tabulated format. Using various time steps, h, each method's accuracy using the section 2.3.5's force model is examined over the course of ten orbits. For document flow the tables of coefficients are located in Appendix B; immediately following is a summary of the error analysis.

2.4.2.1 Error Analysis and Method Selection

The fourth-order Runge-Kutta integrator is likely the most popular due to its simplicity yet relatively good accuracy. The popularity of this method also stems from the fact it can be easily modified into a predictor-corrector variable-step method; the simplest way is to halve the step size and compare two half steps with one full step until an error tolerance is met. Intelligently calculating the derivatives can provide this predictor-corrector portion with little extra computational expense [55]. The higher-order methods include equations for estimating the error for purposes of variable-step integration. As stated earlier, selection of a fixed-step method is preferred for flop estimation; furthermore, correlating times between GPS data and the current model within the EKF is simplified by integrating at a fixed rate that is some integer factor of the GPS receiver sample rate.

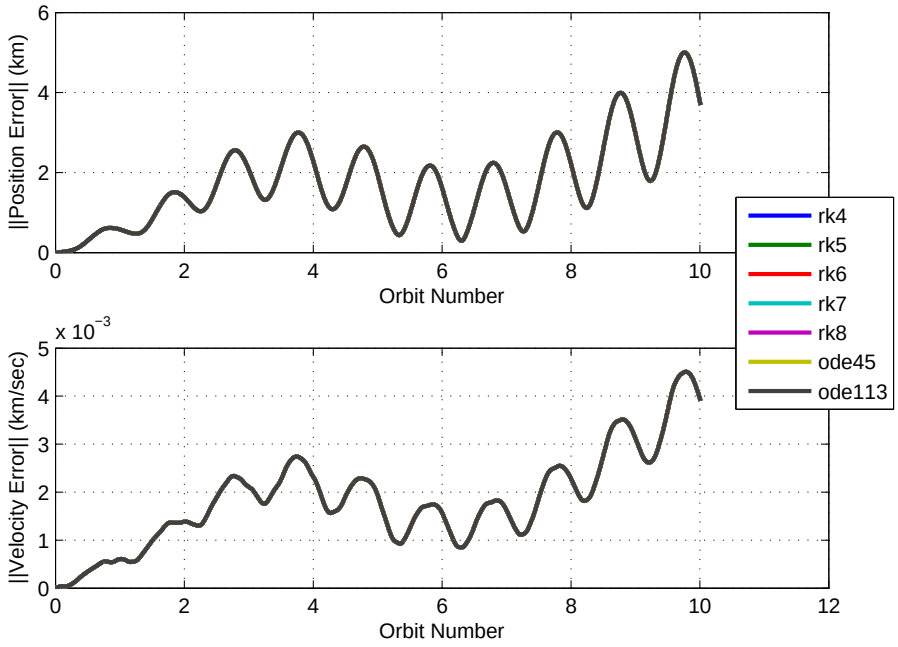

Figure 24: Norm of position and velocity error for Runge-Kutta fixed-step of 15 seconds and built-in *MatLab* integrators.

For comparison time steps of 15, 30, 45, 60, 90, and 120 seconds are used with the fixed-step Runge-Kutta methods and compared to *MatLab*'s `ode45` and `ode113` integrators. Figures 24, 25, 26, 27, 28, ans 29 provide the results for the respective time steps. The error is defined as the difference from the *FreeFlyer* reference orbit. As the time step increases, the lower order RK4 and eventually RK5 methods diverge from the rest of the group. It is interesting that for this reference orbit example the fourth-order routine actually shows reduced error until using the largest time step of 120 seconds; with the 120 second time step, the fifth-order routine depicts the expected behavior of diverging from the group with increased error. In all of these figures the methods for which curves are not clearly depicted are shadowed by the `ode113` curves. Examining the individual curves closely shows differences on the order of meters at the worst case and often the centimeter order of magnitude. Differences of these magnitudes are deemed insignificant considering the absolute error after 10 orbits is approximately 4 km. From comparing the errors alone, the obvious selection would be a high-order method with a small fixed-step or variable-step; consideration of execution time, however, is necessary for a balanced solution.

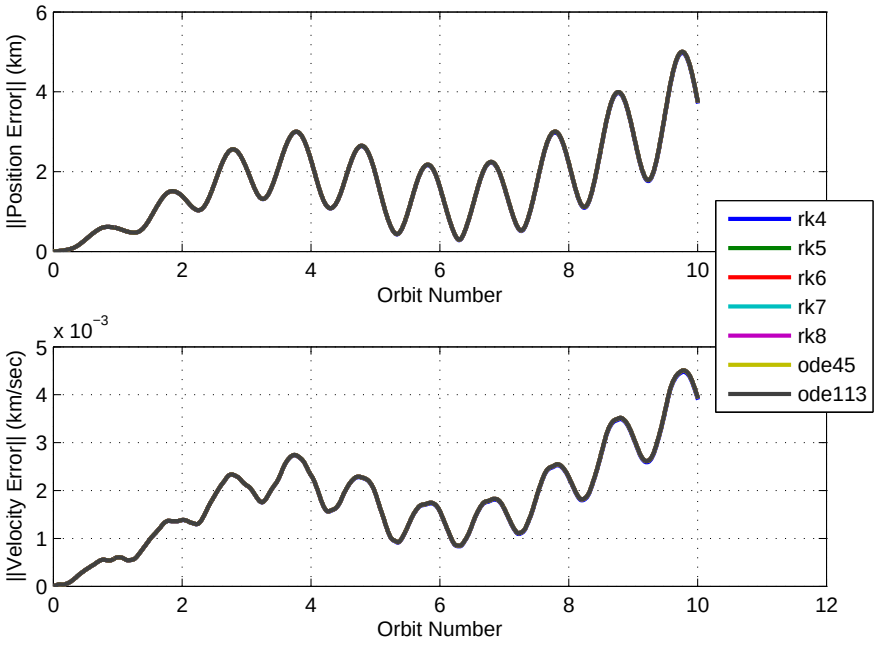

Figure 25: Norm of position and velocity error for Runge-Kutta fixed-step of 30 seconds and built-in *MatLab* integrators.

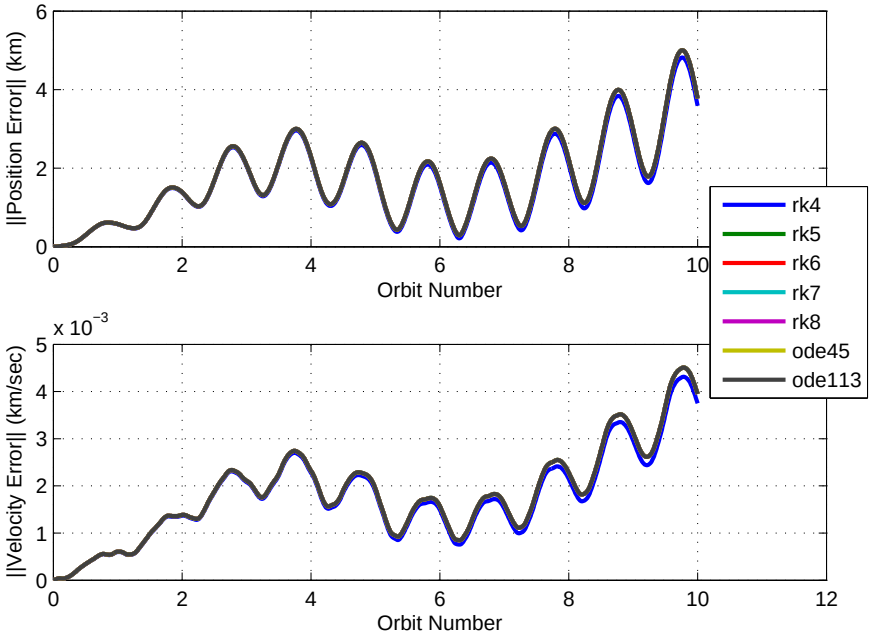

Figure 26: Norm of position and velocity error for Runge-Kutta fixed-step of 45 seconds and built-in *MatLab* integrators.

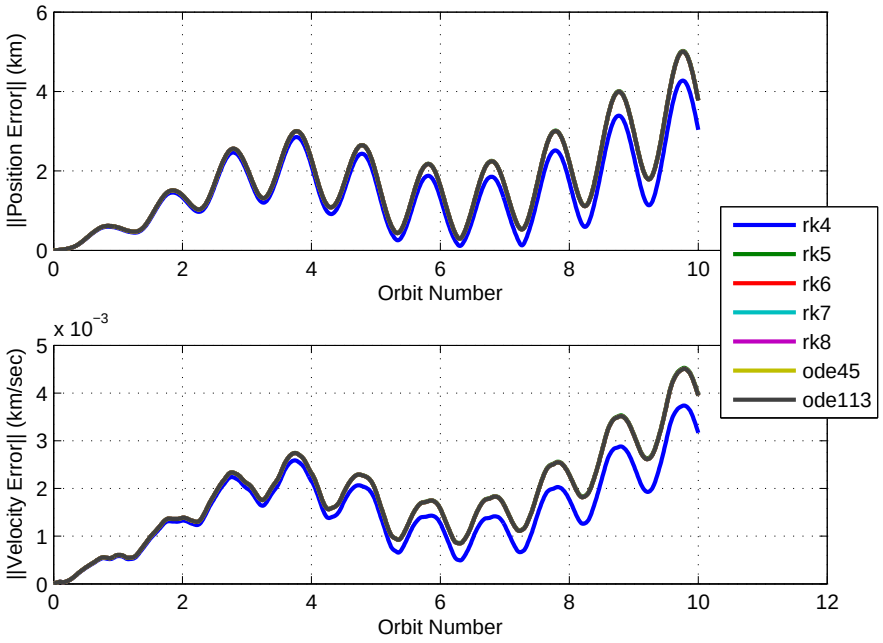

Figure 27: Norm of position and velocity error for Runge-Kutta fixed-step of 60 seconds and built-in *MatLab* integrators.

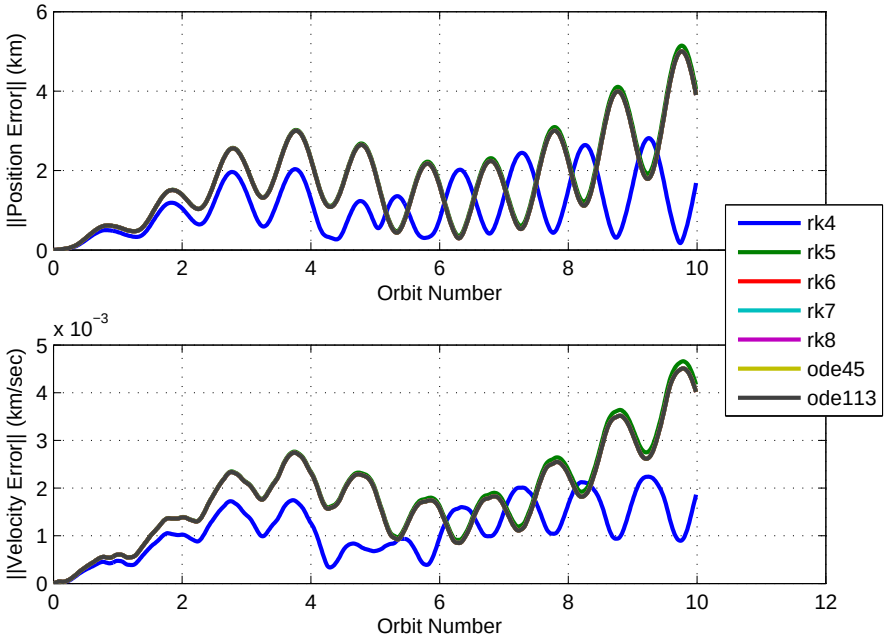

Figure 28: Norm of position and velocity error for Runge-Kutta fixed-step of 90 seconds and built-in *MatLab* integrators.

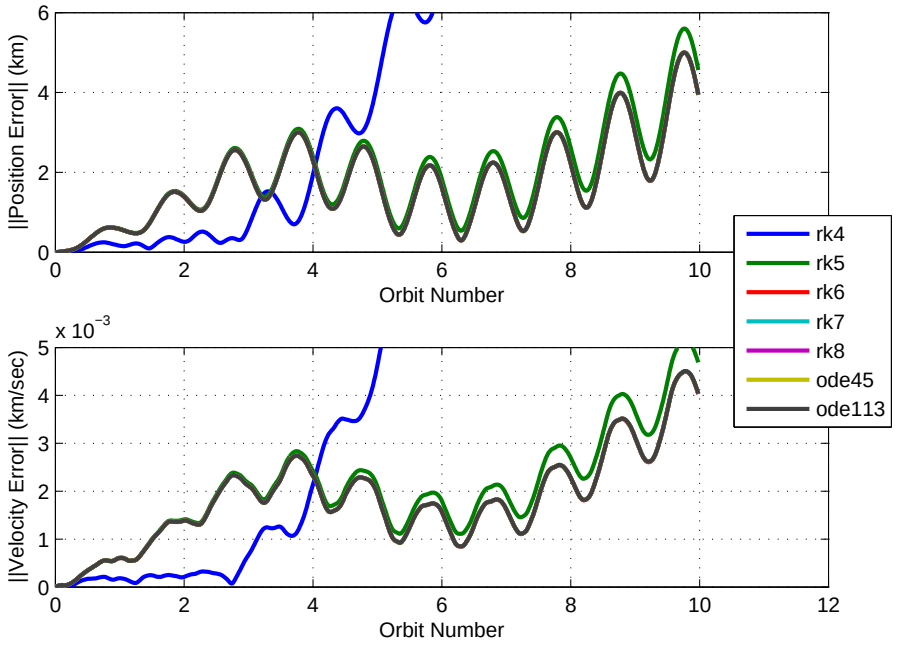

Figure 29: Norm of position and velocity error for Runge-Kutta fixed-step of 120 seconds and built-in *MatLab* integrators.

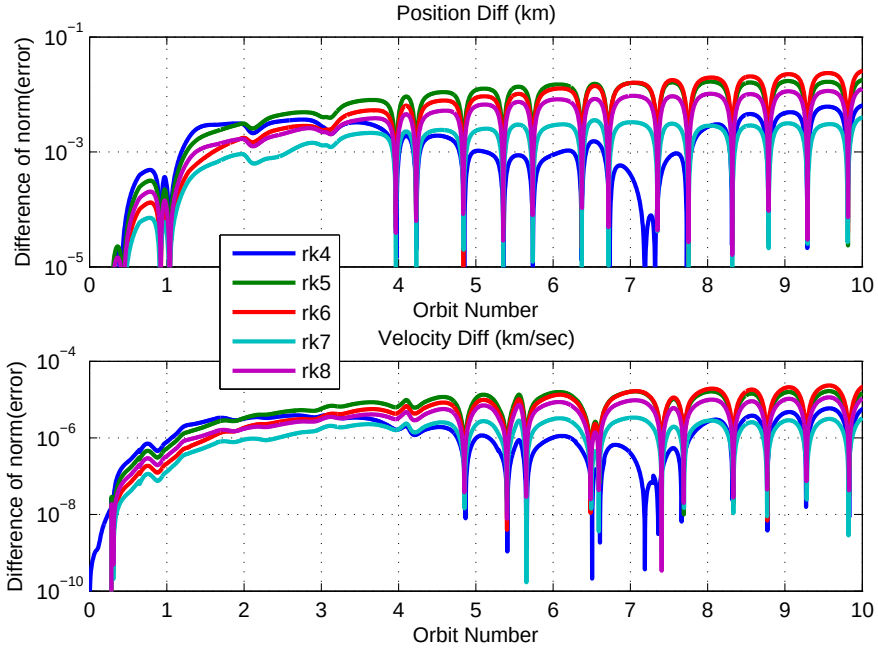

Figure 30: Difference between fixed-step Runge-Kutta methods and *MatLab*'s *ode113*.

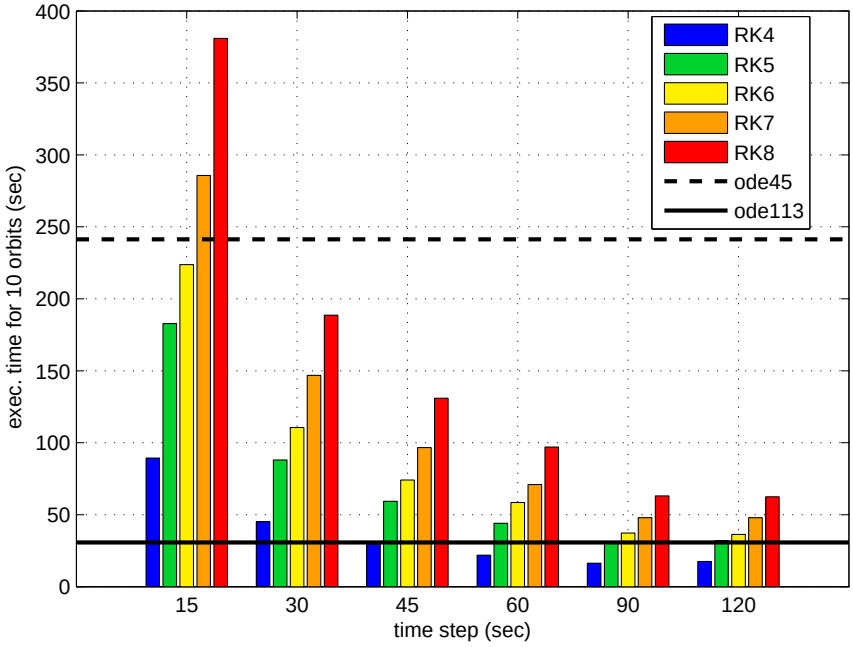

Figure 31: Increasing time steps reduces total execution time as does decreasing the complexity of the Runge-Kutta method. The inverse correlation is true of accuracy, therefore a compromise is required.

Figure 31 provides the execution times to integrate 10 orbits using the *FreeFlyer* reference initial conditions. The two anticipated trends are identified: first, as the method's order increases the execution time increases; second, as the time step increases the execution time decreases. Recalling from immediately preceding figures that accuracy increases with decreasing time step, it is clear reason that a compromise is necessary of method order, time step, and accuracy is necessary.

From this analysis four different integration routines paired with time steps stand-out as possible solutions. First, since RK4 requires the least amount of time for time steps of 45 seconds or greater while providing the least error (although this is seems to be an anomaly), RK4 paired with both a 45 second and 60 second time step are explored. Examining the plots of error closely shows that RK6 provides slightly better accuracy than the other fixed-step and variable-step methods; combining RK6 with a 60 second time step will be tested in the final software implementation. Finally, for the simulations run in the *MatLab* environment, `ode113` with a relative and absolute accuracy threshold of 1×10^{-12} will be examined as a means to determine whether exploration of higher-order variable-step finely-tuned methods such as those provided in *MatLab* should be considered for future module

improvements.

CHAPTER III

ON-BOARD ORBIT DETERMINATION

On-board orbit determination is the second primary piece of software developed for this research. To enable a spacecraft formation to function autonomously, it needs on-board sensors and software in order to determine and maintain knowledge of its position. This thesis takes that argument one step further: maintaining knowledge of other spacecraft in the formation is, if not necessary for autonomous behavior of a single spacecraft, provides a means to efficiently and effectively plan mission tasks such as data collection or formation maintenance and reconfiguration. While this chapter discusses the algorithm and implementation for on-board orbit and attitude determination, the algorithm designed for state knowledge sharing is provided in the Chapter 5.

3.1 On-Board Sensor Selection

Collection of position and possibly velocity data is necessary in order to conduct on-board orbit determination. Range and range rate measurements between known ground stations and a receiver could have been utilized, however there are some disadvantages to this approach. First, it requires the use of ground stations, which may or may not be manned and must be capable of receiving and sending on a specific frequency; although autonomous activity could still been conducted the system of spacecraft is clearly no longer self-sufficient. Generally, range measurements taken from a ground station simply use a radar that requires no interaction from the object that the radiation is bouncing back off of. On a picosatellite it is unreasonable to assume sufficient power is available for this, therefore the ground station needs to play an active role in receiving the signal, processing the data, and sending a response. Note that this means knowledge of the time required to process the signal is also required for each ground station involved.

A second disadvantage is the need for accurate data on the location of each ground station. A final issue, and probably most disconcerting if trying to provide precise velocity estimation, is knowledge of the carrier signal frequency accuracy and any effects that may cause a shift in the frequency are required [70]. If we are using amateur radio ground stations, for example, as locations

to "ping", the accuracy of the return carrier frequency is relatively loose even before we take shift due to Doppler and the ionosphere into consideration. By measuring the Doppler shift of a known frequency, precise velocity determination is possible; if the originating frequency does not guarantee sufficient accuracy on the order of tens of Hertz, then the Doppler shift calculation may not be useful for velocity determination. It is through the use of atomic clocks the GPS satellites are able to provide accurate time to 14 nanoseconds and a transmitted carrier within Hertz of the selected frequency [2].

Although not specifically an on-board implementation, the most common mechanism for orbit determination is to collect data from various ground tracking stations, conduct analysis, and then uplink the latest orbit solution to the spacecraft. This is similar to the approach just described, except opposite in the sense that ground stations are collecting range and range rate data for a spacecraft passing overhead. Either one ground station collecting data over multiple passes or multiple ground stations collecting data, an accurate orbit solution can be calculated. If a specific mission has a global network of ground stations, this approach could be considered since all power and computation necessary, except for receiving the small amount of data from a ground station, is completed on the ground; when compared to a small spacecraft, a ground station's resources are nearly unlimited. However, for this application the goal is to enable autonomous operation of a CubeSat formation; the tracking by ground stations could also be conducted autonomously, but the resources consumed for such an implementation are uncommon in CubeSat operations.

Setting range and range rate data collection aside, the obvious solution is an on-board GPS receiver designed for the LEO environment. Use of GPS receivers on LEO spacecraft has been of interested for nearly a decade and there are now multiple COTS receivers designed for the nano- and picosatellite platform [50, 51, 14]. A survey of the market was conducted in order to determine the expected accuracy of GPS receivers. Position accuracy is consistent across the field of receivers falling within 10 - 20 m. Some receivers specifically stated the accuracy to be a 2σ value; a worst-case 3σ value was assumed when precision was not supplied. Depending on the receiver, velocity measurements are also available. Unlike the position accuracy, velocity accuracy varied from 3 $^{cm}/_{s}$ to 1.5 $^{m}/_{s}$. In order that the module be agnostic to the GPS receiver selected, the orbit determination software is designed to work with solely position measurements while still being able to take

advantage of any velocity measurements for estimation improvement. GPS receivers also have the advantage that their provided position solutions have error representative of Guassian noise. Knowing that the selected sensor error can be assumed Gaussian is required in order to use the extended Kalman filter discussed in the following section.

Considering some of the commercially available GPS receivers have excellent accuracy specifications for both position and velocity, it could be argued that the sensor data could be used directly for comparing the predicted state via propagation versus a spacecraft's actual state. However, the collaborative module design uses an extended Kalman filter for processing GPS measurements for several reasons. First, from the survey it is evident that while some receivers offer an option of filtering and smoothing the data, this is not true of all receivers, especially those that have no or poor accuracy velocity measurements. Second, use of an extended Kalman filter provides a means to update a spacecraft model's gravitational parameters for the three celestial bodies, J_2, and the spacecrafts drag and SRP coefficients. By updating the model and sharing these parameters with spacecraft team members, the frequency of communication between teammates can decrease as the model provided at launch is updated. Although it is a steady-state frequency of communication is reached, by constantly updating the model and not just the state, this steady-state value is reduced achieving one of the goals of this research. Details from the simulations proving this decrease in inter-satellite communications are provided in Chapter 5. An overview of Kalman filters plus an explanation of the specific implementation for the collaborative module software follows.

3.2 Kalman Filter

Originally described by Peter Swerling but refined by Rudolph Kalman in 1960, the Kalman filter (KF, a.k.a sequential estimation algorithm) has become extensively used in the processing of data from known noisy sensors [63]. At its root, a Kalman filter provides a means to optimally combine sensor data with known variance and predications based on propagation via a dynamic model to develop an estimate which is better than the senor data or model alone. Although this implementation assumes a single sensor, the Kalman filter algorithm has no limit on the number of sensor measures that can be taken into consideration and can improve an estimate if more or different measurements are available. Furthermore, it is not necessary for sensors to measure the desired state directly; a

mapping between the measurements and state, often referred to as an observation model, is utilized to expand the realm of useful observations. For example, the orbit determination problem seeks a 6D state vector of either Cartesian coordinates or orbital elements; the state itself does not need to be measured but instead range and range rate measurements from ground stations with known locations can be utilized alongside a model propagation in Cartesian space as long as the mapping from the measurements to the state is provided [63].

There are multiple flavors of the Kalman filter including the standard linear filter, the extended, and the unscented. Per the name just given, the standard or linear Kalman filter requires the use of a linear prediction model and linear relationship between the sensor measurements and desired state. Many times a model can be linearized and still represent reality sufficiently, but when a non-linear model is deemed necessary the extended Kalman filter (EKF) provides a solution. In an EKF the state prediction and measurement to state mapping functions may be non-linear functions. In order to calculate the covariance matrix estimate at each step, however, the Jacobian (a matrix of partial derivatives) of the state and state derivative must be calculated. Depending on the non-linear function, the partial derivatives may be analytical expressions, but if not the Jacobian can be numerically estimated. By using the Jacobian for the covariance estimation, the extended Kalman filter is essentially linearizing the functions in the neighborhood of interest [63]. When a dynamic or observation model is highly non-linear, this linearization can result in the EKF providing poor estimates. The unscented Kalman filter uses the unscented transform method to select a minimal set of points around the model mean to be propagated via the dynamic model. From the individual point propagations the mean (new state estimate) and covariance are estimated [35]. Recalling Chapter 2's discussion of the force model, the problem at hand is not linear, but does not seem to present itself as highly non-linear (the most non-linear portion being the solar radiation pressure due to entering and leaving eclipse). For this effort, an unscented Kalman filter is not selected for exploration, but future work could compare this EKF implementation with a UKF approach for verification of the near-linear assumption. Therefore, an extended Kalman filter using the non-linear force model and a unity mapping between the GPS measurements and the spacecraft state has been implemented.

3.2.1 Extended Kalman Filter

To begin discussion of the EKF implementation Equations (31)-(42) encompassed by summarized steps are provided [63].

1. Initialize state vector of size n. Initialize state transition matrix (STM) to Identity matrix of size n x n. This implementation uses a state vector of size $n = 12$.

$$\begin{aligned} \vec{x} &= \vec{x}_0 \\ \Phi &= I_{n \text{ x } n} \end{aligned} \tag{31}$$

2. Define mapping between observations and state via $\tilde{H}$. For a GPS receiver the observations map directly to the states. If the GPS receiver provides position data only, then m in the Equation (32) is 3; if velocity measurements are also available then m equals 6.

$$\begin{aligned} \tilde{H} &= O_{m \text{ x } n} \\ \tilde{H}(1{:}m, 1{:}m) &= I_{m \text{ x } m} \end{aligned} \tag{32}$$

3. Determine a Sensor Noise Matrix, R. The values for σ_{pos} and σ_{vel} provided are used for simulations and analysis unless otherwise stated [14].

$$\begin{aligned} R &= \text{diag}\left(\sigma_{pos}, \sigma_{pos}, \sigma_{pos}, \sigma_{vel}, \sigma_{vel}, \sigma_{vel}\right) \\ \sigma_{pos} &= 3.33 \times 10^{-3} \\ \sigma_{vel} &= 3.33 \times 10^{-6} \end{aligned} \tag{33}$$

4. Define a Model Noise Matrix, Q, for the specific time step. See section 3.2.2 for details on the selection of these parameters.

$$Q = \text{diag}\left(1 \times 10^{-9}, 1 \times 10^{-9}, 1 \times 10^{-9}, 1 \times 10^{-11}, 1 \times 10^{-11}, 1 \times 10^{-11}, 0, 0, 0, 0, 0, 0\right) \tag{34}$$

5. Select an initial Covariance Matrix, P. The selection of these values was based on estimating the relative error for each initial parameter. For the diagonal entries corresponding to the 6D

state the same variance as the GPS receiver is used. The remaining six parameters are chosen such that the expected error is a few orders of magnitude less than the initial condition.

$$\begin{aligned} P_0 = \text{diag}(\quad & 1 \quad \times 10^{-9}, 1 \times 10^{-9}, 1 \times 10^{-9}, 1 \times 10^{-11}, 1 \times 10^{-11}, 1 \times 10^{-11}, ... \\ & 1 \quad \times 10^{-1}, 1 \times 10^{-5}, 1 \times 10^{-10}, 1 \times 10^{-10}, 1 \times 10^{-1}, 1 \times 10^{5}) \end{aligned} \tag{35}$$

6. Propagate the state and STM forward in time until a new observation is available. The force model and integration routine discussed in Chapter 2 are utilized.

$$\begin{aligned} \left[\dot{\vec{x}}, \dot{\Phi}\right] &= forceModelSTM\left(\vec{x}_{t-1}, I_{n \times n}, t-1\right) \\ \vec{x}_t &= \vec{x}_{t-1} + \dot{\vec{x}} \end{aligned} \tag{36}$$

7. Calculate intermediate Covariance Matrix, $\bar{P}$.

$$\bar{P} = \Phi * P_{t-1} * \Phi^T + Q \tag{37}$$

8. Compute observation-state matrix, H.

$$H = \tilde{T} * \Phi \tag{38}$$

9. Calculate difference between observation and linearized model prediction.

$$\vec{y}_t = Y_t - \tilde{H} * \vec{x}_t \tag{39}$$

10. Compute the optimal Kalman gain.

$$K = \bar{P} * H^T * \left(H * \bar{P} * H^T + R\right)^{-1} \tag{40}$$

11. Calculate the new estimate of state.

$$\hat{\vec{x}}_t = \vec{x}_t + K * \vec{y} \tag{41}$$

12. Update estimate of Covariance Matrix.

$$P = (I_{n \times n} - K * H) * \bar{P} \tag{42}$$

13. Using the new state estimate and updated covariance matrix, go to Step 6 and continue.

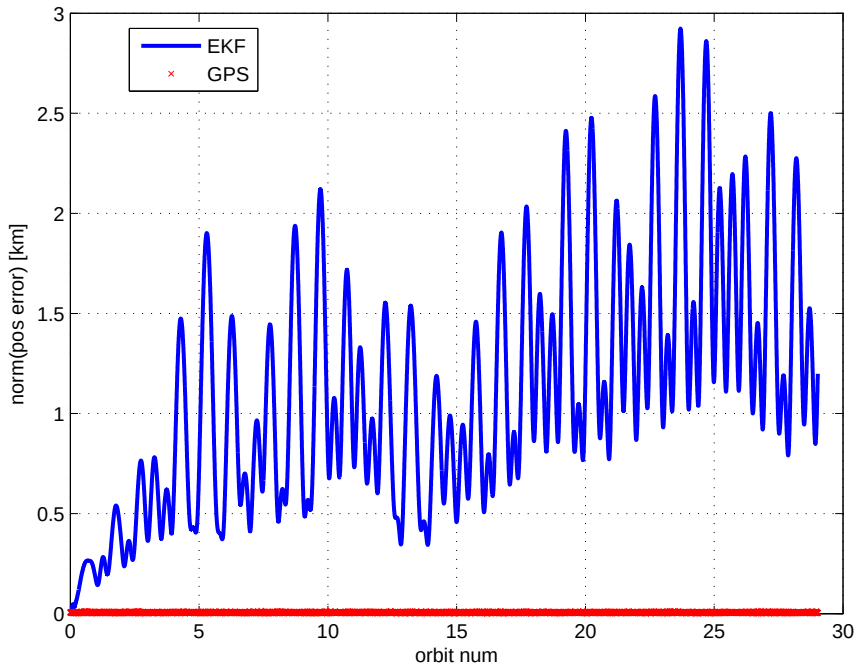

Figure 32: By setting the Model Noise Matrix, Q, to a matrix of zeroes, the filter solution diverges.

3.2.2 Model Noise Matrix, Q

Early in the EKF design process it was determined that the long-term accuracy of the orbit determination solution was highly dependent on the selection of the Model Noise Matrix (a.k.a. Process Noise Matrix or State Noise Compensation Matrix), represented in the EKF equations by Q. If Q is neglected altogether (set to a matrix of zeroes), the solution diverges resulting in a useless orbit determination algorithm [63]. Figure 32 shows how in just 30 orbits (about two days), the filter is already diverging. What appears at the bottom of Figure 32 is actually the noisy GPS measurements that are feeding into the EKF; from this plot one might suggest to drop the filter altogether and use the raw GPS output. In essence, not adding model noise to the filter results in the Kalman Gain matrix decreasing in value until it is at or near zero. Such a low gain results in new sensor measurements being ignored. By adding noise to the dynamic model, one is ensuring that the filter does not "trust" the model too much and places a sufficient priority on the new incoming measurements.

A small trade space was explored in order to chose a reasonable Q; this process is generally referred to as "tuning" the filter. The Model Noise Matrix is usually set to values that correspond to

the square of anticipated error of the state over a period of time, given the selected dynamic model [17]. From Chapter 2, the worst case position error over a 15 second period is on the order of meters, but generally in the area of centimeters. Similarly, the velocity error is on the order of centimeters per second, but generally millimeters per second over such a short period of time. From here initial guesses for the diagonal entries in Q for a 15 second time step are 1×10^{-12} for the position noise and 1×10^{-14} for the velocity noise. Since 1×10^{-14} is already approaching machine precision, a trade space of 1×10^{-6} to 1×10^{-15} per 15 second time step for both the position and velocity model noise is selected, where various combinations of values are paired.

First, it was found that by selecting noise values that were too large, the solution would diverge even more rapidly than if no noise were added at all. An example where the diagonal entries in Q were all set to 1×10^{-6} is provided in Figure 33; one can see that the EKF diverges quicker than in Figure 32. For the opposite case of selecting very small values of noise, the EKF solution does not diverge but has a large steady-state error far greater than the GPS measurement noise. Figure 34, which used a Q with diagonal values all set to 1E–15, shows that the EKF solution does not diverge, but has an average steady-state error of 50 m or more.

While the extrema provide poor performance, settling in the middle of the trade-space range provides accurate convergence results for the filter. An interesting realization during the trade study is that the filter is effected far greater by the selection of the velocity noise rather than the position noise. Considering the derivative function, this seems logical since the velocity is used directly as the derivative of position imparting a greater effect than any change in forces due to meter-sized error in position. Results, analysis, and figures follow in the next section .

3.2.3 Extended Kalman Filter Results

After tuning the propagation technique, force model, and filter initial conditions the convergence of the orbit determination algorithm is tested. For a filter with an appropriate force model, an extended Kalman filter should provide state estimates with variance less than that of any available sensor data. To first test the filter, data was generated using the designed force model discussed in section 2.3.5 and an RK4 integration routine with a time step of 15 seconds. Notice that this time step is less than those identified in Chapter 2; although the 45 or 90 second time step has nearly the same accuracy

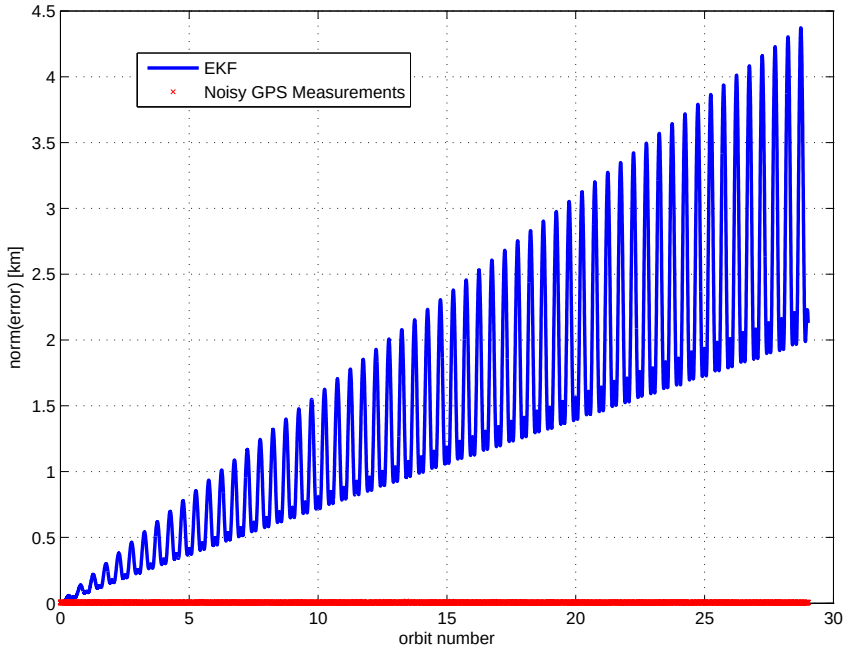

Figure 33: When the Model Noise Matrix, Q, uses large diagonal values such as 1×10^{-6}, the filter diverges even more rapidly than when no model noise is added.

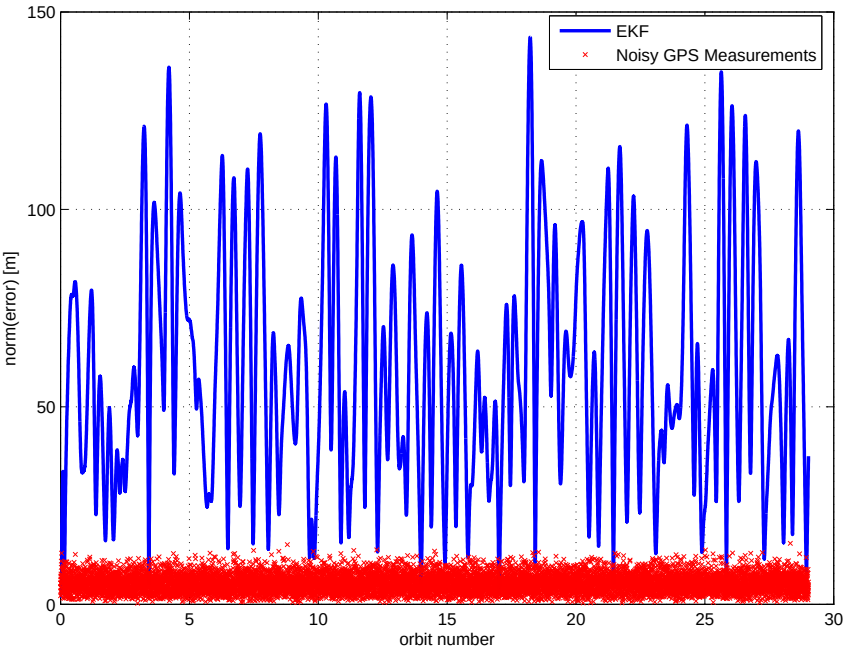

Figure 34: Small diagonal values in the Model Noise Matrix, Q, such as 1×10^{-15} result in the filter converging to a solution with a large steady-state error.

as the 15 second time step for propagation, filter performance is far better with GPS observations every 15 seconds. It is advantageous that the time between GPS observations and the propagation time step are the same (or the propagation time step at least being less than the GPS sample rate), simplifying the combination of the two state predictions in the EKF.

Since our first test generated noisy sensor data using the exact model implemented in the EKF, the filter should converge to an almost perfect estimate. Examining Figures 35 and 36 proves that including about a quarter-orbit warm-up period, the position and velocity estimates over 10 orbits are more accurate than the GPS observation 96.04% and 98.59% of the time, respectively.

A second statistic to consider is how close the final model "constants" including the gravitational parameters, J_2, and drag and SRP coefficients are to the actual model ones. Figure 37 depicts the error of each of the six model parameters being estimated over the period of 10 orbits. Each of the three gravitational parameters' estimation error all hover near zero and are 9 to 15 orders of magnitude less than the actual value. Similarly the J_2 coefficient error settles near three orders of magnitude less the actual model parameter. Although the absolute error for the drag and SRP coefficients is small, relatively speaking it is large. Nonetheless, the behavior of the coefficients settling at small positive values and the accurate estimation of the primary model parameters indicates the filter is operating properly.

After validating the filter is working from testing with data generated using the same force model, use of the *FreeFlyer* reference orbit is employed. Like in the first test, noise is added to the truth data with variance equal to the GPS receiver parameters provide in Equation(33). Figures 38 and 39 show similar results as those when the same model (with added noise) is used to generate the input data. When comparing the accuracy of the filter results to the GPS observation it is found that the position and velocity estimates are more accurate than the observations 92.24% and 93.98% of the time, respectively. With the sensor data coming from a more complex model than the one within the EKF, the warm-up period is slightly longer, appearing to be about half an orbit now.

A final set of tests of the filter removed velocity input altogether and tested the convergence with only 10 m accurate position measurements. As seen in Figure 40, the error is slightly greater than when velocity measurements are also available, but in general the filtered solution has a third to half the error compared to straight GPS measurements. For the velocity, the error is only about

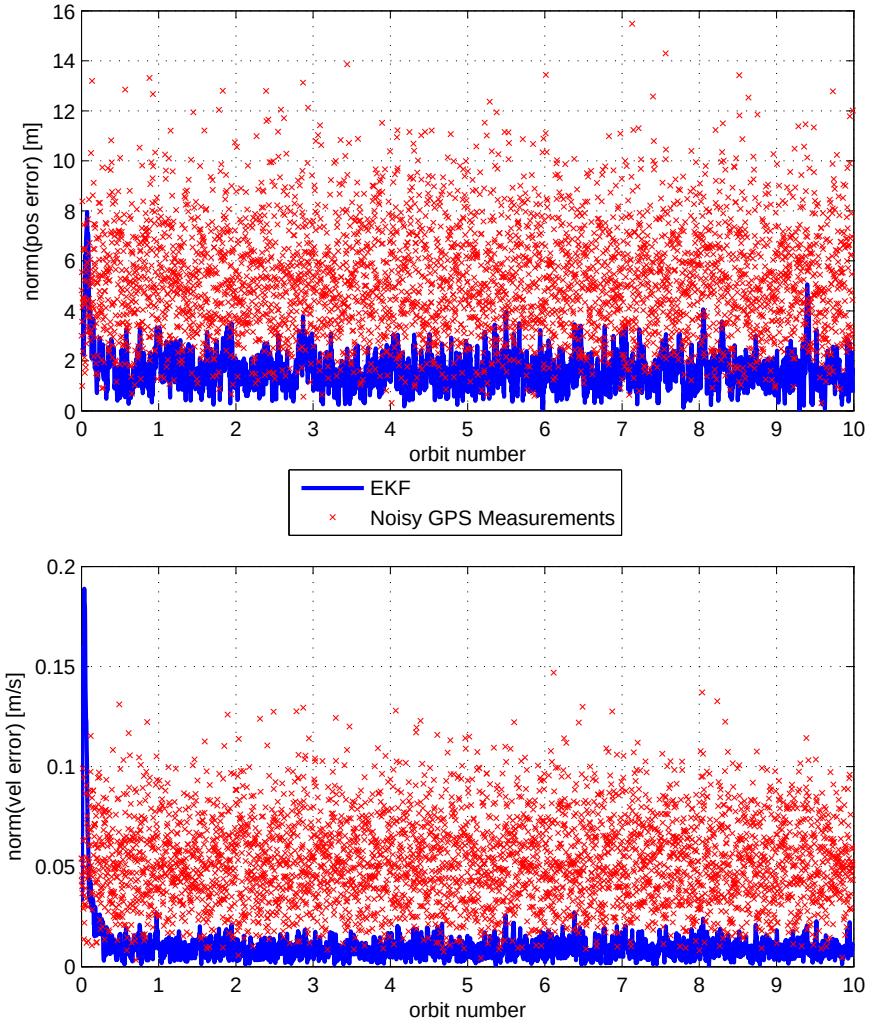

Figure 35: The first validation test of the EKF uses the selected force model described in Chapter 2 to generate data. Noise is added to the data assuming it comes from a sensor with 10 m and 100 $^{cm}/_{s}$ accuracies. The filter converges with accuracy (determined by comparing to the model "truth" without noise) 5 times better than the "sensor" for position and 5 to 10 times better than the "sensor" for velocity.

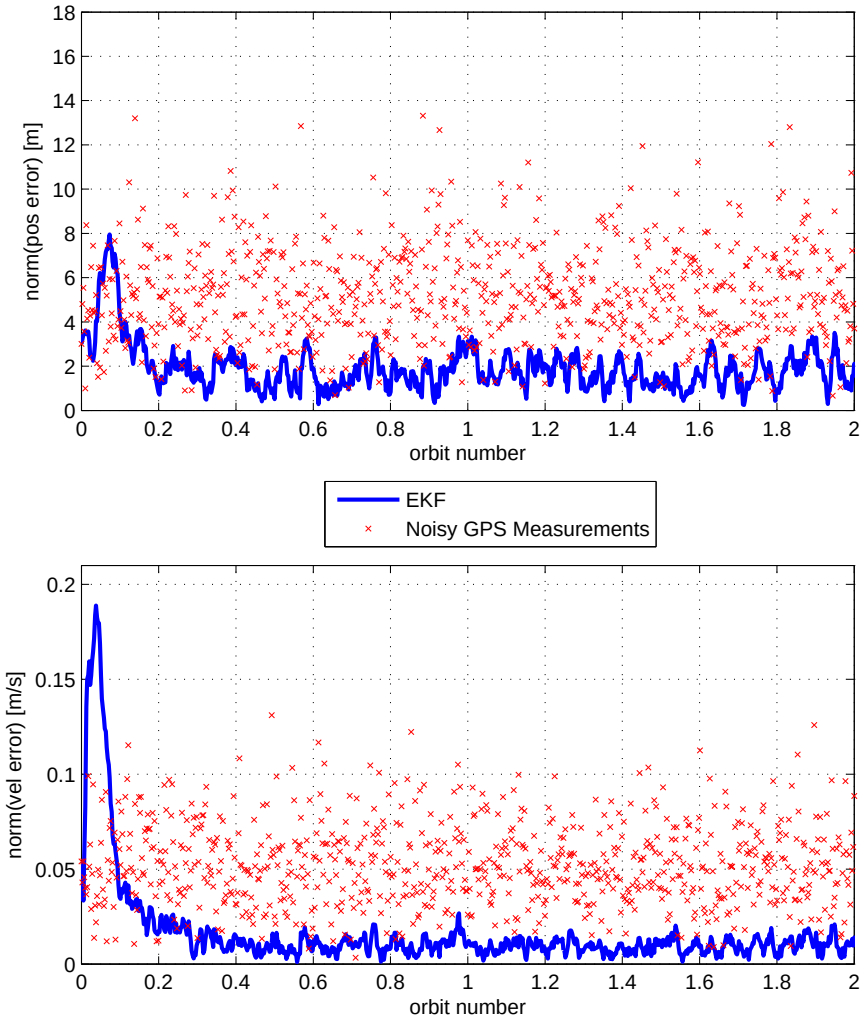

Figure 36: Displaying 2 rather than 10 orbits as in Figure 35 clearly depicts the quarter-orbit warm-up period for the filter to converge.

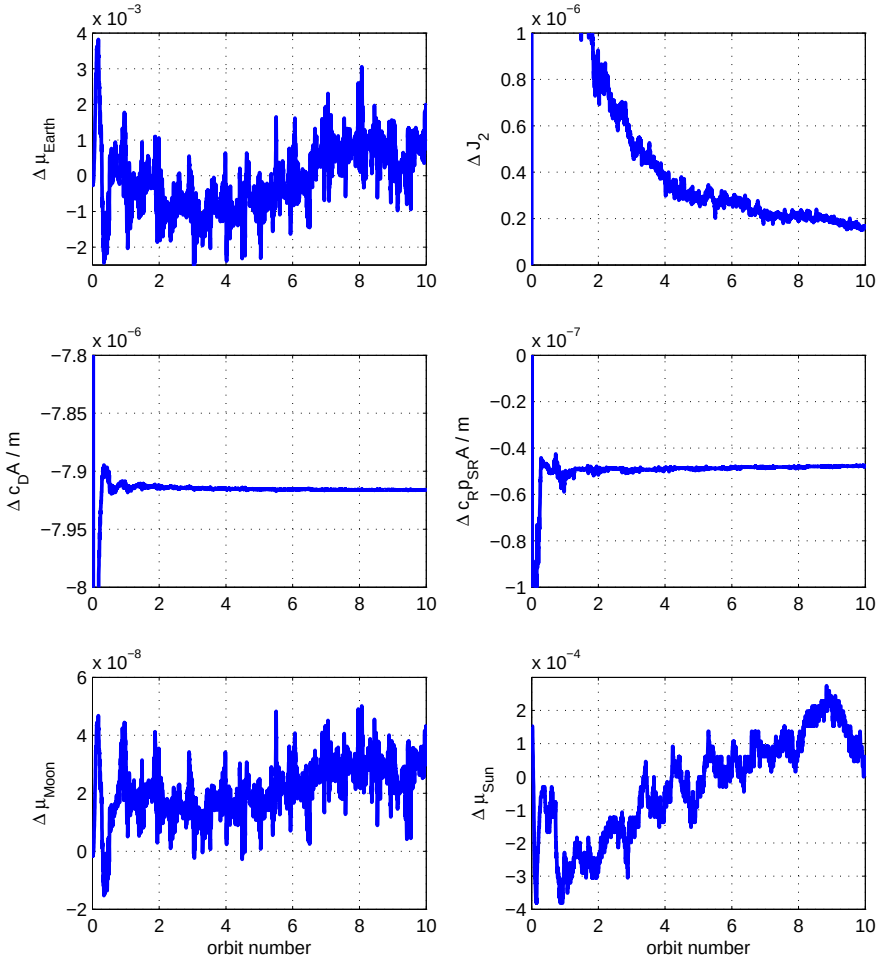

Figure 37: Since the EKF is also estimating six model parameters, these values should settle near zero since the first set of data is generated using a known set of six parameters. In general, the filter performs well in estimating the parameters.

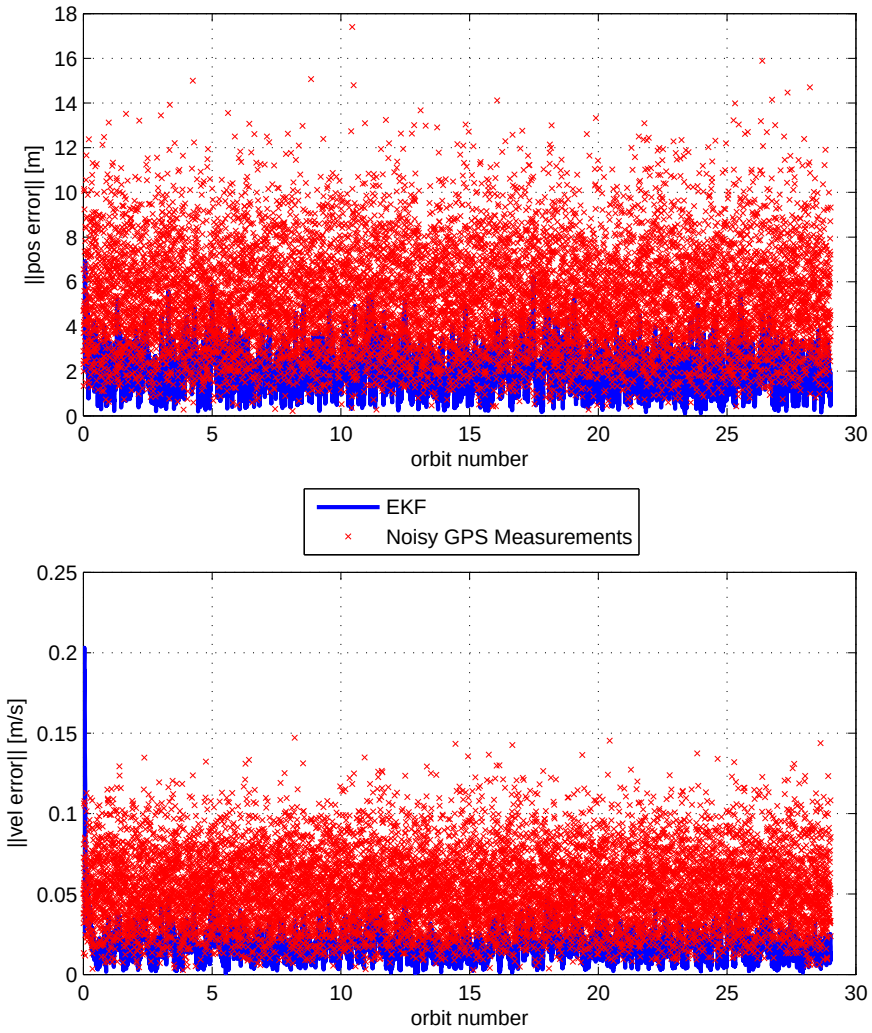

Figure 38: The filter performs well when tested with *FreeFlyer* data, providing nearly the same accuracy as the filter did when compared using noisy data from the same model. The error here is the difference between the measurement or filter solution and the *FreeFlyer* "truth."

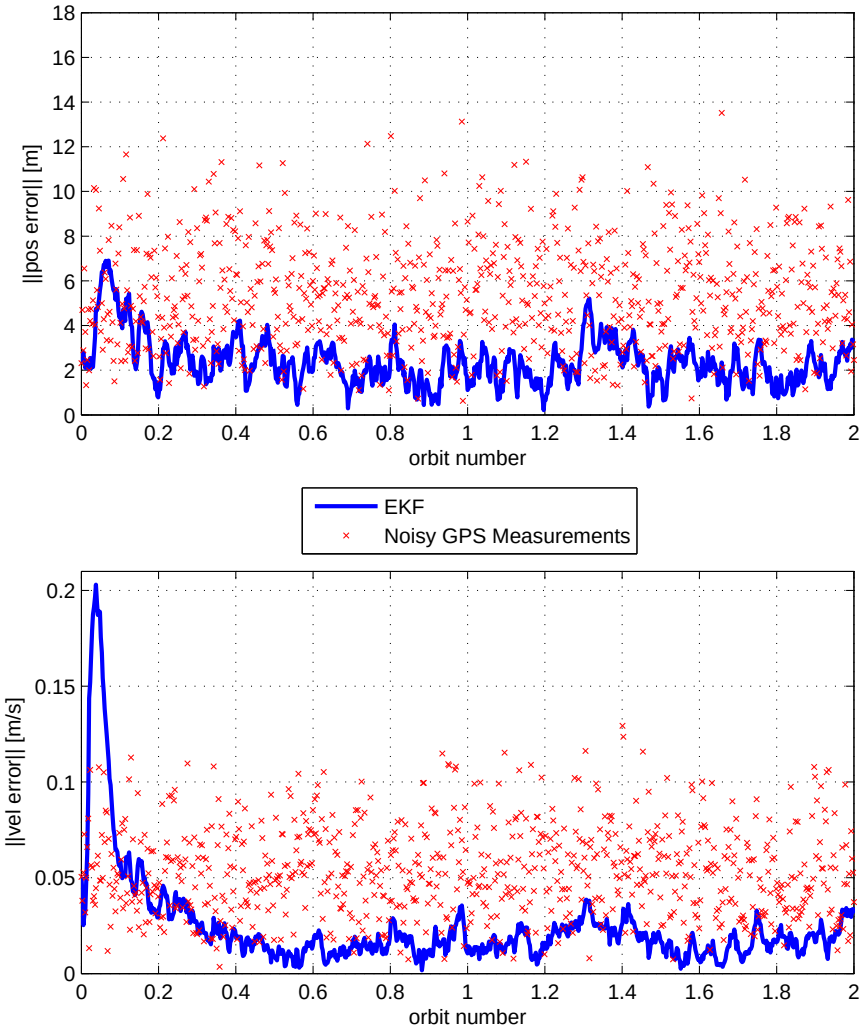

Figure 39: Zooming in on 2 orbits from Figure 38 shows a slightly extended warm-up period of about half an orbit.

twice that of the solution calculated when 100 $^{cm}/_{s}$ velocity data is available to the EKF.

With a well-performing on-board orbit determination algorithm, the pieces required for a single satellite to track team member orbits and accurately maintain its own state are complete. Selection of an appropriate broadcast message and communications framework is the final need for spacecraft to update one another. Chapter 4 next describes the selected message framework and final protocol, RF band, and hardware chosen. Chapter 5 follows by describing the algorithm for individual spacecraft to broadcast new updates as well as provides results showing the scenarios where use of the EKF significantly increases the time between broadcast messages.

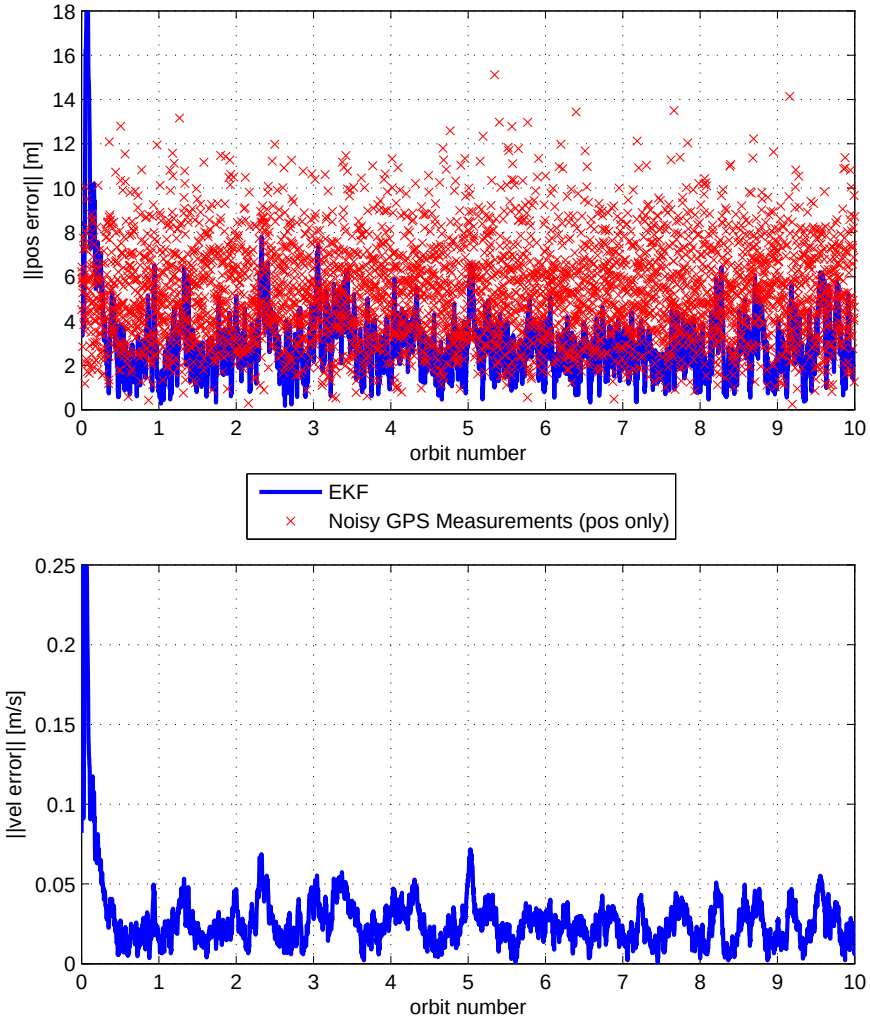

Figure 40: Not all GPS receivers provide velocity output, therefore the filter is also tested with a scenario when only position data is provided

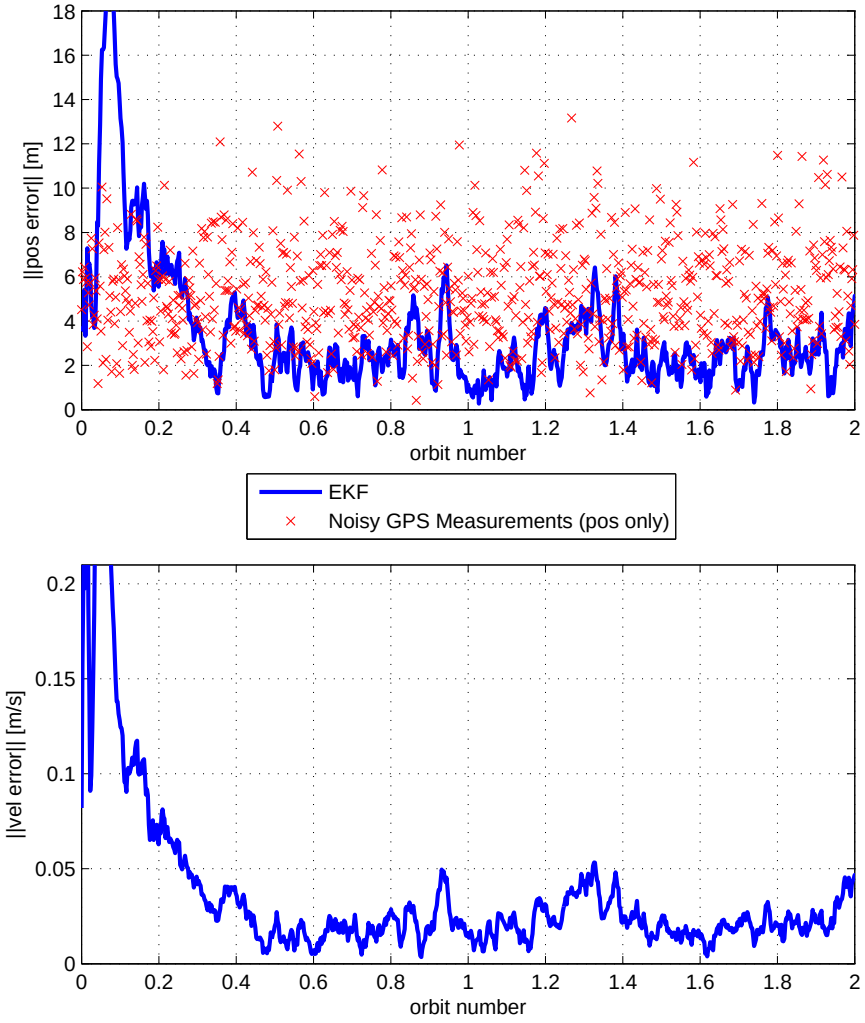

Figure 41: Even with the lack of velocity measurements, the filter still converges in about half an orbit.

CHAPTER IV

INTER-SATELLITE COMMUNICATION

Enabling autonomous collaboration between CubeSats requires communication among the team. Although ISLs are not uncommon in larger satellites, little has been accomplished in inter-satellite communication for small satellites of the nano or pico scale. As will be discussed throughout this chapter, an S-band COTS solution (with flight heritage) for both the hardware and protocol has been selected for the collaborative module. By selecting a commercially available solution, the developed software solely needs the ability to maintain a wired serial communications link with the S-band module. Details on the message structure and the down-select process of protocol, RF band, and hardware follow.

4.1 Message Structure(s)

Before the execution of protocol or hardware trade studies, defining the necessary message structures for the collaborative module is necessary. Besides the aforementioned ephemeris update messages, a wrapper or message preface provides a simple framework for other inter-satellite communication. Details on the message wrapper, ephemeris update message, and sharing of sensor data follows.

4.1.1 Message Wrapper

Using a well-defined protocol, such as those considered in this research, assists in data integrity with bit error checking and ensuring that packets received out of order are properly reordered and processed correctly. With these tasks already being handled by the protocol, only an additional small wrapper or preface is utilized in the collaborative module. Table 6 provides the list of six parameters consuming 29 bytes of data.

The *Orig SC/Module ID* is a unique integer value that indicates either the spacecraft or a specific module (if so desired) on a specific spacecraft within a team. A 2-byte integer provides over $65,000$ unique identifiers, which far exceeds any anticipated cluster size even if each spacecraft has multiple

Table 6: A generic 29-byte message wrapper (or preface) provides the information a spacecraft team member requires to process the appended data.

byte	0 - - - 1	2 - - - 3	4 - - - 11
parameter	Orig SC/Module ID	Destination ID	Message ID
byte	12	13 - - - 20	21 - - - 28
parameter	Message Type	Timestamp	Check Sum

modules which a mission requires to name individually. The only value withheld from the set of unique identifiers is 0; zero is reserved to indicate "all" when conducting a one-to-all broadcast. The *Destination ID* is the same set of integer values used for the originating spacecraft or module identifier; the one exception is that a value of 0 indicates the message is a one-to-all broadcast and is not intended for an individual spacecraft or subsystem. A *Message ID* is utilized so that if a spacecraft broadcasts the same message multiple times, a receiver can disregard it as already processed. Messages may be sent multiple times if acknowledgements are not received by the original sender. *Message Type* is defined by a single byte; an example message type would be the state sharing message discussed in the following section. Additional message types may be defined by a specific mission such as telemetry/health sharing or sensor data necessary to execute a collaborative task. Inclusion of a *Timestamp* is good practice so that a receiver can ensure it is the most recent message of a specific message category (type) sent from a spacecraft; the *Message ID* is an incrementing value, but use of a timestamp provides an additional check. Finally, the wrapper provides a *Check Sum* calculated by using the encapsulated data. After a spacecraft completes receiving an entire message it can run a predefined check sum function on the data. A receiver should calculate the same value and verify the integrity of the received data by comparing it to the check sum received in the message preface. The over-arching protocol provides similar functionality on a lower level, but it is good practice to provide a check sum at the high level to ensure data integrity.

4.1.2 State Message

The contents of the primary state sharing message are a result of the state selected for integration, discussed extensively in Chapters 2 and 3. Besides the thirteen parameters that define the state

Table 7: The spacecraft state broadcast message is composed of 20 different parameters, each of which is provided 8 bytes for double precision. Note that although bytes are allocated for the attitude and attitude rates, these portion of the state is not implemented in this thesis work.

byte	0 - - - 7	8 - - - 15	16 - - - 23	24 - - - 31
parameter	x [km]	y [km]	z [km]	u [$^{km}/_{s}$]
byte	32 - - - 39	40 - - - 47	48 - - - 55	56 - - - 63
parameter	v [$^{km}/_{s}$]	z [$^{km}/_{s}$]	q_0	q_1
byte	64 - - - 71	72 - - - 79	80 - - - 87	88 - - - 95
parameter	q_2	q_3	p [$^{rad}/_{s}$]	q [$^{rad}/_{s}$]
byte	96 - - - 103	104 - - - 111	112 - - - 119	120 - - - 127
parameter	r [$^{rad}/_{s}$]	t_0 [sec]	$\Delta\mu_{\oplus}$ [$^{km^3}/_{s^2}$]	ΔJ_2
byte	128 - - - 135	136 - - - 143	144 - - - 151	152 - - - 159
parameter	$\frac{c_D A}{m}$ [$^{m^2}/_{kg}$]	$\frac{p_S c_R A}{m}$ [$^{N}/_{kg}$]	$\Delta\mu_{\odot}$ [$^{km^3}/_{s^2}$]	$\Delta\mu_{\leftmoon}$ [$^{km^3}/_{s^2}$]

(position, velocity, attitude, and attitude rates), seven additional parameters have been determined as necessary for sharing between formation team members. These seven parameters are the gravitational parameters of the Earth, Sun, and Moon; the Earth's J_2 term; the coefficients used for calculating drag and SRP forces; and the corresponding time defined in seconds since a globally selected epoch. Note that for the three gravitational parameters and J_2 rather than sending the whole value the difference between the spacecraft's estimated value and a globally defined reference value is transmitted. For each value in the message 8 bytes are used to represent the number in double precision. Table 7 summarizes the message structure. With a message of this size, common data rates and protocols will complete transmission of the the 160 bytes with a single packet and in well under a second.

Recall from section 4.1.1 that the generic message wrapper is appended to the front of that state sharing data to ensure that a receiving spacecraft can properly interpret the parameters. Beyond this wrapper, the selected proprietary protocol handles the packetization and ensuring delivery of an entire set of message packets.

4.1.3 Sensor Data

Sharing of sensor data is mission specific. Therefore, no set framework is provided for the distribution of data from one spacecraft to another other than the message wrapper discussion in section

4.1.1. For a specific mission where sensor data sharing is desired the only required tasks are the defining of a message type with given parameters (including the order and allocated size of each parameter) and the providing of a check sum function. Appending the message preface and ensuring delivery of data to intended receivers (via acknowledgement) is handled by the collaborative module.

4.2 Protocol, Frequency Band, and Hardware Selection

Understanding the common message size and data to transmit, while considering the unknown mission-specific sensor data transmissions, trades between protocols could be conducted. A simple yet robust solution is desirous. Although the aforementioned message structure is tailored to the ephemeris data broadcast among CubeSats, selection of a versatile protocol capable of handling packetized data ensures it would meet future ISL needs. The driving factor behind this versatility was to ensure the same protocol could be used for transmitting of mission-specific sensor data between CubeSats. As stated multiple times previously, the end goal of this research is to provide a single hardware plus software module that enables collaborative behavior among CubeSats; enabling larger data transfer is a necessity for such a module to have multiple mission applications. Additional factors that played into the trade studies were average data rate, maximum range, robustness, availability of COTS hardware, and TRL.

The concept of small satellite formations has resulted in the electrical engineering community exploring existing terrestrial protocols to see how robust they are to the LEO environment. In recent years studies have been conducted to determine how common protocols like 802.11 (WiFi) or the newer 802.16 (WiMax) would perform in orbit. Since both of these protocols have plentiful COTS components (although not radiation hardened), have high TRLs, and are known for sufficient data rates for their terrestrial applications both of these were further explored.

Commonality of use among previous CubeSat missions drove selection of a third protocol for examination: X.25 also referred to as AX.25. This protocol used by the amateur radio community is by far the most prevalent from surveying the list of CubeSat missions [64]. Furthermore, this same protocol could be used for ground communications, limiting the total on-board software complexity through code reuse.

After considering specific COTS hardware, a final proprietary serial communications protocol was considered and finally selected. Details on these four protocol options follow concluding with a summary comparison and explanation as to the protocol and hardware selection.

4.2.1 WiFi (IEEE 802.11)

Over the past decade terrestrial WiFi has become ubiquitous. This robust protocol has made the connectivity of various electronic devices to networks and the Internet seamless for an end-user. Given its ability to provide data rates well above 1 Mbps, the option of setting up an *ad-hoc* network of nodes, and the transparent handling of packetized data make this protocol intriguing.

Since 802.11 is designed for terrestrial applications with propagation lengths on the order of hundreds meters or less, adaptations to the protocol are required for the possible kilometer plus distances in LEO. Specifically, protocol parameters such as the slot time and acknowledgement timeout can be optimized (increased to an appropriate value depending on anticipated propagation length) [60, 68]. Research through simulation and analysis has indicated that through parameter optimization WiFi could become a viable ISL protocol. Beyond protocol parameter modification, pairing with a "smart" antenna such as a multiple-input multiple-output (MIMO) or adaptive antenna system (AAS) provide a high-performing ISL from slightly modified COTS components [60].

Even with the necessary modifications to 802.11 to make it fit for an ISL network, a second factor to consider is scalability. For system robustness, an ad-hoc network, as opposed to a hub and spoke as is common for terrestrial applications, should be selected. If a hub and spoke setup were utilized and the hub satellite became disabled the entire network would fail. Additionally, if a spoke satellite was perturbed such that it moved outside the range of the hub it would lose connectivity with the entire network system even if a second spoke satellite was in range. An ad-hoc network, although containing disadvantages itself, provides system robustness through removing single points of failure. One primary disadvantage of ad-hoc networks is the need to design or select routing algorithms. Analysis of ad-hoc scalability for terrestrial mobile networks, including routing algorithm selection, has indicated that although total throughput does degrade rather quickly, for systems on the order of a few dozen nodes or less adequate throughput of 0.2 Mbps is still capable when all nodes are within one-hop communication distances with each other [16, 44, 68].

Throughput also decreases as the chain length between two nodes increases, as expected, therefore, depending on the relative orbit scenario, additional analysis may be required. Still, as in the case of total nodes, although data throughput decrease rapidly at first, the plateau around a 10 node chain length is hundreds of kilobits per second [16, 44]. Higher throughput requirements may be necessary depending on the sensor data requirements, but any link on the order of kilobits per second is sufficient to rapidly transmit an ephemeris update message.

One of the primary attractions to WiFi as a protocol of choice is the extensive use of transceivers on low-power mobile devices. To restate, one of the primary goals of this research is to develop a module to enable collaborative behaviors and improve system performance by increasing the available power for payloads. By selecting an appropriately powerful transceiver (or even better one that can set the output power programmatically), energy savings could be realized not only through reducing the frequency of communication but also the power necessary to communicate. An example implementation of a transmit power control 802.11 network showed that the transmit power could be reduced up to 100 times the default power setting and still successfully deliver packets [59]. In that implementation a receiver calculates the received signal strength (RSSI) and from that determines the optimal broadcast power for future transmissions. When communication is first initiated, a relay message from the receiver to the sender is necessary, but unless the RSSI significantly changes during future transmissions due to node movement or interference, no relay message with a new optimal transmit power setting is required. This technique could be applied to any transceiver package which provides a means to manipulate the transmit power.

Use of COTS components for the collaborative module is desired, even if slight modifications to the original protocol is necessary, making 802.11 an attractive choice. On the contrary, WiFi hardware for the LEO environment is rare if not nonexistent. Additionally, the works referenced above did take moving systems into account, with motion on the order of meters per second. For a satellite system, the relative velocities would be similar to this order of magnitude, but before selecting WiFi hardware for the module additional testing and simulation is necessary to ensure the COTS receivers perform well in the LEO environment in general.

4.2.2 WiMax (IEEE 802.16)

Often thought of as a next-generation WiFi, the WiMax protocol, which has been gaining popularity over the past decade, provides broadband data links to devices with ranges on the order of miles [58]. While WiFi ranges and data rates are appropriate for formations on a small scale, WiMax's kilometer-scale range and order of magnitude improvement in data throughput may enable ISLs for near constellation-sized systems [28]. Similar to the experiments and simulations conducted with a modified WiFi, The Aerospace Corporation has examined the possibility of using WiMax to create a space-based local area network [58]. Since WiMax is already designed for networks spread across miles of area, the modifications to timeout parameters, etc. are not required. Using COTS WiMax hardware, The Aerospace Corporation demonstrated a network of multiple mission satellites communicating with a "hub" at speeds up to 6 Mbps and propagation distances of 17 kilometers. Although throughput per node decreased as the number of nodes connected to the hub increased, the protocol and hardware implementation was sufficiently robust to handle nodes entering and leaving the network. Aerospace's demonstration indicates the reason for concern of using a hub spacecraft as opposed to an ad-hoc network with simpler node-to-node links. While being a simpler setup, loss of the hub results in all data exchange ceasing; furthermore, while operating the throughput between two nodes is fully dependent on the amount of traffic the hub spacecraft is handling. Since WiMax does not support direct ad-hoc networks, use of WiMax for the ISL would require any satellite system to have a hub or base station, introducing the single point of failure concern. Furthermore, low-power COTS components supporting WiMax are becoming more readily available, but personal access points are rare and expensive as opposed to the inexpensive WiFi hardware found in nearly every household.

4.2.3 AX.25

One of the most widely used protocols globally is AX.25. The global amateur radio community adopted the protocol as its standard therefore ample hardware and open-source software is available to those interested. The protocol uses a relatively simple structure composed of three types of frames with each frame containing a few fields. A unique 8-bit field, referred to as a flag, begins and ends each frame. The 8-bit string is guaranteed unique through bit stuffing. Additionally, an address,

control, data or info, and frame check sequence field are provided. The info field is limited to 256 octets, so depending on the size of sensor data to be transmitted data packetization may be necessary and is not inherently handled by the AX.25 [19]. Use of published code or design of packet handling software is a relatively simple task, but handling of packets is transparent to the end user for 802.11, 802.16, and the **Microhard Systems, Inc.** proprietary protocol discussed shortly.

Originally designed to satisfy a need from the amateur radio community to have a set standard for the delivery of information between ground-based terminals, the CubeSat community quickly adopted AX.25 as a popular protocol for ground-to-space links. Due to licensing restrictions and the cost and availability of ground station hardware, the majority of CubeSats to date have selected a UHF frequency around 430 or 440 MHz for data uplink and downlink [64]. These selections have allowed for global involvement from the amateur radio community to assist CubeSat developers by sending and receiving transmissions to the satellites.

With the UHF design choice being common and well-proven, selecting it as the primary data link to the ground seems natural. Assuming this design selection is made, a CubeSat would already have the necessary hardware and software for handling AX.25 communications over UHF making the only addition an omni-directional antenna for inter-satellite communication. If, however, an encrypted channel or link providing greater throughput were needed for mission data downlink, no overlap of hardware or software may be available. Since this research is in response to government initiatives such as System F6 and SENSE, use of S-Band rather than UHF as the primary satellite communications link is anticipated.

With the CubeSat power, mass, and volume restrictions, it is best to design with component reuse in mind. Therefore, since selection of S-Band for ground communications is appropriate, the use of the same band for the ISL is almost necessary. AX.25, being a protocol, can in theory be utilized on any RF band, but use on a 2.4 GHz frequency has been determined to be very rare. Furthermore, AX.25 is not designed for the transfer of protected or encrypted data, therefore this reduces the flexibility of the ISL implementation. Through examination of commercially available solutions, the final protocol discussed in the following section was identified.

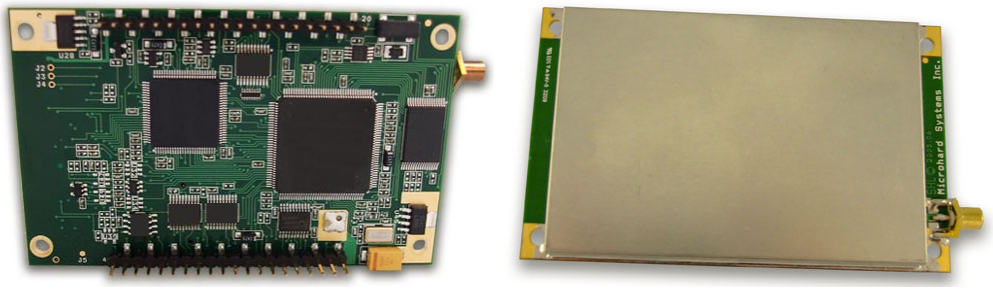

Figure 42: Top and bottom views of the *Mircrohard* MXH-2420 spread spectrum frequency hopping 2.4 GHz modem, designed specifically for the CubeSat framework [9].

4.2.4 Microhard Systems, Inc. MHX-2420

Protocol and hardware trade studies were conducted concurrently; therefore, even though proprietary protocols were not originally considered, examination of hardware solutions led to such protocols. Most CubeSats select the amateur radio spectrum in the UHF band around 400 MHz, but some choose the higher-frequency S-band in the neighborhood of 2.4 GHz. One example is NASA's GeneSat, which chose an S-Band receiver due to data downlink requirements; using the higher frequency provides potential for increase data rates [47]. Specifically, NASA used **Microhard, Inc.**'s *MHX-2400* transceiver. Originally designed as a plug-and-play solution for terrestrial-based applications needing RF links on the order of miles, **Microhard** now provides a modified version designed specifically to interface with **Pumpkin, Inc.**'s CubeSat modules [10].

The MHX-2420, depicted in Figure 42, is a spread spectrum solution that uses frequency hopping. It provides data rates with extended sensitivity of 19.2 up to 230.4 kbps, variable output power ranging from 100 mW to 1 W, and with line of sight can transmit up to 50 km. Even though the functionality of identifying a message for a specific spacecraft is built into the message header, the MHX-2420 can also operate in either point-to-point or point-to-multipoint modes. Specifically designed with the CubeSat framework in mind, the modem provides a low power sleep mode, weighs only 55 grams, integrates directly with the command and data handling system developed by **Pumpkin, Inc.**, and has an industrial-grade temperature survivability range. Finally, all models offer a bit 32-bit CRC (check sum) and forward error bit correction with retransmission; certain models provide support for 128-AES encryption [10].

MHX-2420 provides multiple protocol selection options including MODBUS, but the default

Table 8: **Microhard Systems, Inc.'s** *MHX-2420* provides 11 discrete output power settings so that the minimum power level required for successful transmission may be selected.

dBm	mW
20	100
21	125
22	160
23	200
24	250
25	320
26	400
27	500
28	630
29	800
30	1000

transparent protocol has been utilized; this allows for plug-and-play operation of the transceiver over serial communications [11]. The modem handles the packetizing, modulating and demodulating, and bit error correction seamlessly to provide a robust RF link. Code was easily completed to provide a serial I/O link with the modem; delivery of data to the modem over serial is all that is required to initiate a broadcast removing the complexities of initiating an RF link. The largest packet size is 255 bytes, so the selected message with wrapper can fit into a single packet. Testing of larger data objects, specifically arrays on the order of 1 MB, was conducted to prove the integrated protocol's successful handling and delivering of packets.

Transmit power selection is a set of 11 discrete options, provided in Table 8. Using the same serial link for data transmission the output power setting can be changed at any time. In order to appropriately set the output power, the average received signal strength (RSSI) for the preceding four hop intervals is made available by the modem. It is desirable for the system (which here is defined as one transmitter and one receiver) to have a fade margin of at least 20 dBm [11]; fade margin is defined by Equation (43) [11]. By using the RSSI value and Equations (43), (44), and (45) each spacecraft can set its output power to the lowest value required [11, 42].

$$\text{Fade Margin} = \text{System Gain} - \text{Path Loss} \tag{43}$$

$$\begin{aligned} \text{System Gain} &= \text{Tx Power} + \text{Tx Antenna Gain} - \text{Tx Cable/Connector Losses} \\ &+ \text{Rx Antenna Gain} - \text{Rx Cable/Connector Losses} + |\text{Rx Sensitivity}| \end{aligned} \quad (44)$$

$$\begin{aligned} \text{Free-Space Path Loss} &= 20\log_{10}(d) + 20\log_{10}(f) - 147.55 \quad (45) \\ d &= \text{propagation length measured in meters} \\ f &= \text{frequency measured in Hertz} \end{aligned}$$

At a data rate of 19.2 kbps the transceiver provides a receive sensitivity of at least -108 dBm [11]. Since each spacecraft is tracking the position of each other team member, a good estimate of the propagation distance between itself and the furthest spacecraft is easily obtained. Assuming a carrier frequency of 2.4 GHz and the largest estimated propagation distance, Equation (45) provides a usable estimate of free-space path loss. Depending on the selected antennas and antenna orientations, the system gain can vary. For this study omni-directional "rubber ducky" antennas were utilized. Each antenna has a 3 dBi gain with an estimated 1 dB gain loss due to cabling and connectors; these values were calculated by using the published total system gain and backing the antenna gains and loss out [10]. Even without a real-time RSSI measurement, sufficient data is available to determine a minimum transmit power; this value is used at the start to set the transmit power before any RSSI measurements are available.

Since there could be other unknown system losses, possibly due to spacecraft orientations or other, the collaborative module could use the available RSSI value to appropriately set the power (likely increase it), if needed. Equation (46) provides a means to calculate the transmit power setting for the spacecraft when sufficient RSSI data is available. In order to set the transmit power, when a spacecraft sends an acknowledgement back to a transmitting spacecraft it includes the corresponding RSSI reading. Using this reading and Equation (46) which as has 20 dB *Fade Margin* built into it, the new transmit power can be set. Since a different power may be necessary for each receiving spacecraft, the greatest of the minimum required power settings is utilized when transmitting the broadcast ephemeris update message; if a one-to-one message for sensor data sharing is required than the individually corresponding transmit power setting can be utilized.

$$\begin{aligned} \text{Tx Power} &= \text{RSSI} - \text{Tx Antenna Gain} + \text{Tx Cable/Connector Losses} \\ &+ \text{Path Loss} - \text{Rx Antenna Gain} + \text{Rx Cable/Connector Losses} + 20 \end{aligned} \quad (46)$$

4.3 Summary Send and Receive Communications Processes

On top of the selected protocol itself, several process have been described in the preceding sections. For clarity, the algorithms or processes used for handling inter-satellite communication is provided in the following steps.

1. The collaborative `main` function delivers a message to the `transmit` function. Data to transmit, message type, and, if applicable, the originating subsystem and destination IDs are provided to the function.

2. A message wrapper (see section 4.1.1) is constructed.

   (a) If a subsystem ID is provided, it is passed into the preface, otherwise the spacecraft ID is set as the *Originator ID*.

   (b) If a destination ID is provided, it is passed into the preface, otherwise 0 indicating *one-to-all* is set to the *Destination ID*.

   (c) The incrementing *Message ID* is set (and incremented).

   (d) The provided *Message Type* is set; recall that these are defined by a mission.

   (e) A *Timestamp* is added.

   (f) The corresponding `checksum` function is called on the provided data and set in the preface.

3. The data is appended to the message wrapper.

4. Using previously calculated values or the RSSI acknowledgement message feedback, the *MHX*-2420 transmit power is set accordingly.

5. The collaborative module `transmit` function sends the entire message to the *MHX*-2420 via a wired serial link for transmission.

6. A timeout period of 15 seconds begins while waiting for message acknowledgements from all parties the message was intended for (depends on what the *Destination ID* was set to).
    (a) If the timeout expires before receiving acknowledgements, the message is re-transmitted. For the ephemeris/state message update, this is only done once. For sensor data sharing, the number of attempts can be modified.
7. With the received acknowledgements the `transmit` function updates the transmit power settings for each respective team member.

The corresponding task on the receiver end follows the process below.

1. A message is received from and placed in the buffer by the *MHX-2420*.
2. The collaborative module `receive` function reads the message over serial and pulls it out of the modem buffer. If utilized, the corresponding RSSI is recorded.
3. The *Destination ID* of the message is checked to ensure it is intended for itself or one of its subsystems.
    (a) If the message is not intended for this spacecraft, the message is discarded and the following actions do not take place.
4. The `receive` function checks the *Originating ID* and ensures that the *Message ID* is unique.
    (a) If the *Message ID* is not unique, the message is discarded since action has already been taken and the following actions do not take place.
5. The `receive` function ensures that the *Timestamp* for the given *Message Type* is the most recent.
    (a) If the *Timestamp* for the given *Message Type* is not the most recent the message is discarded and the following actions do not take place.
6. The `receive` function calls the corresponding `check sum` function on the data and compares it to the *Check Sum* in the message preface.

(a) If the calculated and provided *Check Sum* do not equate, the message is discarded since the integrity of the received data is in question. None of the following tasks are completed.

7. The `receive` function delivers the data to the appropriate subsystem or other function (such as to update the state tensor for orbit propagation).

8. Receiver spacecrafy sends an acknowledgement with the *Destination ID* set to the sender's ID; optionally the recorded RSSI value is attached.

Each of the individual components of the collaborative module have been discussed in detail: orbit propagation in Chapter 2, on-board orbit determination in Chapter 3, and inter-satellite communication in this chapter. With these completed pieces an over-arching algorithm for maintaining shared knowledge of state among a CubeSat team can now be implemented. Chapter 5 follows with details on algorithm as well as the simulation results and analysis.

CHAPTER V

FINAL ALGORITHM, SIMULATIONS, AND RESULTS

Each component needed for maintaining spacecraft knowledge among a team of CubeSats has been discussed in the preceding chapters. The details on the algorithm for checking the error tolerance and issuing a broadcast update is first discussed. Various types of simulations are completed, some including hardware-in-the-loop. For each simulation discussion of the setup is provided as well as analysis of the results.

5.1 Maintaining Knowledge of Spacecraft Team

Chapter 2 provides extensive details on the various force models, propagation techniques, and integration routines explored for the purpose of propagating a spacecraft's self (to be used along with GPS measurements in the extended Kalman filter for orbit determination from Chapter 3) as well as the rest of the team. From the error analysis results four different integration routines with specific time steps were initially identified: RK4 with 45 and 90 second time steps, RK6 with a 45 second time step, and for *MatLab* simulations the variable-step `ode113` integrator. After implementing the extended Kalman filter, it was determined that GPS measurements every 15 seconds provide the best results for convergence and error reduction. Since sensor measurements are processed every 15 seconds, it is best to have a model that provides corresponding state predictions. With the reduction in the timestep, use of the common yet simple RK4 integrator is selected. For the small timestep, the higher order RK6 does not provide sufficient error improvement for consideration and the variable step `ode113` provided by *MatLab* initiates with very small timesteps so it requires far more derivative calls than the four required by the fixed-step method in order to get the 15 second propagation step. Lastly, it was determined that integrating in Cartesian coordinates (specifically the Mean of J2000 Earth Ecliptic frame) is preferable for the simplicity of mapping measurements to states.

With the orbit propagation alone it cannot be expected for each satellite to maintain accurate

knowledge of the rest of team. One-to-many broadcast updates are necessary to capture the perturbation effects not represented in the force model. Within this broadcast a satellite will provide an update on its current Cartesian state at a specific time plus six estimated model parameters; refer back to Chapter 4 for details. The rest of the spacecraft in the team will then use these updated state values and model parameters to use for prediction of team members' position and velocity.

5.1.1 Update Algorithm

Maintaining shared knowledge state among the team demands defining a process for a spacecraft to determine when a broadcast update is necessary. A relatively simple tolerance check algorithm has been developed; the process is defined in the following steps.

1. Initialize tensor of state vectors by each spacecraft conducting a one-to-all broadcast that contains a raw GPS measurement at a defined system-wide epoch.

2. Each spacecraft begins receiving GPS measurements every 15 seconds from an on-board sensor. Concurrently, the command and data handling subsystem begins propagating the spacecraft team maintaining states with time steps of 15 seconds. Two options are available for propagation: individual direct Cartesian or using a state transition matrix and initial perturbations to calculate a linearized prediction of state for each spacecraft. Note that for the second option a CubeSat still maintains a straight propagation of its state from its last broadcast that is not influenced by the EKF.

3. After receiving a GPS measurement, the extended Kalman filter (discussed in Chapter 3) processes the data while using the spacecraft's own propagated state in order to estimate its actual position.

4. The estimated state is compared with the propagated state and checked against a mission-specific tolerance.

   (a) If the tolerance is met, no broadcast is required and the satellite continues with any previously defined mission tasks.

   (b) If the tolerance is not met, the spacecraft conducts a one-to-all broadcast to update its state plus six parameters for all team members.

i. CubeSat uses its *MHX*-2420 S-Band modem to send message. As few as one broadcast is required regardless of team size.

ii. CubeSats waits 15 seconds (or mission-specific timeout period) to receive an acknowledgement from each spacecraft team member.

A. If an acknowledgement is not received from each teammate the identical message is re-transmitted.

B. Regardless of whether acknowledgements are received from the second message or not, the CubeSat does not continue trying to transmit the update. If a specific teammate is not "heard" from for an entire day (this tolerance can be modified depending on mission requirements), the spacecraft assumes the teammate is no longer part of the team and ceases checking for acknowledgements.

5. Go to 3.

In the following sections details and analysis on various simulations is provided. The above defined process is implemented, with the exclusion the additional acknowledgement which goes above and beyond the protocol's own process.

5.2 MatLab *Sequential Shared-Memory Simulation using Simple Propagation*

5.2.1 Simulation Setup

Before integrating the actual S-Band communications hardware, or even a cable serial link, message passing can be simulated via a globally shared memory framework. In order to demonstrate the advantage of using the selected force model and extended Kalman filter, the message count and related time between broadcast messages is obtained using this type of setup. In this first simulation each CubeSat propagates every other spacecraft on the team using the standard Cartesian integration. This process allows for spacecraft *A* to know exactly where spacecraft *B* thinks spacecraft *A* is; using such an implementation is not scalable on a single core therefore the state transition matrix approach is provided in the following section. To begin this simulation process, each CubeSat is given its own unique initial state vector; the six model parameters, model noise matrix, and covariance matrix are

identical for all spacecraft. In a sequential process, each individual CubeSat then completes the following tasks.

1. A CubeSat checks to see if it has received a message from another spacecraft. If so, it updates its own set of spacecraft team member states including the model parameters.

2. For each spacecraft in the team, including itself, it propagates the states forward 15 seconds in simulation time. The state obtained using direct propagation for oneself is kept separate and is not included in the filter.

3. A spacecraft uses the current state estimate (from any previous filtering) and propagates it forward using the model and associated parameters 15 seconds in simulation time. Additionally, the state transition matrix is integrated alongside for use in the EKF.

4. Using the propagation prediction and a new GPS measurement, the EKF algorithm discussed in section 3.2.1 provides an updated estimate of state including the six model parameters.

5. Compare the EKF estimate with the straight propagation. This is equivalent to comparing the best state estimate with the state that each teammate currently holds.

   (a) If the difference is greater than a mission-specific tolerance then a "message" is broadcasted with the updated state. In the *MatLab* implementation this is completed by setting a Boolean value in a `messages` matrix from 0 to 1. This matrix is checked in Step 1 for the previous timestep to see if a "message" was sent.

   (b) If the difference is within the tolerance, the new estimate and covariance matrix are saved and the simulation progresses onto the next CubeSat.

6. Once all CubeSats in a team have completed one time step increment, the algorithm returns to Step 1 and continues with the next time step.

5.2.2 Analysis

Increasing the time between the one-to-many broadcasts is the goal of this research; the main driving factor is that the power required for an RF transmission is far greater than computation on a

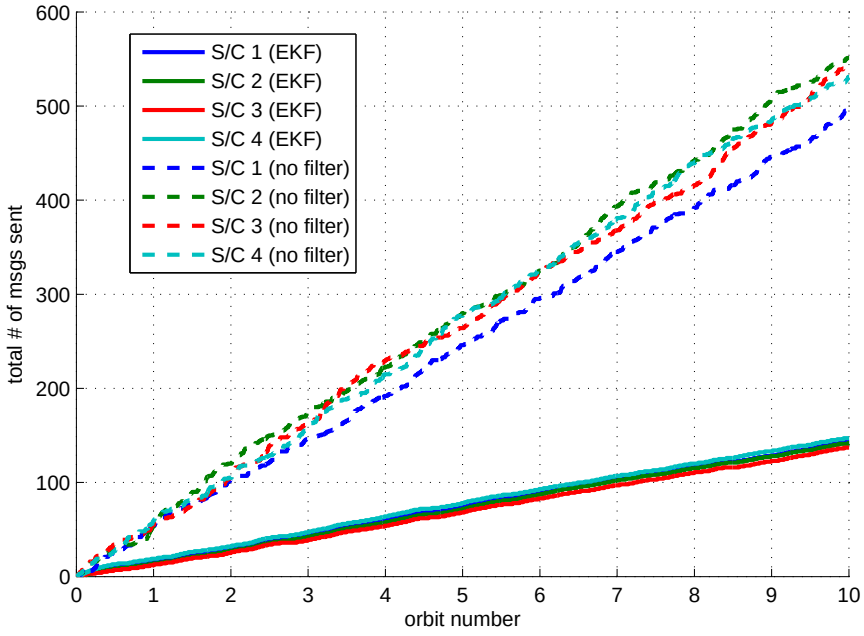

Figure 43: With a GPS receiver that provides position and velocity measurements with accuracies of 10 m and 30 $^{cm}/_{s}$, respectively, use of the EKF increases the time between messages from 1.87 to 6.86 minutes when a 10 *m* knowledge tolerance is required.

low-power micro-controller such as those utilized on CubeSats. To demonstrate the algorithm's effectiveness in reducing the amount of communication, simulations of 10 orbits with varied GPS receiver accuracy, knowledge tolerance, and number of spacecraft are conducted.

Analyzing the set of simulations, the value of the on-board propagation and state estimation (versus not including these collaboration module components at all) is highly dependent on the relationship between mission-specific tolerance and the GPS receiver accuracy. This is made evident in the following discussion.

In Figure 43, GPS receiver accuracies of 10 m for position and 30 $^{cm}/_{s}$ for velocity are used to generate sensor data. A knowledge tolerance of 10 m was selected. It is clear that the two groups, EKF versus no filter, display similar results between companion CubeSats. Over the course of 10 orbits each spacecraft using the filter provides a broadcast update every 6.86 minutes on average while the spacecraft not using a filter sent a message every 1.87 minutes, which is approximately once every 7 or 8 GPS measurements.

If the tolerance is significantly loosened, however, then the filter provides little no advantage.

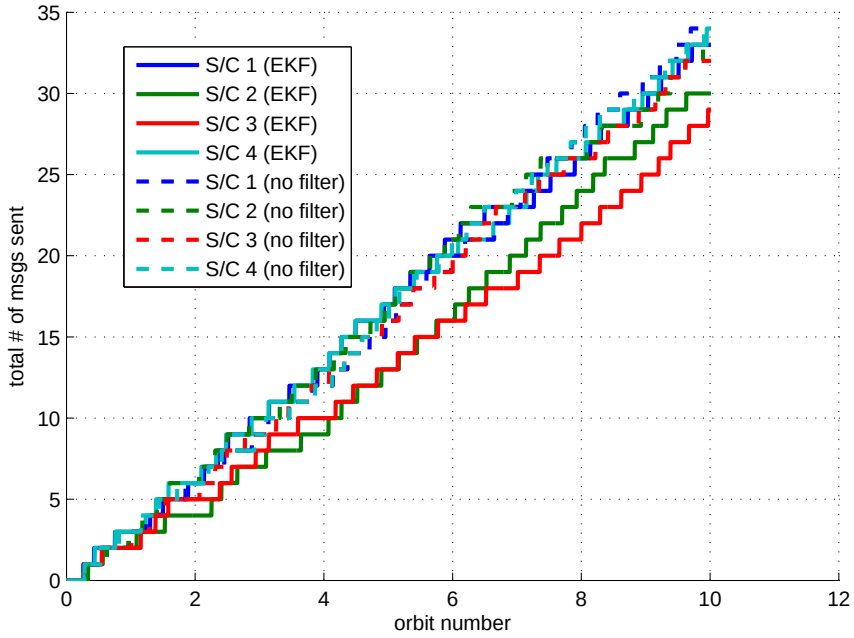

Figure 44: Using a GPS receiver that provides position and velocity measurements with accuracies of 10 m and 30 $^{cm}/_{s}$, respectively, satellites with the EKF broadcast every 29.59 minutes compared to 28.42 minutes with no filter when a 100 m knowledge tolerance is required.

Figure 44's example actually shows only a slightly higher average time between messages when the EKF is used versus no implemented filter. Figure 44 uses the same GPS accuracy values as Figure 43 and only changes the tolerance from 10 m up to 100 m. Although not displayed, the trend continues with low tolerances on the order of 1 km.

By examining Figure 45 it can be noted that if the GPS receiver provides reduced accuracy, for this example it was 100 m and 1 $^{m}/_{s}$, the filter once again can provide improvement. This example displays a 13x improvement when the EKF is utilized and less than 5 messages per orbit are necessary per spacecraft.

Not all GPS receivers provide velocity measurements, therefore Figure 46 displays results from a simulation similar to that used for Figure 43, except only the 10 m accuracy position data is used in the EKF. Without a receiver that provides a velocity measurement, it can be argued that the model filter is necessary regardless since an update message cannot provide the velocity portion of the state. Figure 46 is actually comparing one scenario in which an EKF using position data only versus a scenario where the GPS receiver provides both position and velocity data but is not filtered.

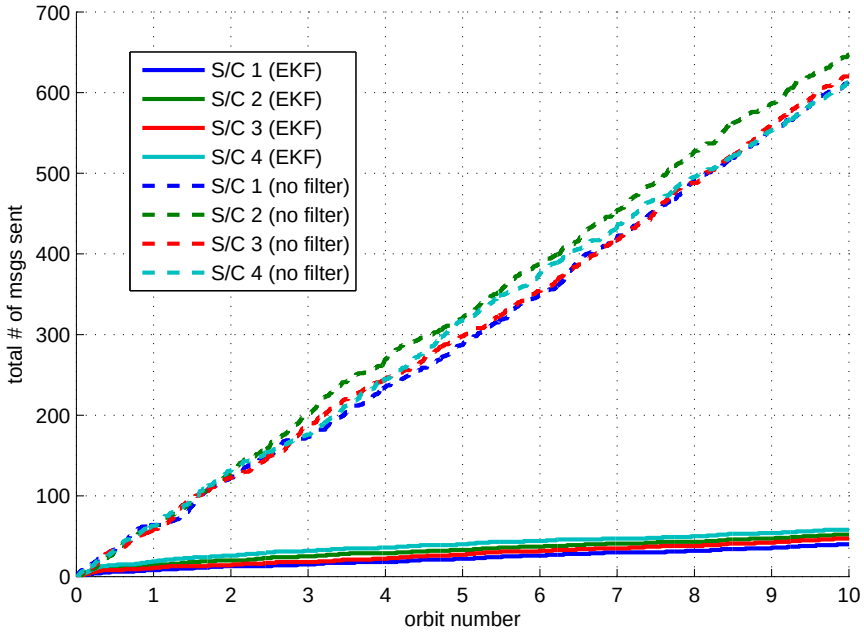

Figure 45: Using a GPS receiver that provides position and velocity measurements with accuracies of 100 m and 1 $^{m}/_{s}$, respectively, use of the EKF increases the time between messages from 1.59 to 20.68 minutes when a 100 *m* knowledge tolerance is required.

By using the filtered estimates based on position measurements alone, the time between messages increases from an average of 1.52 to 4.88 minutes.

A final comparison with a low accuracy GPS receiver and a relatively tight tolerance is provided in Figure 47. Here the sensor data was generated using 100 m and 1 $^{m}/_{s}$ accuracies for position and velocity; the knowledge tolerance was set to 20 m. Since the knowledge requirement is a fifth of the sensor accuracy, a message is required just about every time a new GPS measurements is available if no filter is used; with an implemented EKF the time between messages is just under 4 minutes for an approximate 15x improvement.

5.3 MatLab *Sequential Shared-Memory Simulation using STM Propagation*

5.3.1 Simulation Setup

The final on-board implementation, depending on mission-specific needs, may use the STM in order to calculate a neighboring spacecraft's state rather than directly propagating each teammate. Using the STM for a formation with separations on the order of kilometers or less is reasonable, even considering the linearizations involved, however the disadvantage is that spacecraft *A* does

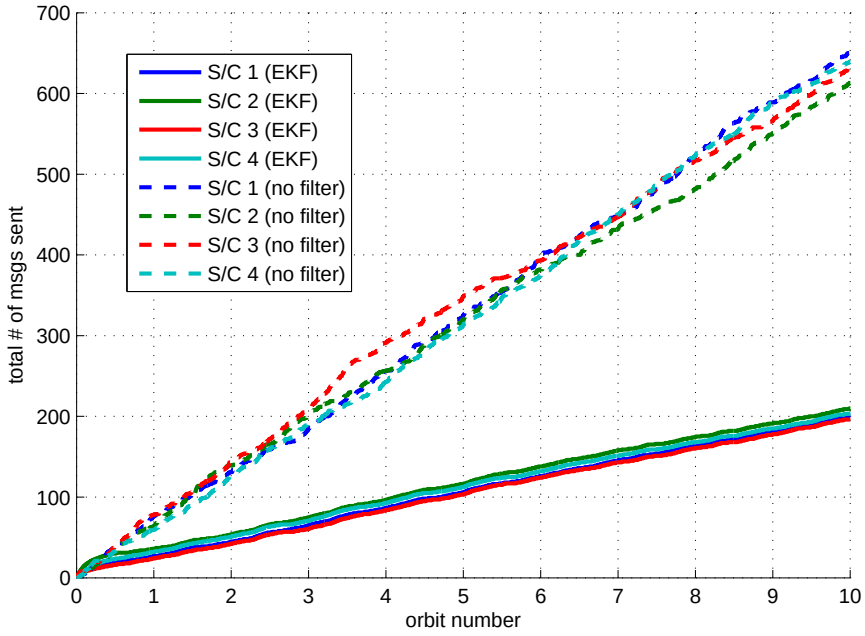

Figure 46: Using a GPS receiver that provides position measurements only with accuracy of 10 m, the EKF increases the time between messages from 1.52 to 4.88 minutes when a 10 m knowledge tolerance is required.

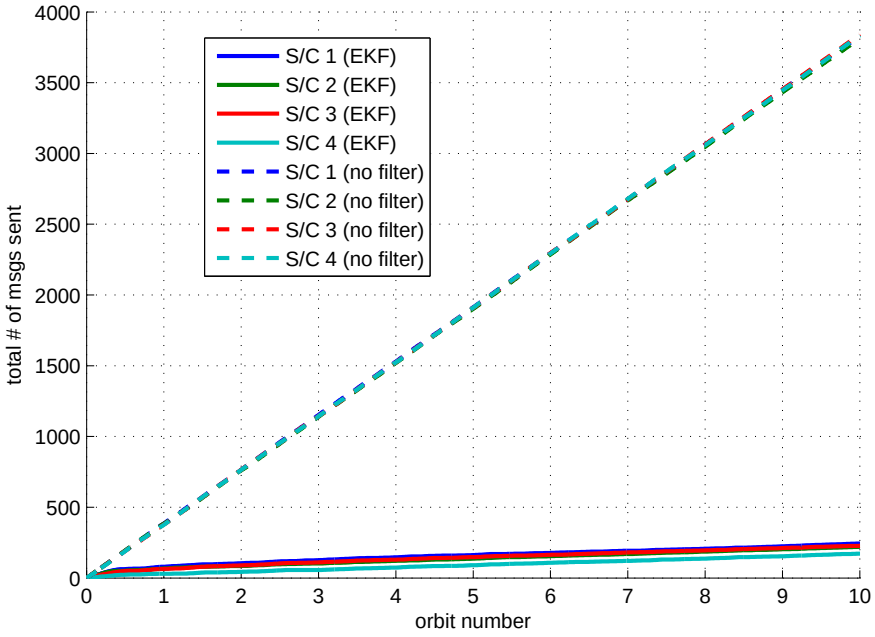

Figure 47: Using a GPS receiver that provides position and velocity measurements with accuracies of 100 m and 1 $^{m}/_{s}$, respectively, use of the EKF increases the time between messages from 0.26 (every time a GPS measurement is available) to 3.81 minutes when a 20 m knowledge tolerance is required.

not know exactly where spacecraft B "thinks" spacecraft A is located. Therefore, even though a tolerance of 10 m may be selected the linearization from using the STM could result in the actual error in spacecraft B's prediction of spacecraft A's position being greater than the tolerance. The goal of this simulation is to determine the impact of using the STM rather than direct Cartesian propagation. A slight modification to the process from section 5.2 is required; this simulation's process is outline below.

1. A spacecraft checks to see if it has received a message from another spacecraft. If so, it updates its own set of spacecraft team member states including the model parameters. Note that for the STM implementation it is $d\vec{x}_0$ rather than $\vec{x}_0$ which is being stored. The broadcast message, however, is not modified.

2. The spacecraft propagates itself forward in time 15 seconds. Recall that the state obtained using direct propagation is kept separate and is not included in the filter.

3. A spacecraft uses the current state estimate (from any previous filtering) and propagates it forward using the model and associated parameters 15 seconds in simulation time. Additionally, the state transition matrix is integrated alongside for use in the EKF and for calculating the location of teammates.

4. The CubeSat uses the STMs from the previous steps and most current $d\vec{x}_0$ vectors to predict the location of each spacecraft in the team. Recall that the product of adjacent STMs is equivalent to an STM propagated from the earliest point in time to the latest given the bounds of the STMs multiplied together [66].

5. Using the propagation prediction from the filter and a new GPS measurement the EKF algorithm (discussed in section 3.2.1) provides an updated estimate of state including the six model parameters.

6. Compare the EKF estimate with the straight propagation. This is **not** exactly equivalent to comparing the actual state with the state that each teammate is currently using, but is a close approximation at a fraction of the computational cost.

Table 9: The first four sets of initial conditions are utilized for the first set of plots; due to the close proximity of S/C 1 and S/C 5, this pair is used for the second piece of analysis within this section. Note that all five spacecraft share the same initial velocity.

S/C ID	x_0 [km]	y_0 [km]	z_0 [km]
S/C 1	$3.849641578128 \times 10^{+3}$	$-5.965756060445 \times 10^{+3}$	-3.787771949
S/C 2	$3.851703094281 \times 10^{+3}$	$-5.978608875901 \times 10^{+3}$	8.23671817
S/C 3	$3.855605290087 \times 10^{+3}$	$-5.971266112583 \times 10^{+3}$	-15.018258914
S/C 4	$3.844619757199 \times 10^{+3}$	$-5.966861545635 \times 10^{+3}$	3.585608637
S/C 5	$3.850289507479 \times 10^{+3}$	$-5.967283420321 \times 10^{+3}$	-2.751854204
S/C ID	u_0 [$^{km}/_{sec}$]	v_0 [$^{km}/_{sec}$]	w_0 [$^{km}/_{sec}$]
All	-0.898937943	-0.584783786	7.415581285

(a) If the difference is greater than a mission-specific tolerance then a "message" is broadcasted with the updated state. In the *MatLab* implementation this is completed by setting a Boolean value in a `messages` matrix from 0 to 1. This matrix is checked in Step 1 for the previous timestep to see if a "message" needs to be received..

(b) If the difference is within the tolerance, the new estimate, covariance matrix, and incremental STM are saved and the simulation progresses onto the next spacecraft.

7. Once all spacecraft in a team have completed one timestep increment, the algorithm returns to Step 1 and continues with the next timestep.

5.3.2 Analysis

The previous section's simulation chose an arbitrary set of four spacecraft initial conditions. It was not necessary at first to provide the initial conditions, but with the introduction of the STM, the relative spacing becomes important. Table 9 provides the initial conditions for the five satellites used in this analysis section.

This simulation is setup to answer the question of how large is the valid domain region when using an STM for calculating teammate states. First, however, it is a good check to ensure that the same number of messages are sent by spacecraft to provide updates. The same number of messages as the previous section is expected since the process for checking the error tolerance is identical.

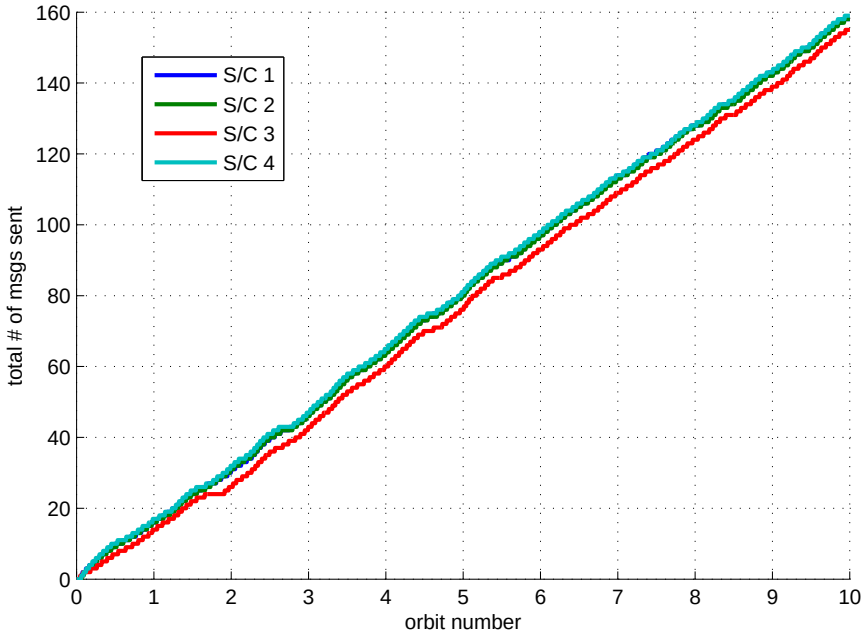

Figure 48: As expected, approximately the same number of ephemeris update messages are broadcast by each satellite when using the STM to determine teammate positions since the tolerance check process is the same. The same tolerances and sensor accuracies from Figure 43 are utilized.

A quick examination of Figure 48 shows that approximately the same number of messages per spacecraft are transmitted as in Figure 43. The small variation is a result of slight difference in noise added to the GPS measurements.

Figures 49 and 50 each provide two separate pieces of information. For both figures, the upper plot displays the error between the position prediction S/C 1 holds for S/C 2 with the desired tolerance plotted as the dotted red line. The lower plot provides the relative distance between S/C 1 and S/C 2. With these two pieces of information, it is easily explained why S/C 1's position prediction of S/C 2 is on the order or tens or even hundreds of kilometers after five orbits; even though the two spacecraft start out within 20 km of each other they quickly separate to over 1000 km in as little as five orbits. Separation distances of this size fall outside the valid domain of the STM linearization.

If only the first two orbits of Figure 49 are examined, as is displayed in Figure 50, the error is significantly reduced since the spacecraft separation distance is at a maximum of 450 km. To further explore the region of STM validity two CubeSats with closer initial conditions are selected; specifically S/C 1's and S/C 5's initial conditions from Table 9 are utilized.

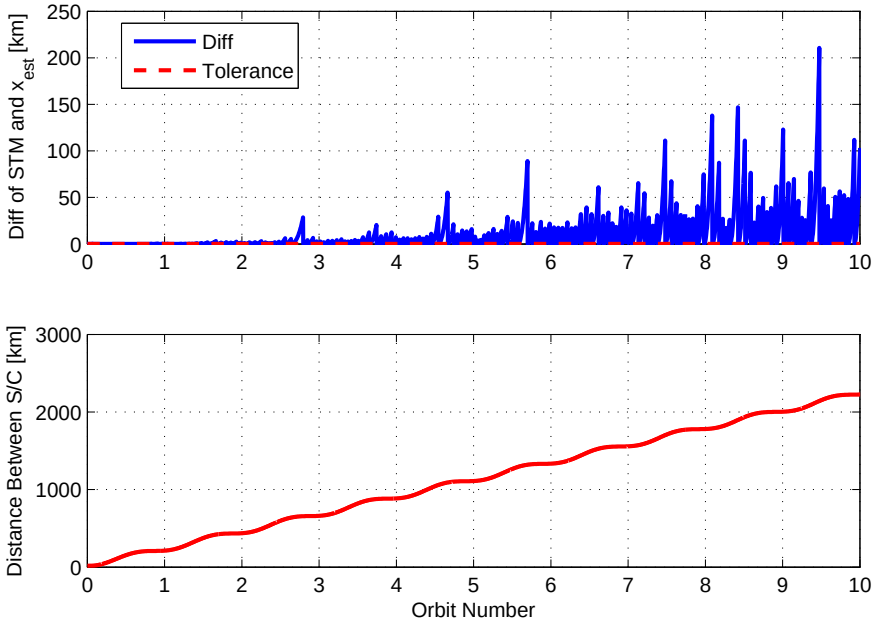

Figure 49: Use of an STM is limited to systems where spacecraft remain within a couple hundred kilometers or less of another. This figure depicts S/C 1's prediction of S/C 2's position is up to 200 km off even when a tolerance of 10 m is selected.

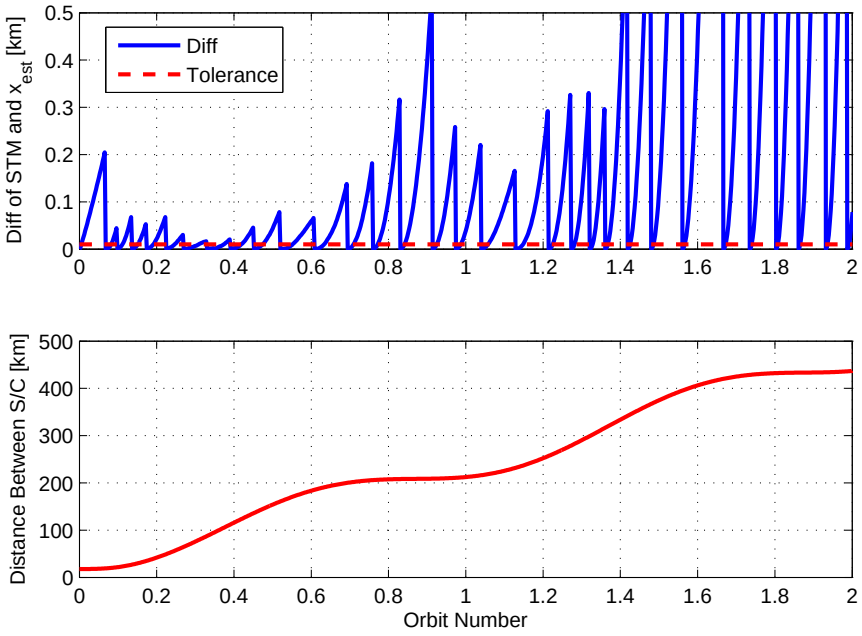

Figure 50: Examining only the first two orbits from Figure 49 where S/C 1 and S/C 2 remain within 450 km of each other the STM provides a more reasonable position prediction, although still up to 50x the selected tolerance.

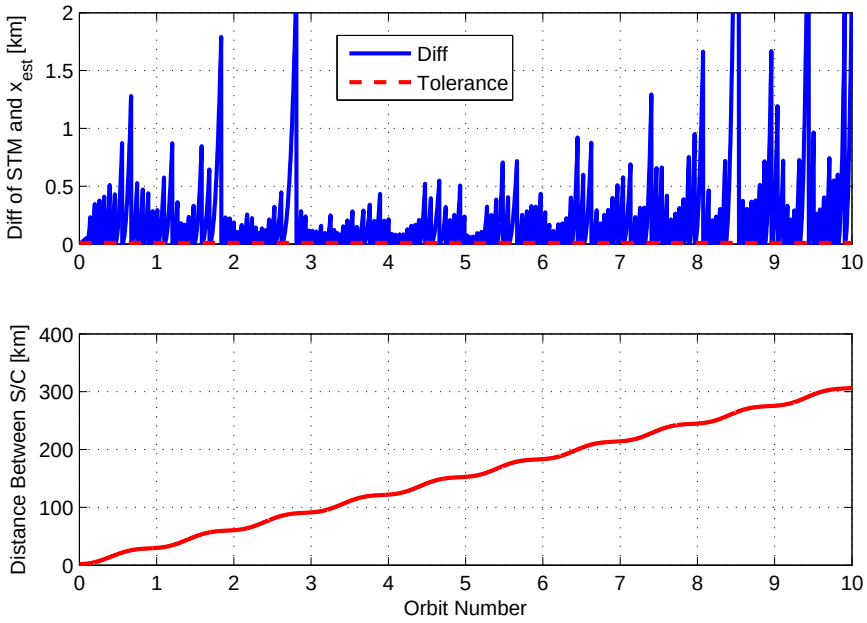

Figure 51: S/C 1 and S/C 5 from Table 9 maintain close proximity over ten orbits providing improved, yet still lacking, performance for missions require tight tolerances.

Figures 51 and 52 provide corresponding plots data as Figures 49 and 50 except S/C 1 is now tracking S/C 5. First, notice that over the course of ten orbits the two spacecraft only separate to about 300 *km*. Although the STM is still not meeting the 10 m tolerance, it does maintain errors usually on the order of hundreds of meters with spikes on the order of kilometers. Depending on a mission's tolerance needs, and assuming the CubeSats remain in close proximity, the STM shows here that it could provide sufficient prediction accuracy while significantly saving on computational cost. However, if a mission desires to maintain state knowledge on the order or meters or 10s of meters, the separate state propagation may be required.

5.4 MatLab *Simulation with S-Band Transceiver*

5.4.1 Simulation Setup

The first extension to include COTS hardware is to tie the S-Band transceivers into the *MatLab* simulations. In the first two simulations discussed a global `messages` matrix is used as a means to "communicate" between spacecraft. Although this simulation will still be conducted in a near sequential manner, it is completed to demonstrate the working communications framework discussed

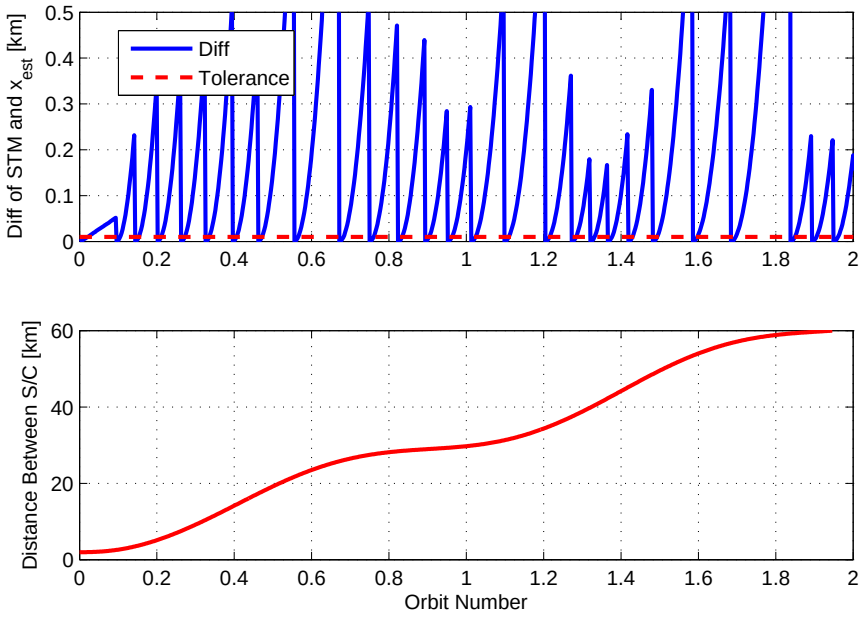

Figure 52: Closely examining Figure 51 over only two orbits shows that even with spacecraft being tens of kilometers apart, the errors from the STM prediction are on the order of hundreds of meters.

in Chapter 4. Instead of simply setting a binary marker in the `messages` matrix, two separate instances of *MatLab* execute concurrently and communicate with one another over S-Band. The two instances run on the same machine, but could also be run on separate machines; no other changes would be necessary since each instance has its own dedicated serial I/O port. In order to ensure that each instance progresses at the same simulation time rate additional communications are necessary. Specifically, rather than broadcasting only when an ephemeris update is required, a `null` message with the current simulation time is transmitted after each GPS measurement is processed. It is this need for communication that results in a near-sequential execution of the simulation; although the two spacecraft can and do execute in parallel, they often are held-up by the blocking receives waiting for the other "CubeSat" to send a message.

5.4.2 Analysis

Examining the message count and prediction error plots as seen before ensures that when actual flight hardware is tied to the simulation the same results are obtained. Such a hardware-in-the-loop simulation provides confidence that future integration of all software components on a CubeSat

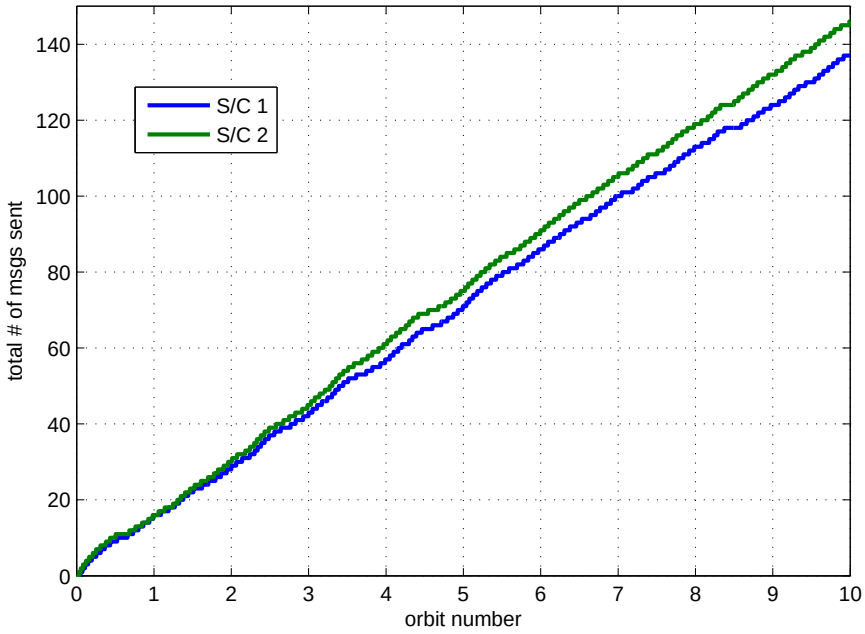

Figure 53: When conducting hardware-in-the-loop simulations with the *MHX-2420* transceivers similar results are obtained, indicating the software will integrate well in a final CubeSat bus implementation.

command and data handling micro-controller subsystem will operate properly.

For verification Figures 53 and 54 provide the message count and prediction error plots, respectively. The message count plot provides results very similar to Figure 43 which was the *MatLab* sequential implementation with global variable sharing. This is expected since the only modification from the first simulation set was replacing the global `messages` matrix with serial I/O to the S-Band modems and the modem-to-modem RF link. Additionally, Figure 54 shows the error between the globally shared prediction estimates for S/C 1 and 2 with the tolerance of 10 m displayed as the red dotted line. Note that the prediction error is the difference between the current prediction and the "truth" provided from *FreeFler*; if the difference between the current prediction and estimate from the EKF were plotted than all error would be bounded by 10 m. This plot provides confidence that over 80% of the time the absolute difference is within the set tolerance and more than 99% of the time the difference between predicted and "actual" is within twice the selected tolerance.

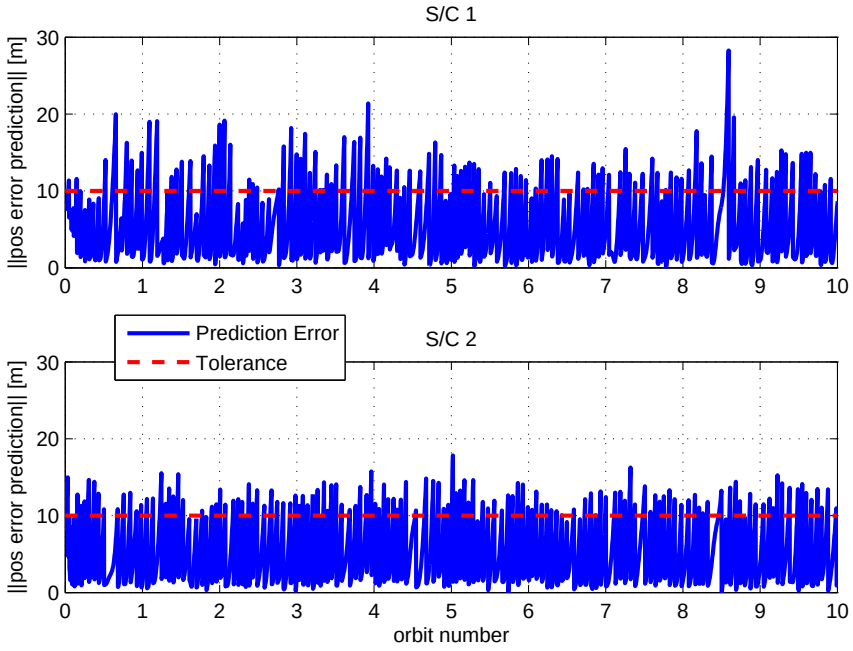

Figure 54: Direct propagation of each spacecraft, rather than using the STM, can almost ensure that the error between "truth" and a team members predication are within two times the set tolerances; over 80% of the time the tolerance itself is met.

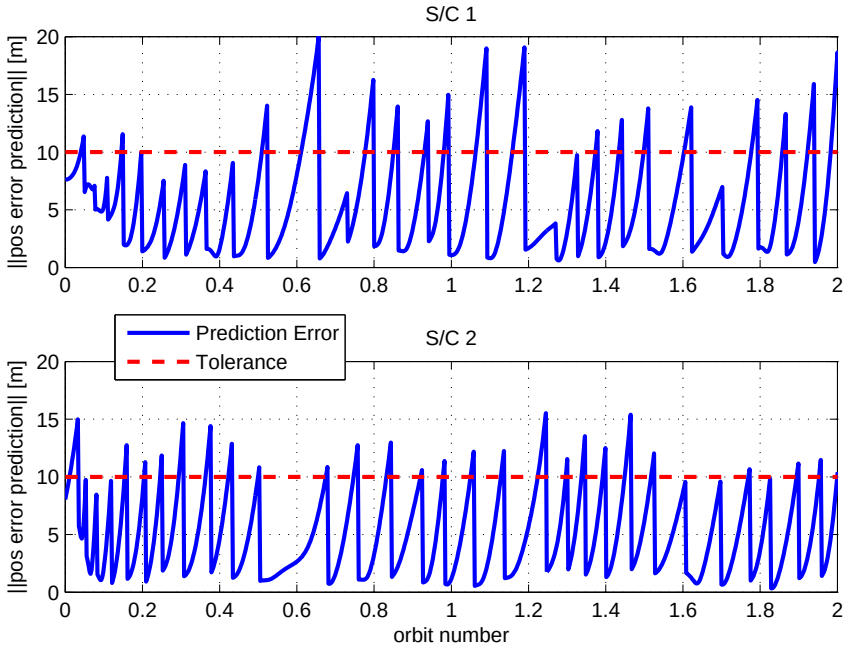

Figure 55: Examining Figure 54 closely shows sinusoidal-like behavior of the prediction error drifting away from zero and then return to only 1 or 2 m error when a new EKF estimate of state is broadcasted.

5.5 OpenMP Shared Memory Parallel Propagation-Only Simulation

5.5.1 Simulation Setup

Scalability of the collaborative software is an important aspect to test to ensure it is capable of handling formations of an arbitrary size. In section 5.3 it was determined that depending on the mission tolerance required, using an STM may not be capable of providing sufficient accuracy. If an STM can be utilized then the amount of computation necessary minimally increases since only matrix multiplication, rather than numerical integration, is required. If, however, tighter tolerances near 10 m are desired, individually propagating each spacecraft may be required; in this scenario utilizing multiple computation cores may be advantageous. By using a multi-core chipset the same performance (execution time) as a powerful single-core could be provided while reducing total power usage through core throttling. Core throttling is the reducing of the supply voltage to a digital processor which in turn results in a slower clock speed. The power savings comes from the fact that the power required to operate a microprocessor is related to frequency raised to the second to third power [41]. In other words, if the frequency is doubled, then the power requirement four to eight times the baseline power. Inversely, if the frequency can be halved, then the required power could be reduced by as much as 70 or 80%. Therefore, determining if the collaborative algorithms are parallelizable assists in estimating the power savings achievable by using multiple cores at lower frequencies.

Before providing the analysis, a brief description of the hardware used for parallelization testing is in order. Although the long-term goal is to demonstrate the feasibility on multi-core micro-controllers, the hardware utilized for is a 6-node server. The Hogwarts cluster, available to Georgia Institute of Technology Computational Science and Engineering students, allows for execution of jobs on up to 5 nodes utilizing all 8 cores per node. Each node houses 2 quad-core Intell' Xeon X5570 processors with 8 MB of cache per processor and both Hyperthreading and Turbo Boost technology enabled. Forty-eight GB of RAM are available on each node as well as ample hard disk storage. The 6 nodes are connected via a gigabit Ethernet.

5.5.2 Analysis

Two different types of scaling need to be examined: strong and weak scaling. In strong scaling the problem size is maintained while the number of cores increases; "perfect" strong scaling occurs when the execution time for a program requires $\frac{t_{seq}}{p}$ where t_{seq} is the time necessary for sequential execution and p is the number of cores. Weak scaling modifies the process by increasing the problem size and number of cores at the same time. For example, if the CubeSat team increases from 2 satellites to 4 satellites then the number of cores increases from 1 to 2. "Perfect" weak scaling occurs when the time to execute a program requires the exact same amount of time as t_{seq}. Results for each type of scaling are provided in the following two sections.

5.5.2.1 Strong Scaling

Strong scaling tests chose a problem size of 16 CubeSats each propagated for 3456000 seconds; such a long propagation time was selected in order to have a long enough period to measure the time differential. The number of cores was varied from one to 16. Examining Figure 56 the algorithm scales nicely up to about four cores, but then plateaus except for a jump at 8 and 16 cores. The plateaus are a function of the hardware architecture and algorithm combined. For four cores the problem is nicely divided such that each quad-core die utilizes two cores and each core propagates four satellites. For each of the quad-core chips a total of 8 MB of cache is shared among them, so when only two threads are executing on a single processor the cache sharing works relatively well. When a fifth core is added now one quad-core chip is using three cores, and it appears that there is insufficient cache to keep all cores busy at the same time due to memory I/O. The work is no longer distributed evenly due to how the parallelization in the code is completed. Division of threads such that each thread handles a single CubeSat's propagation is at first the most apparent, and was therefore implemented. When the work cannot be distributed evenly, there is a "tail" of execution that must be completed by the busiest core. This behavior explains the jumps in speed-up at both 8 and 16 cores, since the work transitions from a plateau of unevenly distributed labor to a problem size that allows for equal work among all cores.

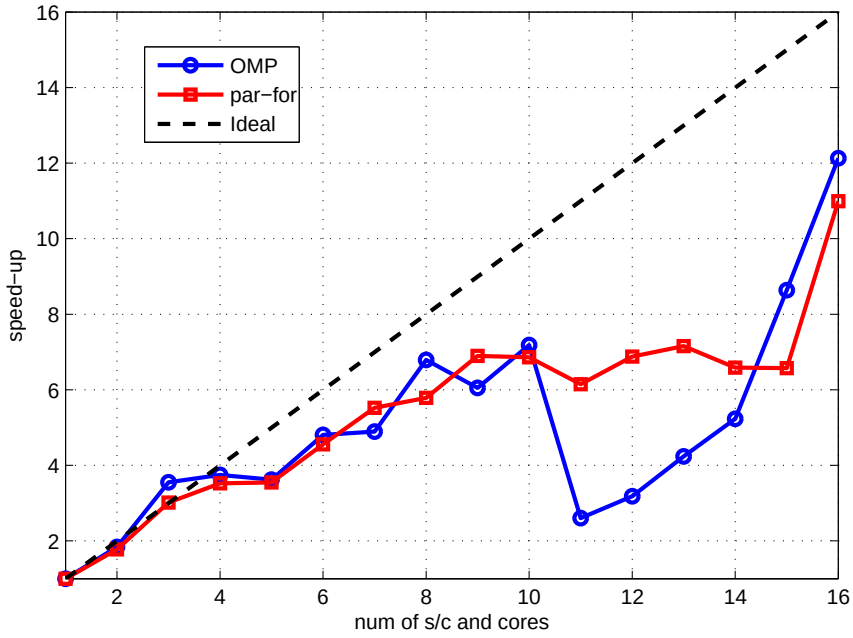

Figure 56: Up to 8 cores the propagation algorithm scales nicely, but then plateaus until 16 threads are utilized. The plateau is a function of the architecture and algorithm itself, due to cache sharing and the parallelization being large blocks of code.

5.5.2.2 Weak Scaling

Weak scaling is conducted by matching the number of spacecraft with the number of cores. With the exception of the anomalous OMP data point at 7 cores, the two different implementations (OMP vs. OMP `parfor`) show similar performance. Using `parfor` simplifies the code slightly since the distribution of remaining work when it cannot be evenly distributed is handled automatically; when using straight OMP constructs the work has to manually distributed in some manner by the code. The long-term scaling evolution shows good weak scaling since only 1.5x more execution time is required when the problem size and number of threads is 16 times greater than the baseline. As in the strong scaling, the gradual decrease in performance is most likely a function of the cache sharing and memory I/O traffic.

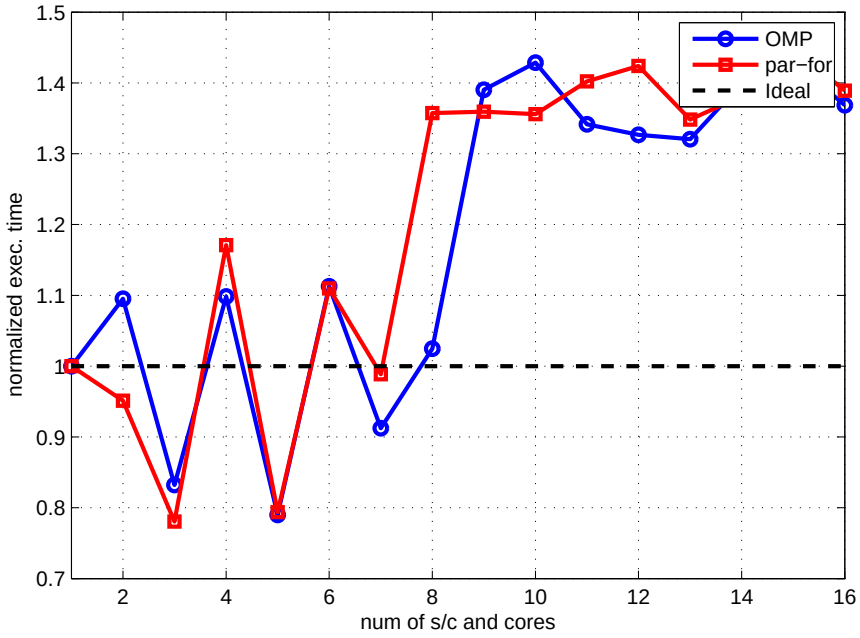

Figure 57: The long-term evolution of the weak scaling shows only a 1.5x increase in execution time, indicating that adding low-power cores as the number of spacecraft in the formation increases will provide the necessary computational power without breaking the power budget.

5.6 OpenMPI Distributed Memory Parallel Propagation-Only Simulation

5.6.1 Simulation Setup

Although rarely selected as the parallel implementation on a CMP (chip multiprocessor), Message Passing Interface (MPI) is the distributed memory counterpart to OpenMP's shared memory framework. MPI can be implemented on either a shared or distributed memory platform, but assumes each processing core has its own set of private memory. In order to share data among processes messages are passed between cores with the necessary data. Testing algorithm scalability on a distributed memory platform provides more hardware versaility when a final implementation is selected; if the algorithms scale well using message passing, then the hardware can either support shared or distribted memory and the performance will still be acceptable. These simulations ran on the same hardware described previously. Again, these simulations test the scalability of the propagation, which is nearly entirely parallel when divided by spacecraft: on-board a separate processor conducts the numerical integration of the teammates to acquire state predictions. The following sections provide the strorng and weak scaling results; only powers of 2 were tested so that the load

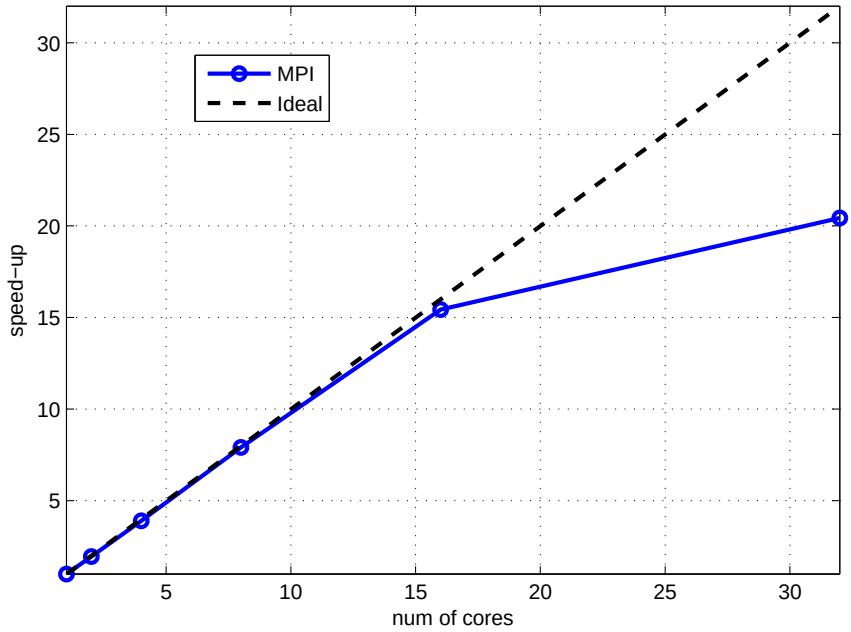

Figure 58: Strong scaling is nearly ideal up to 16 cores when each quad-core processor is only processing two spacecraft propagations. Sharing of caches, resulting in memory misses, when all four cores on the quad core are executing is the likely cause for the declined improvement from 16 to 32 cores.

on each node is evenly distributed.

5.6.2 Analysis

5.6.2.1 Strong Scaling

By distributing the processing load across multiple nodes the strong scaling speed-up exceeds the shared memory implementation. Figure 59 shows nearly perfect scaling for 1 through 16 cores. The decline in speed-up improvement between 16 and 32 cores is most likely the result of limited cache for storing the propagated states. When only 2 cores on the quad-core CMP are active there is 4 MB of cache available to each CubeSat propagation routine; halfing this appears to result in memory I/O traffic resulting in only a slight overall speed-up improvement when the number of processors are double a final time.

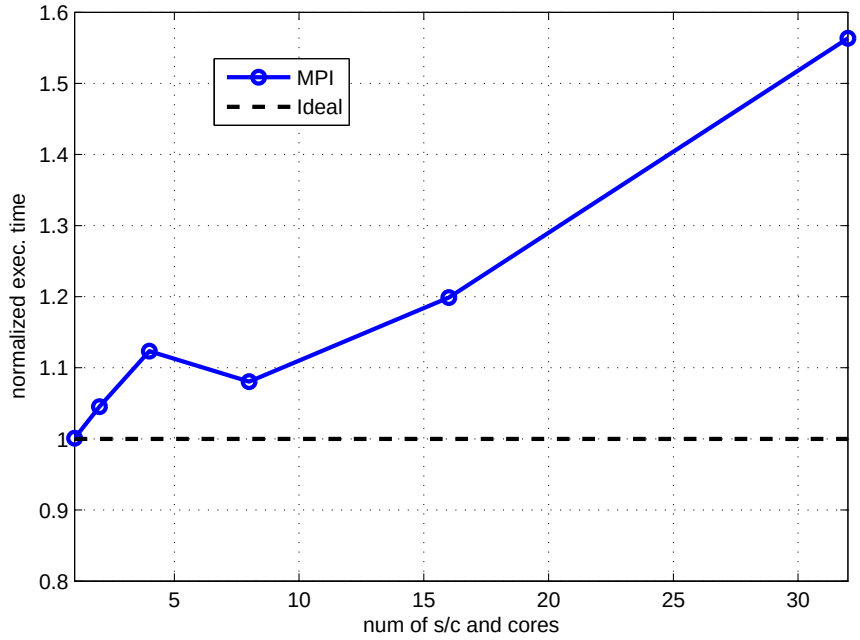

Figure 59: The long-term evolution of the weak scaling shows only a 1.5x increase in execution time, indicating that by add low-power cores as the number of spacecraft in the formation increases will provide the necessary computational power without breaking the power budget.

5.6.2.2 Weak Scaling

Like the strong scaling results, the weak scaling results for the MPI implementation (see Figure 59) show superior performance over the shared memory OpenMP results. Once again, however, the signifanct degredation in performance between 16 and 32 cores as seen in the MPI strong scaling analysis appears once again. Considering the cause (limited caches), these results are expected. Increasing the amount of fast caches or improving memory I/O throughput could allow the near ideal scaling results seen for the 1 to 16 cores.

5.7 OpenMP Shared Memory Parallel Simulation

5.7.1 Simulation Setup

The first OpenMP shared memory simulation showed that the integration of the team member spacecrafts is a scalable algorithm. In that simulation, however, an assumption was made that each satellite would be propagated for multiple orbits at one time. The actual software implementation (like the algorithm implemented in *MatLab*) conducts propagations on the order of 15 seconds followed

by reading a GPS measurement and creating an updated state estimate. Propagation of the satellites can be conducted in parallel while the sensor data processing and EKF is relatively sequential. The software written for this simulation limited the parallel integration to being just the 15 second portion followed by the sequential filter; a total simulation time of 10 orbits was completed.

5.7.2 Analysis

For the problem at hand the desire is to show that if parallel execution is available, then the wall clock time required per iteration is constant, regardless of formation size. Additional work for simulation result verification, but current results are showing promise with regards to this type of scalability. It is good to note that portions of the code executed in sequence still have room for parallelization. For example, the STM matrix multiplication has parallelization potential as does the separation of computation paths that do not contain dependencies. Refering to the preceding scaling type definitions, the end goal is successful weak scaling, but examining the strong scaling results is still good practice.

5.7.2.1 Strong Scaling

For this strong scaling implementation 32 CubeSats make-up the spacecraft team. Neglecting the use of hyperthreading, the simulation ran using 1 to 8 cores. Considering the known portion of sequential code, the results are encouraging. Figure 60 provides the strong scaling results. These results are promising since there could have been extensive overhead from splitting and merging threads every iteration (an iteration was 15 seconds of simulation time), but even when a thread is only provided 4 spacecraft to propagate one time step, the speed-up curve does not appear to level off.

5.7.2.2 Weak Scaling

Successful weak scaling is of the most interest. To assist in collecting execution times that were long enough, each core was provided two spacecraft to integrate. The results in Figure 61 resemble those from the weak scaling parallelization testing conducted on the propagation alone. As the problem size is increased 8x, adding equivalent computational power results in total execution time only increasing 1.6x.

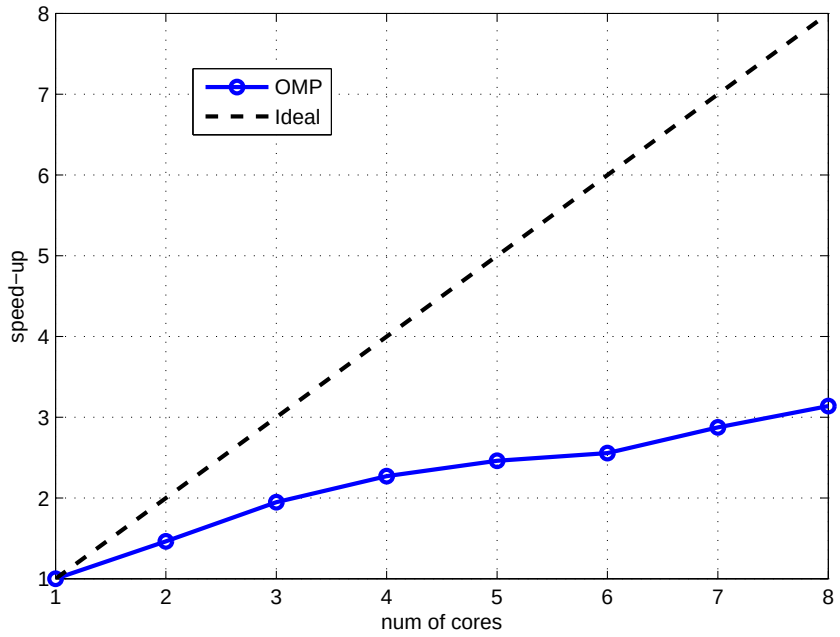

Figure 60: Strong scaling results for a team of 32 satellites is provided. As expected from the portion of sequential overhead and splitting/combining of threads for each time step iteration, the algorithm displays reduced scalability.

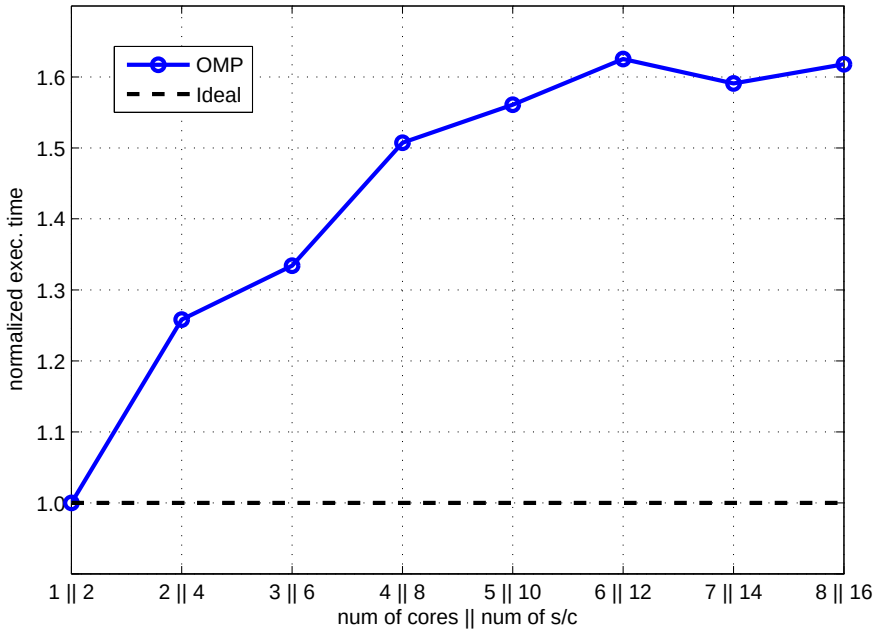

Figure 61: Even with the introduction of sequential code pieces, increasing the problem size and number of cores 8x together, the execution time is still only 1.6x longer than the simplest team: a pair of spacecraft executed on a single core.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

In this thesis the software and hardware components for a CubeSat module to enable collaborative behaviors has been discussed in detail. Through the use of an on-board GPS receiver, COTS S-Band transceiver, and standard microcontroller, all of the necessary functions can be satisfied. It was determined that for this application a relatively simple force model including Earth two-body and J_2 effects, drag, SRP, and third-body accelerations from the Sun and Moon is sufficient to represent the LEO environment. Furthermore, by applying an extended Kalman filter to GPS measurements, six parameters (Earth, Sun, and Moon gravitational parameters, J_2 of the Earth, and coefficients for drag and SRP) can be updated to improve the model accuracy. Sharing these parameters with spacecraft teammates improves the on-board orbit propagation so that the time between broadcast messages increases anywhere from 3 to 15x, depending on the GPS receiver accuracy and mission-specific tolerance. This reduction in inter-satellite communication provides for realization of power savings, arguably the most sparse resource available to a CubeSat. Using a common microcontroller, less than 50 mJ of energy per orbit per spacecraft are required to conduct the filtering and propagation; likewise, for every second of transmitting or receiver 4 J or 1 J, respectively, of energy is required. Assuming a data rate of 19.2 kbps sharing the raw GPS state will require approximately 0.1 seconds of transmission [10]. Since for every transmit there must be a receive, a broadcast can be estimated at costing 0.5 J. Therefore, in order to justify the use of the filter, at least one broadcast for every 10 orbits must be removed. Far greater savings (reducing total number of messages per orbit by 2-15x), is successfully demonstrated for most accuracy-tolerance pairings in Chapter 5's simulations.

The first suggestions for future work is to including CubeSat propulsion in the extended Kalman filter. Maintaining a CubeSat formation will require each spacecraft to be fitted with some sort of propulsion device, therefore extensions to the model could include cold gas or nano-arc thrusters with an associated controller. Furthermore, completing the small necessary changes to include attitude propagation and determination should be completed. To date, the most difficult challenge

with regards to implementing attitude as well as position/velocity is the need for reliable comparison test data from a COTS simulator or comparable HORIZONS resource.

The algorithms demonstrated in this thesis provide a solution for one of the pieces needed to enable collaborative behaviors among CubeSats. By using a tuned propagation model and EKF, tracking the Cartesian state of teammates reduces the amount of inter-satellite communication required. Initial parallel implementations of these algorithms demonstrate good scalability characteristics. Using multicore microcontrollers, with core throttling implemented, could provide further power savings; by adding cores with reduced clock speeds, total power consumed is reduced while keeping the execution time constant. This stems from the fact that the power consumed by a processor is proportional to the square or cube of its clock frequency [41].

With the goal being to provide a technology to enable collaborative behaviors, mentioning of specific implementations which could benefit from this module is necessary. The first, most obvious enabled task is collision avoidance. Although not a collaborative behavior, teams of satellites in close proximity must be equipped with a mechanism to ensure collision avoidance. Using the software discussed in this thesis provides this capability directly. A second capability provided is a light-weight, standardized message and message preface for inter-satellite communication. Iteration on the message structure is in order, but defining a standard allows for development of hardware and software which meet that standard. Satellites not originally intended to communicate when in LEO could, as a result of extended mission modifications, be fitted such that inter-satellite communication is already enabled. Additional work is planned for the messaging framework discussed. Current standards for heterogeneous systems, such as JAUS, are far too large and complex for a low-power CubeSat communications system. A light-weight framework such as the one proposed in this thesis is better suited to the CubeSat platform; however, modifications and extensions are needed for the standard to work well for a system of heterogeneous vehicles including unmanned aerial, ground, or surface vehicles.

Situational awareness (SA) is becoming a popular topic among mission proposals. One example mission is the acquisition, observation, and identification of an unknown object in orbit. A team of CubeSats, equipped with visual and infrared imagers, lasers for range measurements, and other sensors can acquire and observe an unknown object. Extending this work to include high

level commands for data collection could then use this module's outputs for mission planning. For example, if a team of three CubeSats is each fitted with a different SA sensor, the CubeSat with an infrared sensor may identify a feature of interest better captured by a visual imager. At the same time, the CubeSat fitted with a laser range sensor identifies a different feature which also merits visual imager observations. The spacecraft fitted with the visual imager receives a message from each of its teammates with a recommended task. With the collaborative module, first, the communication framework is already in place allowing the sharing of sensor data or results. Second, the visual imaging CubeSat already has the necessary information of where each of the other CubeSats was located at the time the feature was detected; therefore, it can optimally plan a trajectory, while considering its current tasking, to explore these two other features. The problem has the additional complexity of understanding the unknown object's orbit and attitude. Once its motion is characterized, however, the module can propagate its state just as if it were a teammate.

A second application, possibly more demanding depending on SA tolerances, is synthetic aperture radar (SAR) [69]. SAR takes distributed sensor data such as from an antenna or imager and fuses it to create more valuable data. In the case of distributed antennas, a group of satellites, fitted with small dishes, can "simulate" a much larger antenna that may not even fit on a launch vehicle. For imagers observing the Earth, higher resolution or possibly even 3D renderings of features, is enabled with an SAR system. Using the collaborative module for sharing state information greatly simplifies execution of formation maintenance algorithms. SAR has demanding tolerances for pointing and even more so for knowledge. For a team of spacecraft in a SAR formation, having shared state knowledge allows for optimal formation reorganization to occur in a distributed rather than centralized system. With each spacecraft equipped with the same formation reorganization algorithm, each CubeSat will come to the same solution and the reorganization can occur without inter-satellite communication.

Simulations also indicate that the various algorithms provide good scalability, specifically for shared memory architectures. Near-term work includes implementing the algorithms on a shared memory multicore microcontroller architecture such as the XMOS or ARM Cortex. Both XMOS and ARM Cortex chips provide exceptional computational capabilities while requiring power more similar to a microcontroller than the chip multiprocessors (CMPs) becoming popular in current

laptop and desktop computers. Implementing the collaborative module software on a CMP microcontroller platform will also, in general, improve the CubeSat on-board computational capabilities. Many of the data fusion algorithms that could be utilzed for distributed sensing in LEO are computationally intensive; providing some horsepower on-board could reduce the amount of downlink required. Power consumption measurements can be compared with a simple microcontroller, Texas Instrument MSP430 for example, to experimentally determine energy savings. Additionally, extensions to this work to include control algorithms for attitude and orbit control devices within the propagation model and EKF will improve its performance for satellites equipped with such hardware. With these additions, simulations of formations conducting SA or SAR operations will provide further insight into the achievable power savings and overall system efficiency utilizing the collaborative module.

Finally, the end goal of this research is to develop a CubeSat and demonstrate the capability in space. Work has already begun on designing a CubeSat bus structure composed mostly of COTS components; the developed CubeSat will piggy-back a ride into LEO with a companion spacecraft designed to demonstrate other technologies to help realize small satellite formations. The CubeSat and componanion nanosat will be fitted with complementary sensors to demonstrate the advantage with regards to data value when using a distributed sensor network in LEO rather than a conventional monolith. The next decade or two will hopefully prove successful in developing and delivering a full suite of technologies that will realize some of the first high-performing small satellite formations.

APPENDIX A

SPHERICAL HARMONIC COEFFICIENTS

The coefficients utilized for the spherical harmonic disturbed force models discussed in Chapter 2 are provided in this appendix. Table 10 provides the J vector of 20 elements, Table 11 provides the 200 coefficients that compose the lower triangular C matrix, and Table 17 provides the 200 coefficients that compose the lower triangular S matrix. For details on how this set of coefficients is utilized reference section 2.3.1.1. Recall that the C and S matrices are lower triangular; all zero-values have been removed for conciseness.

Table 10: First 20 Elements of the Earth's Non-Spherical Potential J Coefficient Vector.

Degree	**Value**
1	0
2	$1.082635666551098 \times 10^{-3}$
3	$-2.532473691332948 \times 10^{-6}$
4	$-1.619974305782220 \times 10^{-6}$
5	$-2.279051260821010 \times 10^{-7}$
6	$5.406167899402316 \times 10^{-7}$
7	$-3.505229256320887 \times 10^{-7}$
8	$-2.040167670104854 \times 10^{-7}$
9	$-1.221502454541323 \times 10^{-7}$
10	$-2.443429842937576 \times 10^{-7}$
11	$2.434963382615578 \times 10^{-7}$
12	$-1.821848394711900 \times 10^{-7}$
13	$-2.168367521591649 \times 10^{-7}$
14	$1.221120058595970 \times 10^{-7}$
15	$-1.221474112725108 \times 10^{-8}$
16	$2.707353862540012 \times 10^{-8}$
17	$-1.135130103414931 \times 10^{-7}$
18	$-3.709037562832905 \times 10^{-8}$
19	$2.062977958047096 \times 10^{-8}$
20	$-1.380299702218782 \times 10^{-7}$

Table 11: First 200 Elements of the Earth's Non-Spherical Potential C Coefficient Vector.

Degree	Order	Value
2	1	$-2.64116010268655 \times 10^{-10}$
2	2	$1.57457641956283 \times 10^{-6}$
3	1	$2.19316394848765 \times 10^{-6}$
3	2	$3.09048263864750 \times 10^{-7}$
3	3	$1.00578979458189 \times 10^{-7}$
4	1	$-5.08661062228600 \times 10^{-7}$
4	2	$7.83767762961274 \times 10^{-8}$
4	3	$5.92149143016924 \times 10^{-8}$
4	4	$-3.98210283783597 \times 10^{-9}$
5	1	$-5.38678113415053 \times 10^{-8}$
5	2	$1.05533056232411 \times 10^{-7}$
5	3	$-1.49275613434991 \times 10^{-8}$
5	4	$-2.29957041408669 \times 10^{-9}$
5	5	$4.30383433928167 \times 10^{-10}$
6	1	$-5.97205817458183 \times 10^{-8}$
6	2	$6.05489424956441 \times 10^{-9}$
6	3	$1.18670938107258 \times 10^{-9}$
6	4	$-3.25642689634495 \times 10^{-10}$
6	5	$-2.15624065075569 \times 10^{-10}$
6	6	$2.20556947824574 \times 10^{-12}$
7	1	$2.05589683116214 \times 10^{-7}$
7	2	$3.29101822510991 \times 10^{-8}$
7	3	$3.52783813794626 \times 10^{-9}$
7	4	$-5.83948189565351 \times 10^{-10}$
7	5	$5.87556579908026 \times 10^{-13}$
7	6	$-2.49047562056313 \times 10^{-11}$
7	7	$2.79424568680407 \times 10^{-14}$
8	1	$1.59151737841018 \times 10^{-8}$
8	2	$6.57198291762479 \times 10^{-9}$
8	3	$-1.95920490784550 \times 10^{-10}$
8	4	$-3.18948041346518 \times 10^{-10}$
8	5	$-4.65082488395712 \times 10^{-12}$
8	6	$-1.84224421783171 \times 10^{-12}$
8	7	$3.42969746843817 \times 10^{-13}$
8	8	$-1.58120067188201 \times 10^{-13}$

Table 11: First 200 Elements of the Earth's Non-Spherical Potential C Coefficient Vector. (cont.)

Degree	Order	Value
9	1	$9.23672628114424 \times 10^{-8}$
9	2	$1.48324528879733 \times 10^{-9}$
9	3	$-1.21390072672810 \times 10^{-9}$
9	4	$-8.02209934611960 \times 10^{-12}$
9	5	$-1.66824519548229 \times 10^{-12}$
9	6	$8.28999459310154 \times 10^{-13}$
9	7	$-2.24850853425979 \times 10^{-13}$
9	8	$6.14948507100870 \times 10^{-14}$
9	9	$-3.66381155041911 \times 10^{-15}$
10	1	$5.17595012382178 \times 10^{-8}$
10	2	$-5.58831049231352 \times 10^{-9}$
10	3	$-4.09099978221606 \times 10^{-11}$
10	4	$-4.97511085346748 \times 10^{-11}$
10	5	$-3.06014643924183 \times 10^{-12}$
10	6	$-2.60877892497165 \times 10^{-13}$
10	7	$6.95269233611296 \times 10^{-15}$
10	8	$4.65000717331664 \times 10^{-15}$
10	9	$2.32978065448690 \times 10^{-15}$
10	10	$4.17255737331005 \times 10^{-16}$
11	1	$9.21100659963780 \times 10^{-9}$
11	2	$1.04138255826201 \times 10^{-9}$
11	3	$-1.41055347197303 \times 10^{-10}$
11	4	$-1.59803534739817 \times 10^{-11}$
11	5	$1.48886065662372 \times 10^{-12}$
11	6	$-6.14145554566159 \times 10^{-15}$
11	7	$1.93423754345355 \times 10^{-15}$
11	8	$-3.00235014532265 \times 10^{-16}$
11	9	$-1.91056807972482 \times 10^{-16}$
11	10	$-4.95810407053177 \times 10^{-17}$
11	11	$9.35450933241795 \times 10^{-18}$
12	1	$-3.03380435788867 \times 10^{-8}$
12	2	$6.50886501653315 \times 10^{-10}$
12	3	$1.47566168791252 \times 10^{-10}$
12	4	$-2.10261283577496 \times 10^{-11}$
12	5	$8.21922257969734 \times 10^{-13}$

Table 11: First 200 Elements of the Earth's Non-Spherical Potential C Coefficient Vector. (cont.)

Degree	Order	Value
12	6	$7.43013490690677 \times 10^{-15}$
12	7	$-4.22968437221207 \times 10^{-15}$
12	8	$-5.74984702896357 \times 10^{-16}$
12	9	$1.01579170368922 \times 10^{-16}$
12	10	$-1.84960473761282 \times 10^{-18}$
12	11	$4.99678662951826 \times 10^{-19}$
12	12	$-2.18167801025339 \times 10^{-20}$
13	1	$-2.80204610503059 \times 10^{-8}$
13	2	$2.24553948148825 \times 10^{-9}$
13	3	$-6.59647559259381 \times 10^{-11}$
13	4	$-8.57757145799593 \times 10^{-13}$
13	5	$1.07646074194629 \times 10^{-12}$
13	6	$-5.24187216898044 \times 10^{-14}$
13	7	$3.81757645748035 \times 10^{-16}$
13	8	$-1.13223672611720 \times 10^{-16}$
13	9	$2.66009859222223 \times 10^{-17}$
13	10	$4.60156761858571 \times 10^{-18}$
13	11	$-5.87395217685783 \times 10^{-19}$
13	12	$-5.84119950514783 \times 10^{-20}$
13	13	$-2.23995037680356 \times 10^{-20}$
14	1	$-9.86487798180249 \times 10^{-9}$
14	2	$-1.30891945373917 \times 10^{-9}$
14	3	$9.31470786143613 \times 10^{-11}$
14	4	$2.90709761278424 \times 10^{-13}$
14	5	$3.85538668660129 \times 10^{-13}$
14	6	$-1.86942510805174 \times 10^{-14}$
14	7	$2.84652835090872 \times 10^{-15}$
14	8	$-2.12984493258408 \times 10^{-16}$
14	9	$1.65787766677918 \times 10^{-17}$
14	10	$1.83787722321921 \times 10^{-18}$
14	11	$7.41147641145211 \times 10^{-20}$
14	12	$4.54035085653103 \times 10^{-21}$
14	13	$2.35273446775454 \times 10^{-21}$
14	14	$-7.15424566088809 \times 10^{-22}$

Table 11: First 200 Elements of the Earth's Non-Spherical Potential C Coefficient Vector. (cont.)

Degree	Order	Value
15	1	$4.79291317623550 \times 10^{-9}$
15	2	$-6.76312321871048 \times 10^{-10}$
15	3	$1.15054304841824 \times 10^{-10}$
15	4	$-5.73020374730396 \times 10^{-12}$
15	5	$1.17732492120362 \times 10^{-13}$
15	6	$2.17982245217838 \times 10^{-14}$
15	7	$2.81308140077735 \times 10^{-15}$
15	8	$-1.11561112255507 \times 10^{-16}$
15	9	$3.56695137989511 \times 10^{-18}$
15	10	$2.24697962464184 \times 10^{-19}$
15	11	$-2.51342051484080 \times 10^{-21}$
15	12	$-5.99090261184504 \times 10^{-21}$
15	13	$-5.71999225798909 \times 10^{-22}$
15	14	$1.38030331811531 \times 10^{-23}$
15	15	$-9.20098089182282 \times 10^{-24}$
16	1	$1.28980648108199 \times 10^{-8}$
16	2	$-7.34729217213856 \times 10^{-10}$
16	3	$-6.23443148737958 \times 10^{-11}$
16	4	$4.65703602509090 \times 10^{-12}$
16	5	$-8.70487133836523 \times 10^{-14}$
16	6	$6.40250963655633 \times 10^{-15}$
16	7	$-2.45449888813586 \times 10^{-16}$
16	8	$-4.39126302487508 \times 10^{-17}$
16	9	$-3.28290368156584 \times 10^{-18}$
16	10	$-1.28114141866672 \times 10^{-19}$
16	11	$1.63003421386454 \times 10^{-20}$
16	12	$1.41036536727480 \times 10^{-21}$
16	13	$9.21778446573651 \times 10^{-23}$
16	14	$-1.36453778782960 \times 10^{-23}$
16	15	$-1.29034994152451 \times 10^{-24}$
16	16	$-6.06592392864022 \times 10^{-25}$

Table 11: First 200 Elements of the Earth's Non-Spherical Potential C Coefficient Vector. (cont.)

Degree	Order	Value
17	1	$-1.21315456428897 \times 10^{-8}$
17	2	$-5.51372828519940 \times 10^{-10}$
17	3	$9.99611837013368 \times 10^{-12}$
17	4	$5.98281191335467 \times 10^{-13}$
17	5	$-8.85983713237340 \times 10^{-14}$
17	6	$-3.85680456011221 \times 10^{-15}$
17	7	$5.05418174182951 \times 10^{-16}$
17	8	$4.99076399470657 \times 10^{-17}$
17	9	$2.91327761699583 \times 10^{-19}$
17	10	$-2.16400042417390 \times 10^{-20}$
17	11	$-6.51285752916271 \times 10^{-21}$
17	12	$8.85059919961787 \times 10^{-22}$
17	13	$4.15512685521009 \times 10^{-23}$
17	14	$-3.22547479140843 \times 10^{-24}$
17	15	$1.27700228242612 \times 10^{-25}$
17	16	$-8.66258166306962 \times 10^{-26}$
17	17	$-1.68972008714521 \times 10^{-26}$
18	1	$3.34979121558634 \times 10^{-9}$
18	2	$3.71493360829993 \times 10^{-10}$
18	3	$-6.94640583993597 \times 10^{-12}$
18	4	$4.13854165974782 \times 10^{-12}$
18	5	$2.52373157526144 \times 10^{-14}$
18	6	$3.24439382432337 \times 10^{-15}$
18	7	$9.37548274551529 \times 10^{-17}$
18	8	$2.48877695156683 \times 10^{-17}$
18	9	$-9.71839001398277 \times 10^{-19}$
18	10	$1.63137900383718 \times 10^{-20}$
18	11	$-1.41495191556696 \times 10^{-21}$
18	12	$-4.21566549936396 \times 10^{-22}$
18	13	$-6.49866004388497 \times 10^{-24}$
18	14	$-6.81153045408980 \times 10^{-25}$
18	15	$-2.89297260586498 \times 10^{-25}$
18	16	$7.19017665194425 \times 10^{-27}$
18	17	$2.95155555980705 \times 10^{-28}$
18	18	$4.21950544899519 \times 10^{-29}$

Table 11: First 200 Elements of the Earth's Non-Spherical Potential C Coefficient Vector. (cont.)

Degree	**Order**	**Value**
19	1	$-4.06463744950267 \times 10^{-9}$
19	2	$8.32813466724127 \times 10^{-10}$
19	3	$-9.10700957676345 \times 10^{-12}$
19	4	$9.92932806629640 \times 10^{-13}$
19	5	$3.43939304878330 \times 10^{-14}$
19	6	$-8.49712520088513 \times 10^{-16}$
19	7	$5.43027503570958 \times 10^{-17}$
19	8	$1.59819999493666 \times 10^{-17}$
19	9	$1.00145578582373 \times 10^{-19}$
19	10	$-6.06184679265185 \times 10^{-20}$
19	11	$1.77508372322647 \times 10^{-21}$
19	12	$-1.69314200197585 \times 10^{-23}$
19	13	$-3.48056728138120 \times 10^{-24}$
19	14	$-1.55740937420344 \times 10^{-25}$
19	15	$-4.42908165591516 \times 10^{-26}$
19	16	$-4.63943912714216 \times 10^{-27}$
19	17	$5.89929581456879 \times 10^{-28}$
19	18	$8.34719597577682 \times 10^{-29}$
19	19	$-1.04520491698405 \times 10^{-30}$

Table 11: First 200 Elements of the Earth's Non-Spherical Potential C Coefficient Vector. (cont.)

Degree	**Order**	**Value**
20	1	$2.45967387997350 \times 10^{-9}$
20	2	$4.38500566030569 \times 10^{-10}$
20	3	$-5.04966776030095 \times 10^{-12}$
20	4	$2.24529191040978 \times 10^{-13}$
20	5	$-2.66226002126215 \times 10^{-14}$
20	6	$1.61889929196719 \times 10^{-15}$
20	7	$-1.45347321792197 \times 10^{-16}$
20	8	$1.82870837228241 \times 10^{-18}$
20	9	$3.30654335361792 \times 10^{-19}$
20	10	$-3.42055712785231 \times 10^{-20}$
20	11	$8.69089313094141 \times 10^{-22}$
20	12	$-2.29147215182579 \times 10^{-23}$
20	13	$5.98089226085209 \times 10^{-24}$
20	14	$1.63089700284423 \times 10^{-25}$
20	15	$-2.51654630895412 \times 10^{-26}$
20	16	$-9.04242845236091 \times 10^{-28}$
20	17	$2.69357247643100 \times 10^{-29}$
20	18	$8.60107033131425 \times 10^{-30}$
20	19	$-1.93001695389513 \times 10^{-31}$
20	20	$3.74403531184239 \times 10^{-32}$

Table 12: First 200 Elements of the Earth's Non-Spherical Potential S Coefficient Vector.

Degree	Order	Value
2	1	$1.80328626938575 \times 10^{-9}$
2	2	$-9.03867945725804 \times 10^{-7}$
3	1	$2.68058739588047 \times 10^{-7}$
3	2	$-2.11430258564897 \times 10^{-7}$
3	3	$1.97222471690002 \times 10^{-7}$
4	1	$-4.49266073792088 \times 10^{-7}$
4	2	$1.48125963780527 \times 10^{-7}$
4	3	$-1.20105710832137 \times 10^{-8}$
4	4	$6.52506165632199 \times 10^{-9}$
5	1	$-8.08164344684904 \times 10^{-8}$
5	2	$-5.23289948626866 \times 10^{-8}$
5	3	$-7.10243684371874 \times 10^{-9}$
5	4	$3.87895436721787 \times 10^{-10}$
5	5	$-1.64815842321244 \times 10^{-9}$
6	1	$2.08634402837331 \times 10^{-8}$
6	2	$-4.65008028282337 \times 10^{-8}$
6	3	$1.85269554235614 \times 10^{-10}$
6	4	$-1.78456634580693 \times 10^{-9}$
6	5	$-4.32981557852457 \times 10^{-10}$
6	6	$-5.53107085428193 \times 10^{-11}$
7	1	$6.96204506847799 \times 10^{-8}$
7	2	$9.26155156996146 \times 10^{-9}$
7	3	$-3.05871864014794 \times 10^{-9}$
7	4	$-2.63460085763113 \times 10^{-10}$
7	5	$6.34719143805473 \times 10^{-12}$
7	6	$1.05360213165884 \times 10^{-11}$
7	7	$4.47369524249475 \times 10^{-13}$
8	1	$4.04728618844063 \times 10^{-8}$
8	2	$5.36162685305228 \times 10^{-9}$
8	3	$-8.69234389256859 \times 10^{-10}$
8	4	$9.11135247887674 \times 10^{-11}$
8	5	$1.61442504707036 \times 10^{-11}$
8	6	$8.62841778881997 \times 10^{-12}$
8	7	$3.81795125729228 \times 10^{-13}$
8	8	$1.53676970155775 \times 10^{-13}$

Table 12: First 200 Elements of the Earth's Non-Spherical Potential S Coefficient Vector. (cont.)

Degree	Order	Value
9	1	$1.39050436306340 \times 10^{-8}$
9	2	$-2.19534577375178 \times 10^{-9}$
9	3	$-5.61446805385228 \times 10^{-10}$
9	4	$1.70321750576656 \times 10^{-11}$
9	5	$-5.52744002892468 \times 10^{-12}$
9	6	$2.94393735757914 \times 10^{-12}$
9	7	$-1.84734580755720 \times 10^{-13}$
9	8	$-9.80794185392354 \times 10^{-16}$
9	9	$7.46372586866818 \times 10^{-15}$
10	1	$-8.10033309697057 \times 10^{-8}$
10	2	$-3.04903872482325 \times 10^{-9}$
10	3	$-8.98731199970904 \times 10^{-10}$
10	4	$-4.65467695981544 \times 10^{-11}$
10	5	$-3.14207495798943 \times 10^{-12}$
10	6	$-5.53678606726025 \times 10^{-13}$
10	7	$-2.56494799575600 \times 10^{-15}$
10	8	$-1.05057486537151 \times 10^{-14}$
10	9	$-7.05243506852606 \times 10^{-16}$
10	10	$-9.91346360330873 \times 10^{-17}$
11	1	$-1.60090891751217 \times 10^{-8}$
11	2	$-5.12616535556671 \times 10^{-9}$
11	3	$-6.86538355768725 \times 10^{-10}$
11	4	$-2.68509192751822 \times 10^{-11}$
11	5	$1.97301425610179 \times 10^{-12}$
11	6	$1.35022266195787 \times 10^{-13}$
11	7	$-3.72979204545534 \times 10^{-14}$
11	8	$1.16911305459673 \times 10^{-15}$
11	9	$2.58679180734327 \times 10^{-16}$
11	10	$-1.74817362005535 \times 10^{-17}$
11	11	$-1.40939375155297 \times 10^{-17}$
12	1	$-2.44365691924361 \times 10^{-8}$
12	2	$1.41848953628937 \times 10^{-9}$
12	3	$9.33591432942468 \times 10^{-11}$
12	4	$1.19215660024536 \times 10^{-12}$
12	5	$2.02029648802772 \times 10^{-13}$

Table 12: First 200 Elements of the Earth's Non-Spherical Potential S Coefficient Vector. (cont.)

Degree	Order	Value
12	6	$9.24196800677749 \times 10^{-14}$
12	7	$7.93588616109859 \times 10^{-15}$
12	8	$3.76138289027949 \times 10^{-16}$
12	9	$6.04932858644344 \times 10^{-17}$
12	10	$9.23069081522324 \times 10^{-18}$
12	11	$-2.81067438154076 \times 10^{-19}$
12	12	$-9.96830585198647 \times 10^{-20}$
13	1	$2.10792953682211 \times 10^{-8}$
13	2	$-2.54546202980612 \times 10^{-9}$
13	3	$2.98936582105566 \times 10^{-10}$
13	4	$-2.75740767110331 \times 10^{-12}$
13	5	$1.23975596990552 \times 10^{-12}$
13	6	$-9.38588730973625 \times 10^{-15}$
13	7	$-9.25858098138928 \times 10^{-16}$
13	8	$-1.11091664998257 \times 10^{-16}$
13	9	$4.92719029912883 \times 10^{-17}$
13	10	$-4.12410923660567 \times 10^{-18}$
13	11	$-6.37723453510177 \times 10^{-20}$
13	12	$1.64080669247011 \times 10^{-19}$
13	13	$2.49368475733474 \times 10^{-20}$
14	1	$1.51674745051786 \times 10^{-8}$
14	2	$-1.47712326523880 \times 10^{-10}$
14	3	$5.02464333209363 \times 10^{-11}$
14	4	$-4.10944650690018 \times 10^{-12}$
14	5	$-2.20837939834081 \times 10^{-13}$
14	6	$2.41207792911208 \times 10^{-15}$
14	7	$-2.97543188928986 \times 10^{-16}$
14	8	$-9.41397652268562 \times 10^{-17}$
14	9	$1.47686377882241 \times 10^{-17}$
14	10	$-6.13061326060726 \times 10^{-20}$
14	11	$-1.84947364206172 \times 10^{-19}$
14	12	$-1.66921202043323 \times 10^{-20}$
14	13	$3.29541105822990 \times 10^{-21}$
14	14	$-6.63894919630624 \times 10^{-23}$

Table 12: First 200 Elements of the Earth's Non-Spherical Potential S Coefficient Vector. (cont.)

Degree	Order	Value
15	1	$5.32824901600067 \times 10^{-9}$
15	2	$-9.98281156939221 \times 10^{-10}$
15	3	$3.80375254984811 \times 10^{-11}$
15	4	$9.72202959474123 \times 10^{-13}$
15	5	$7.33014856094482 \times 10^{-14}$
15	6	$-2.42021685661327 \times 10^{-14}$
15	7	$2.39335318234786 \times 10^{-16}$
15	8	$7.70787768701261 \times 10^{-17}$
15	9	$1.01907876723587 \times 10^{-17}$
15	10	$3.21665185842029 \times 10^{-19}$
15	11	$3.55730648073479 \times 10^{-20}$
15	12	$2.88454544137511 \times 10^{-21}$
15	13	$-9.23674968098107 \times 10^{-23}$
15	14	$-6.46682428792150 \times 10^{-23}$
15	15	$-2.26819093515914 \times 10^{-24}$
16	1	$1.64219010554590 \times 10^{-8}$
16	2	$8.40463311912177 \times 10^{-10}$
16	3	$-3.92214116329476 \times 10^{-11}$
16	4	$5.46992723731139 \times 10^{-12}$
16	5	$-2.47357377248890 \times 10^{-14}$
16	6	$-1.64327733957018 \times 10^{-14}$
16	7	$-2.63303212186549 \times 10^{-16}$
16	8	$1.11906985494827 \times 10^{-17}$
16	9	$-5.80912641631725 \times 10^{-18}$
16	10	$1.25214825372666 \times 10^{-19}$
16	11	$-2.73323414768626 \times 10^{-21}$
16	12	$4.84289000226218 \times 10^{-22}$
16	13	$7.04239753710891 \times 10^{-24}$
16	14	$-2.72660113661557 \times 10^{-23}$
16	15	$-2.93348588643821 \times 10^{-24}$
16	16	$4.68709890250818 \times 10^{-26}$

Table 12: First 200 Elements of the Earth's Non-Spherical Potential S Coefficient Vector. (cont.)

Degree	Order	Value
17	1	$-1.51640077693142 \times 10^{-8}$
17	2	$1.86855645147505 \times 10^{-10}$
17	3	$8.05118296913977 \times 10^{-12}$
17	4	$2.34009237751064 \times 10^{-12}$
17	5	$4.38638257752418 \times 10^{-14}$
17	6	$-9.67952829418396 \times 10^{-15}$
17	7	$-8.87441374095659 \times 10^{-17}$
17	8	$4.65000742407504 \times 10^{-18}$
17	9	$-2.31232991275343 \times 10^{-18}$
17	10	$1.04594540588936 \times 10^{-19}$
17	11	$4.50809725268444 \times 10^{-21}$
17	12	$6.30693655831826 \times 10^{-22}$
17	13	$5.06773578791232 \times 10^{-23}$
17	14	$2.61415646969174 \times 10^{-24}$
17	15	$1.20897165459252 \times 10^{-25}$
17	16	$1.04688974140677 \times 10^{-26}$
17	17	$-9.67657850696758 \times 10^{-27}$
18	1	$-1.82805467436530 \times 10^{-8}$
18	2	$2.73365274334325 \times 10^{-10}$
18	3	$-8.04453005248590 \times 10^{-12}$
18	4	$-5.99388350892957 \times 10^{-14}$
18	5	$1.10361567516150 \times 10^{-13}$
18	6	$-3.16548523765197 \times 10^{-15}$
18	7	$1.03071806007004 \times 10^{-16}$
18	8	$3.54532659549040 \times 10^{-18}$
18	9	$1.79410694113643 \times 10^{-18}$
18	10	$-1.32726733825660 \times 10^{-20}$
18	11	$4.34486589636556 \times 10^{-22}$
18	12	$-2.34784869000279 \times 10^{-22}$
18	13	$-3.63051588470158 \times 10^{-23}$
18	14	$-1.05393632338125 \times 10^{-24}$
18	15	$-1.44980840074406 \times 10^{-25}$
18	16	$4.60608979832419 \times 10^{-27}$
18	17	$3.70636679659280 \times 10^{-28}$
18	18	$-1.52869079721608 \times 10^{-28}$

Table 12: First 200 Elements of the Earth's Non-Spherical Potential S Coefficient Vector. (cont.)

Degree	**Order**	**Value**
19	1	$5.40849472797829 \times 10^{-10}$
19	2	$-5.52603624132760 \times 10^{-11}$
19	3	$1.33137907183221 \times 10^{-12}$
19	4	$-5.11191741990552 \times 10^{-13}$
19	5	$9.08805378318538 \times 10^{-14}$
19	6	$3.31887753486970 \times 10^{-15}$
19	7	$-8.41382053306638 \times 10^{-17}$
19	8	$-5.34443176002404 \times 10^{-18}$
19	9	$2.20484325902558 \times 10^{-19}$
19	10	$-1.35692028551230 \times 10^{-20}$
19	11	$1.12750734302307 \times 10^{-21}$
19	12	$6.53933578136793 \times 10^{-23}$
19	13	$-1.31771791084986 \times 10^{-23}$
19	14	$-4.21857604362023 \times 10^{-25}$
19	15	$-3.54368999726927 \times 10^{-26}$
19	16	$-1.50212711879719 \times 10^{-27}$
19	17	$-3.14203196442524 \times 10^{-28}$
19	18	$-2.31139276377640 \times 10^{-29}$
19	19	$2.00784771517528 \times 10^{-30}$

Table 12: First 200 Elements of the Earth's Non-Spherical Potential S Coefficient Vector. (cont.)

Degree	**Order**	**Value**
20	1	$3.10567431790212 \times 10^{-9}$
20	2	$3.71409466460781 \times 10^{-10}$
20	3	$4.13197548698299 \times 10^{-11}$
20	4	$-1.18990400040569 \times 10^{-12}$
20	5	$-2.17163377347690 \times 10^{-14}$
20	6	$-5.80869731145242 \times 10^{-16}$
20	7	$-4.83426894070179 \times 10^{-18}$
20	8	$7.72170791472084 \times 10^{-19}$
20	9	$-1.34932692785011 \times 10^{-19}$
20	10	$-5.07491206653631 \times 10^{-21}$
20	11	$-1.15215095234853 \times 10^{-21}$
20	12	$6.42675587820124 \times 10^{-23}$
20	13	$1.47571124336529 \times 10^{-24}$
20	14	$-2.03352781954058 \times 10^{-25}$
20	15	$-8.51650023096582 \times 10^{-28}$
20	16	$-2.48404874599197 \times 10^{-29}$
20	17	$-8.20777114784416 \times 10^{-29}$
20	18	$-4.97292860401105 \times 10^{-31}$
20	19	$6.92732017962630 \times 10^{-31}$
20	20	$-1.27340256602360 \times 10^{-31}$

APPENDIX B

COEFFICIENTS FOR FOURTH- THROUGH EIGHTH-ORDER RUNGE-KUTTA INTEGRATORS

Section 2.4.2 discusses the generic Runge-Kutta method for integrating ODEs. For neatness the Butcher Tableaus containing the necessary coefficients for fourth- through eighth-order Runge-Kutta methods are placed in this appendix rather than Chapter 2. The fourth-order coefficients are commonly available [55]. The higher-order coefficients are available from [31].

Table 13: Butcher Tableau for a Fourth-Order Runge-Kutta Integrator.

		$\beta_{\kappa\lambda}$			
$\kappa \mid \lambda$	α_κ	0	1	2	c_κ
0	0	0			$\frac{1}{3}$
1	$\frac{1}{2}$	$\frac{1}{2}$			$\frac{1}{6}$
2	$\frac{1}{2}$	0	$\frac{1}{2}$		$\frac{1}{3}$
3	1	0	0	1	$\frac{1}{3}$

Table 14: Butcher Tableau for a Fifth-Order Runge-Kutta Integrator [31].

		$\beta_{\kappa\lambda}$					
$\kappa \mid \lambda$	α_κ	0	1	2	3	4	c_κ
0	0	0					$\frac{31}{384}$
1	$\frac{1}{6}$	$\frac{1}{6}$					0
2	$\frac{4}{15}$	$\frac{4}{75}$	$\frac{16}{75}$				$\frac{1125}{2816}$
3	$\frac{2}{3}$	$\frac{5}{6}$	$-\frac{8}{3}$	$\frac{5}{2}$			$\frac{9}{32}$
4	$\frac{4}{5}$	$-\frac{8}{5}$	$\frac{144}{25}$	-4	$\frac{16}{25}$		$\frac{125}{768}$
5	1	$\frac{361}{320}$	$-\frac{18}{5}$	$\frac{407}{128}$	$-\frac{11}{80}$	$\frac{55}{128}$	$\frac{5}{66}$

Table 15: Butcher Tableau for a Sixth-Order Runge-Kutta Integrator [31].

		$\beta_{\kappa\lambda}$							
$\kappa \mid \lambda$	α_κ	0	1	2	3	4	5	6	c_κ
0	0	0							$\frac{77}{1440}$
1	$\frac{2}{33}$	$\frac{2}{33}$							0
2	$\frac{4}{33}$	0	$\frac{4}{33}$						0
3	$\frac{2}{11}$	$\frac{1}{22}$	0	$\frac{3}{22}$					$\frac{1771561}{6289920}$
4	$\frac{1}{2}$	$\frac{43}{64}$	0	$-\frac{165}{64}$	$\frac{77}{32}$				$\frac{32}{105}$
5	$\frac{2}{3}$	$-\frac{2383}{486}$	0	$\frac{1067}{54}$	$-\frac{26312}{1701}$	$\frac{2176}{1701}$			$\frac{243}{2560}$
6	$\frac{6}{7}$	$\frac{10077}{4802}$	0	$-\frac{5643}{686}$	$\frac{116259}{16807}$	$-\frac{6240}{16807}$	$\frac{1053}{2401}$		$\frac{16807}{74880}$
7	1	$-\frac{733}{176}$	0	$\frac{141}{8}$	$-\frac{335763}{23296}$	$\frac{216}{77}$	$-\frac{4617}{2816}$	$\frac{7203}{9152}$	$\frac{11}{270}$

Table 16: Butcher Tableau for a Seventh-Order Runge-Kutta Integrator [31].

		$\beta_{\kappa\lambda}$										
$\kappa \mid \lambda$	α_κ	0	1	2	3	4	5	6	7	8	9	c_κ
0	0	0										$\frac{41}{840}$
1	$\frac{2}{27}$	$\frac{2}{27}$										0
2	$\frac{1}{9}$	$\frac{1}{36}$	$\frac{1}{12}$									0
3	$\frac{1}{6}$	$\frac{1}{24}$	0	$\frac{1}{8}$								0
4	$\frac{5}{12}$	$\frac{5}{12}$	0	$-\frac{25}{16}$	$\frac{25}{16}$							0
5	$\frac{1}{2}$	$\frac{1}{20}$	0	0	$\frac{1}{4}$	$\frac{1}{5}$						$\frac{34}{105}$
6	$\frac{5}{6}$	$-\frac{25}{108}$	0	0	$\frac{125}{108}$	$-\frac{65}{27}$	$\frac{125}{54}$					$\frac{9}{35}$
7	$\frac{1}{6}$	$\frac{31}{300}$	0	0	0	$\frac{61}{225}$	$-\frac{2}{9}$	$\frac{13}{900}$				$\frac{9}{35}$
8	$\frac{2}{3}$	2	0	0	$-\frac{53}{6}$	$\frac{704}{45}$	$-\frac{107}{9}$	$\frac{67}{90}$	3			$\frac{9}{280}$
9	$\frac{1}{3}$	$-\frac{91}{108}$	0	0	$\frac{23}{108}$	$-\frac{976}{135}$	$\frac{311}{54}$	$-\frac{19}{60}$	$\frac{17}{6}$	$-\frac{1}{12}$		$\frac{9}{280}$
10	1	$\frac{2383}{4100}$	0	0	$-\frac{341}{164}$	$\frac{4496}{1025}$	$-\frac{301}{82}$	$\frac{2133}{4100}$	$\frac{45}{82}$	$\frac{45}{164}$	$\frac{18}{41}$	$\frac{41}{840}$

Table 17: List of Coefficients for an Eighth-Order Runge-Kutta Integrator [31].

$Variable_{\kappa,\lambda}$		$Value$
α_1	=	0.44368940376498182109599404281370
α_2	=	0.66553410564747274664399106422055
α_3	=	0.99830115847120911996598659633083
α_4	=	0.31550000000000000000000000000000
α_5	=	0.50544100948169068626516126737384
α_6	=	0.17142857142857142857142857142857
α_7	=	0.82857142857142857142857142857143
α_8	=	0.66543966121011562534953769255586
α_9	=	0.24878317968062652069722274560771
α_{10}	=	0.10900000000000000000000000000000
α_{11}	=	0.89100000000000000000000000000000
α_{12}	=	0.39950000000000000000000000000000
α_{13}	=	0.60050000000000000000000000000000
α_{14}	=	1.00000000000000000000000000000000
$\beta_{1,0}$	=	0.44368940376498183109599404281370
$\beta_{2,0}$	=	0.16638352641186818666099776605514
$\beta_{2,1}$	=	0.49915057923560455998299329816541
$\beta_{3,0}$	=	0.24957528961780227999149664908271
$\beta_{3,2}$	=	0.74872586885340683997448994724812
$\beta_{4,0}$	=	0.20661891163400602426556710393185
$\beta_{4,2}$	=	0.17707880377986347040380997288319
$\beta_{4,3}$	=	$-0.68197715413869494669377076815048 \times 10^{-1}$
$\beta_{5,0}$	=	0.10927823152666408227903890926157
$\beta_{5,3}$	=	$0.40215962642367995421990563690087 \times 10^{-2}$
$\beta_{5,4}$	=	0.39214118169078980444392330174325
$\beta_{6,0}$	=	$0.98899281409164665304844765434355 \times 10^{-1}$
$\beta_{6,3}$	=	$0.35138370227963966951204487356703 \times 10^{-2}$
$\beta_{6,4}$	=	0.12476099983160016621520625872489
$\beta_{6,5}$	=	$-0.55745546834989799643742901466348 \times 10^{-1}$

Table 17: List of Coefficients for an Eighth-Order Runge-Kutta Integrator [31]. (cont)

$Variable_{\kappa,\lambda}$		$Value$
$\beta_{7,0}$	=	$-0.36806865286242203724153101080691$
$\beta_{7,4}$	=	$-0.22273897469476007645024020944166 \times 10^{+1}$
$\beta_{7,5}$	=	$0.44368940376498182109599404281370$
$\beta_{7,6}$	=	$0.20497390027111603002159354092206 \times 10^{+1}$
$\beta_{8,0}$	=	$0.45467962641347150077351950603349 \times 10^{-1}$
$\beta_{8,5}$	=	$0.32542131701589147114677469648853$
$\beta_{8,6}$	=	$0.28476660138527908888182420573687$
$\beta_{8,7}$	=	$0.97837801675979152435868397271099 \times 10^{-2}$
$\beta_{9,0}$	=	$0.60842071062622057051094145205182 \times 10^{-1}$
$\beta_{9,5}$	=	$-0.21184565744037007526325275251206 \times 10^{-1}$
$\beta_{9,6}$	=	$0.19596557266170831957464490662983$
$\beta_{9,7}$	=	$-0.42742640364817603675144835342899 \times 10^{-2}$
$\beta_{9,8}$	=	$0.17434365736814911965323452558189 \times 10^{-1}$
$\beta_{10,0}$	=	$0.54059783296931917365785724111182 \times 10^{-1}$
$\beta_{10,6}$	=	$0.11029825597828926530283127648228$
$\beta_{10,7}$	=	$-0.12565008520072556414147763782250 \times 10^{-2}$
$\beta_{10,8}$	=	$0.36790043477581460136384043566339 \times 10^{-2}$
$\beta_{10,9}$	=	$-0.57780542770972073040840628571866 \times 10^{-1}$
$\beta_{11,0}$	=	$0.12732477068667114646645181799160$
$\beta_{11,7}$	=	$0.11448805006396105323658875721817$
$\beta_{11,8}$	=	$0.28773020709697992776202201849198$
$\beta_{11,9}$	=	$0.50945379459611363153735885079465$
$\beta_{11,10}$	=	$-0.14799682244372575900242144449640$
$\beta_{12,0}$	=	$-0.36526793876616740535848544394333 \times 10^{-2}$
$\beta_{12,5}$	=	$0.81629896012318919777819421247030 \times 10^{-1}$
$\beta_{12,6}$	=	$-0.38607735635693506490517694343215$
$\beta_{12,7}$	=	$0.30862242924605106450474166025206 \times 10^{-1}$
$\beta_{12,8}$	=	$-0.58077254528320602815829374733518 \times 10^{-1}$
$\beta_{12,9}$	=	$0.33598659328884971493143451362322$
$\beta_{12,10}$	=	$0.41066880401949958613549622786417$

Table 17: List of Coefficients for an Eighth-Order Runge-Kutta Integrator [31]. (cont)

$Variable_{\kappa,\lambda}$		$Value$
$\beta_{12,11}$	=	$-0.11840245972355985520633156154536 \times 10^{-1}$
$\beta_{13,0}$	=	$-0.12375357921245143254979096135669 \times 10^{+1}$
$\beta_{13,5}$	=	$-0.24430768551354785358734861366763 \times 10^{+2}$
$\beta_{13,6}$	=	$0.54779568932778656050436528991173$
$\beta_{13,7}$	=	$-0.44413863533413246374959896569346 \times 10^{+1}$
$\beta_{13,8}$	=	$0.10013104813713266094792617851022 \times 10^{+2}$
$\beta_{13,9}$	=	$-0.14995773102051758447170985073142 \times 10^{+2}$
$\beta_{13,10}$	=	$0.58946948523217013620824539651427 \times 10^{+1}$
$\beta_{13,11}$	=	$0.17380377503428984877616857440542 \times 10^{+1}$
$\beta_{13,12}$	=	$0.27512330693166730263758622860276 \times 10^{+2}$
$\beta_{14,0}$	=	$-0.35260859388334522700502958875588$
$\beta_{14,5}$	=	$-0.18396103144848270375044198988231$
$\beta_{14,6}$	=	$-0.65570189449741645138006879985251$
$\beta_{14,7}$	=	$-0.39086144880439863435025520241310$
$\beta_{14,8}$	=	$0.26794646712850022936584423271209$
$\beta_{14,9}$	=	$-0.10383022991382490865769858507427 \times 10^{+1}$
$\beta_{14,10}$	=	$0.16672327324258671664727346168501 \times 10^{+1}$
$\beta_{14,11}$	=	$0.49551925855315977067732967071441$
$\beta_{14,12}$	=	$0.11394001132397063228586738141784 \times 10^{+1}$
$\beta_{14,13}$	=	$0.51336696424658613688199097191534 \times 10^{-1}$
c_0	=	$0.32256083500216249913612900960247 \times 10^{-1}$
c_8	=	$0.25983725283715403018887023171963$
c_9	=	$0.92847805996577027788063714302190 \times 10^{-1}$
c_{10}	=	$0.16452339514764342891647731842800$
c_{11}	=	$0.17665951637860074367084298397547$
c_{12}	=	$0.23920102320352759374108933320941$
c_{13}	=	$0.39484274604202853746752118829325 \times 10^{-2}$
c_{14}	=	$0.30726495475860640406368305522124 \times 10^{-1}$

REFERENCES

[1] "Cubesat (wiki)." website, March. 2.1

[2] "Gps: Essentials of satellite navigation." website, compendium. 3.1

[3] "Wikipedia: File: Orbit1.svg." website, October. (document), 1

[4] "Nasa goddard space flight center: Formation flying - the afternoon 'a-train' satellite constellation," *The Earth Science Enterprise Series*, March 2003. FS-2003-1-053-GSFC. 1.2

[5] "California polythechnic state university and stanford university: Cubesat design specification." document, August 2009. http://www.cubesat.org/images/developers/cds_rev12.pdf. (document), 1.1

[6] "Air force space command: Space environment nanosat experiment." Solicitation, August 2010. FA8814-10-R-0002. 1.1

[7] "Darpa tactical technology office: System f6." Broad Agency Announcement (BAA), October 2010. DARPA-BAA-11-01. 1.2

[8] "Clyde space." website, 2011. http://www.clyde-space.com/. (document), 1.7

[9] "Microhard systems, inc.: 2.4 ghz industrial wireless modem." website, 2011. http://www.microhardcorp.com/MHX2420.php. (document), 42

[10] "Microhard systems, inc.: 2.4 ghz oem industrial wireless modem." website, 2011. http://www.microhardcorp.com/brochures/MHX2420.Brochure.Rev.3.11.pdf. 4.2.4, 4.2.4, 6

[11] "Microhard systems, inc.: Operating manual: Mhx-2420." website, 2011. http://www.microhardcorp.com/MHX2420.php. 4.2.4, 4.2.4

[12] "Nasa's jet propulsion lab: Horizons online ephemeris system." website, 2011. http://ssd.jpl.nasa.gov/horizons.cgi. 2.1

[13] "Pumpkin, inc.: Cubesat kit." website, 2011. http://www.cubesatkit.com/. 1.7

[14] "Space quest, ltd.: Gps-12-v1." data sheet, May 2011. http://www.spacequest.com/products/GPS-12-V1.pdf. 3.1, 3

[15] "Texas instruments: Msp430x2xx family users guide." web document, April 2011. Literature Number: SLAU144H. (document)

[16] Abdullah, A. N. M., Moinudeen, H., and Al-Khateeb, W., "Scalability and performance analysis of ieee 802.11a," in *Canadian Conference on Electrical and Computer Engineering*, (Saskatoon), May 2005. 4.2.1

[17] Aghav, S. and Gangal, S. A., "Development of on-board orbit determination system for low eart orbit (leo) satellite using global navigation satellite system (gnss) receiver," in *Dimensions and Directions of Geospatial Industry*, (Hyderabad, India), Geospatial World Forum, Jan. 2011. Paper Reference No. PN-288. 1.6, 3.2.2

[18] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*. New York, NY: Dover Publications, Inc., first ed., 1971. 2, 2.1

[19] Beech, W. A., Nielsen, D. E., and Taylor, J., "Ax.25 link access protocol for amateur packet radio (version 2.2)." website. 4.2.3

[20] Belousov, S. L., *Tables of normalized associated Legendre polynomials*. Oxford: Pergamon Press, 1962. 2.3.1.1

[21] Berry, M. M. and Healy, L. M., "Implementation of gauss-jackson integration for orbit propagation," *The Journal of the Astronautical Sciences*, vol. 52, July-September 2004. pp. 331-357. 1.5

[22] Brammer, R. F., "Real-time shipboard orbit determination using kalman filtering techniques," in *IEEE Transactions on Aerospace and Electronic Systems*, 1975. ISSN: 0018-9251. 1.6

[23] Chen, J. L., Wilson, C. R., Tapley, B. D., Blankenship, D., and Young, D., "Antarctic regional ice loss rates from grace," *Earth and Planetary Scient Letters*, vol. 266, February 2008. doi:10.1016/j.epsl.2007.10.057. 2.3.1

[24] Chui, C. K. and Chane, G., *Kalman Filtering with Real-Time Applications*. New York, NY: Springer-Verlag, second ed., 1971. 1.6

[25] Clohessy, W. H. and Wiltshire, R. S., "Terminal guidance for rendezvous in space," *Journal of Aerospace Science*, vol. 27, September 1960. 1.3, 2

[26] Cole, S., "Nasa names mishap board for taurus xl launch failure investigation." website, March 2011. http://www.nasa.gov/home/hqnews/2011/mar/11-071_Glory_MIB.html. 1

[27] Cole, S., "Nasas glory satellite fails to reach orbit." website, March 2011. http://www.nasa.gov/home/hqnews/2011/mar/HQ_11-050_N0_Glory.html. 1

[28] Colitti, W., Steenhaut, K., Descouvermont, N., and Dunkels, A., "Satellite based wireless sensor netowrks – global scale sensing with nano- and pico-satellites," in *Proceedings of 6th ACM Conference on Embedded Network Sensor Systems*, (Raleigh, NC, USA), ACM, 2008. ACM 978-1-59593-990-6/08/11. 4.2.2

[29] D'Amico, S., Gill, E., and Montenbruck, O., "Relative orbit control design for the prisma formation flying mission," in *Proceedings of AIAA Guidance, Navigation, and Control Conference*, (Keystone, CO, USA), IAA, 2006. 1

[30] D'Amico, S. and Montenbruck, O., "Proximity operations of formation flying spacecraft using an eccentricity/inclination vector separation," *AIAA Journal of Guidance, Control and Dynamics*, vol. 29, May-June 2006. pp. 554-563. 1

[31] Feehlberg, E., "Classical fifth-, sixth-, seventh-, and eighth-order runge-kutta formulas with stepsize control," *NASA Technical Reports*, 1968. (document), 1.5, 2, 2.4.2, 2.4.2, B, 14, 15, 16, 17, 17, 17

[32] Finch, C., "Po.daac grace data access." website, April 2010. http://podaac.jpl.nasa.gov/grace/data access.html. 2.3.1

[33] Gill, E., D'Amico, S., and Montenbruck, O., "Autonomous formation flying for the prisma mission," *AIAA Journal of Spacecraft and Rockets*, vol. 44, May-June 2007. pp. 671-681, DOI 10.2514/1.23015. 1

[34] Gill, E., Montenbruck, O., D'Amico, S., and Persson, S., "Autonomous satellite formation flying for the prisma technology demonstration mission," in *Proceedings of 16th AAS/AIAA Space Flight Mechanics Conference*, (Tampa, FL, USA), AAS/AIAA, 2006. 1

[35] Haykin, S., *Kalman Filtering and Neural Networks*. New York, NY: Wiley-Interscience Publication, first ed., 2001. 3.2

[36] Iyengar, S. S. and Brooks, R. R., *Distributed Sensor Networks*. Boca Raton, FL: Chapman and Hall, CRC Press, first ed., 2005. ISBN 1-58488-383-9. 1.4

[37] Jeffreys, H. and Jeffreys, B., *Methods of Mathematical Physics*. Cambridge, MA, USA: Cambridge University Press, third ed., 2000. ISBN-10: 9780521664028. 1.5

[38] Jekeli, C., Bastos, L., and Fernandes, J., *Gravity, Geoid and Space Missions*, vol. 129. New York, NY, USA: Springer, 2004. ISBN: 0939-9585. 2.3.1

[39] Karlgaard, C. D., *Second-Order Relative Motion Equations*. Master thesis, Virginia Polytechnic Institute and State University, Department of Aerospace Engineering, 2001. 1.3

[40] Kaula, W. M., *Theory of Satellite Geodesy*. Waltham, Massachusetts: Blaisdell Publishing Company, first ed., 1966. 2, 2.3.1.1

[41] Korthikanti, V. A. and Agha, F., "Analysis of parallel algorithms for energy conservation in scalable multicore architectures," in *International Conference on Parallel Processing*, ICPP, 2009. 5.5.1, 6

[42] Larson, W. J. and Wertz, J. R., *Space Mission Analysis and Design*. New York, NY and El Segundo CA: Microcosm Press and Springer, third ed., 2006. 2.3.2, 2.3.3, 4.2.4

[43] L'Ecuyer, T. S. and Jiang, J. H., "Touring the atmosphere aboard the a-train," *Physics Today*, vol. 63, July 2010. pp. 36-41. 1

[44] Li, J., Blake, C., Couto, D. S. J. D., Lee, H. I., and Morris, R., "Capacity of ad hoc wireless networks," in *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking*, (Rome, Italy), ACM, 2001. 4.2.1

[45] Liggins, M. E., Hall, D. L., and Llinas, J., *Handbook of Multisensor Data Fusion: Theory and Practice*. Boca Raton, FL: CRC Press, second ed., 2009. ISBN-13 978-1-4200-5308-1. 1.4

[46] Lovell, R. A., Horneman, K. R., Tragesser, S. G., and Tollefson, M. V., "A guidance algorithm for formation reconfiguration and maintenance based on the perturbed clohessy-wiltshire equations," in *Proceedings of AAS/AIAA Astrondynamics Specialist Conference*, (Big Sky, MT, USA), AAS/AIAA, 2003. AAS 03-649. 1.3

[47] Mas, I. A. and Kitts, C. A., "A flight-proven 2.4ghx ism band cots communications system for small satellites," in *Proceedings of Small Satellite Conference*, (Logan, UT, USA), 2007. (document), 4.2.4

[48] Mazzacone, E., "System f6 to exploit benefits of democratization of innovation." News Release, August 2010. http://www.darpa.mil/WorkArea/DownloadAsset.aspx?id=1798. 1.2

[49] Molette, P., Cougnet, C., Saint-Aubert, P., Young, R. W., and Helas, D., "Technical and economical comparison between a modular geostationary space platform and a cluster of satellites," *Acta Astronautica*, vol. 11, December 1984. pp. 771-784. 1.1

[50] Montenbruck, O., Gill, E., and Markgraf, M., "Phoenix-xns – a miniature real-time navigation system for leo satellites," in *3rd ESA Workshop on Satellite Navigation User Equipment Technologies*, (Noordwijk, The Netherlands), NAVITEC, 2006. 3.1

[51] Montenbruck, O., Nortier, B., and Mostert, S., "A miniature gps receiver for precise orbit determination of sunsat 2004 micro-satellite," in *ION National Technical Meeting*, (San Diego, CA, USA), ON, 2004. 3.1

[52] Persson, S., Bodin, P., Gill, E., Harr, J., and J J. 1

[53] Persson, S., Jakobsson, B., and Gill, E., "Prisma - demonstration mission for advanced rendezvous and formation flying technologies and sensors," in *Proceedings of 56th International Astronautical Congress*, (Fukuoka, Japan), IAF, 2005. 1

[54] Pluym, J. P. and Damaren, C. J., "Second-order relative motion model for spacecraft under j_2 perturbations," in *AIAA/AAS Astrodynamics Specialists Conference and Exhibit*, (Keystone, CO, USA), 2006. 1.3

[55] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in C: The Art of Scientific Computing*. New York, NY: Cambridge University Press, second ed., 1992. 1.5, 2.4.2.1, B

[56] Rupp, T., D'Amico, S., Montenbruck, O., and Gill, E., "Autonomous formation flying at dlrs german space operations center (gsoc)," in *Proceedings of 58th International Astronautical Congress*, (Hyderabad, India), IAF, 2007. IAC-07-D1.2.01. 1

[57] Russell, R. P., "Examples of effects of non-spherical harmonic coefficients.." Georgia Tech AE 6353 lecture notes., August 2009. These images were used with direct permission from the author. (document), 5, 6, 7

[58] Scrofano, R., Anderson, P. R., Seidel, J. P., Train, J. D., Wang, G. H., Abramowitz, L. R., Bannister, J. A., and Borgeson, M. D., "Space-based local area network," in *Military Communications Conference*, (Boston, MA, USA), IEEE, 2009. doi 10.1109/MILCOM.2009.5379944. 4.2.2

[59] Sheth, A. and Han, R. 4.2.1

[60] Sidibeh, K. and Vladimirova, T., "Wireless communication in leo satellite formations," in *NASA/ESA Conference on Adaptive Hardware Systems Proceedings*, IEEE, 2008. AHS.2008.6. 4.2.1

[61] Stein, K., "Darpa releases system f6 program details." News Release, July 2007. http://www.freerepublic.com/focus/f-news/1873020/posts. 1.2

[62] Tapley, B. D., Bettadpur, S., Watkins, M., and Reigber, C., "The gravity recovery and climate experiment: Mission overview and early results," *Geophysical Research Letters*, vol. 31, May 2004. doi:10.1029/2004GL019920. 2.3.1

[63] Tapley, B. D., Schutz, B. E., and Born, G. H., *Statistical Orbit Determination*. New York, NY: Elsevier Academic Press, first ed., 2004. 1.6, 3.2, 3.2.1, 3.2.2

[64] Thomsen, M., "Michaels list of cubesat satellite missions." website, August 2009. http://mtech.dk/thomsen/space/cubesat.php. 1, 2.1, 4.2, 4.2.3

[65] Twiggs, R., Malphrus, B., and Muylaert, J., "The qb50 program, the first cubesat constellation doing science," in *Proceedings of 24th Annual Small Satellite Conference*, (Logan, UT, USA), AIAA/USU, 2010. SSC10-XII-3. 1

[66] Vallado, D. A., *Fundamentals of Astrodynamics and Applications*. Hawthorne, PA and New York, NY: Microcosm Press and Springer, third ed., 2007. (document), 2, 2.1, 2.3.1.1, 2.3.1.1, 2.3.1.1, 2.3.1.4, 2.3.2, 2.3.2, 4, 2.3.2, 2.3.3, 5, 2.3.4, 4

[67] Vivian, "Real-time orbit determination using gps navigation solutions," *Journal of the Brazilian Society of Mechanics, Science, and Engineering*, vol. XXIX, July-September 2007. 1.6

[68] Vladimirova, T., Wu, X., and Bridges, C. P., "Development of a satellite sensor network for future space missions," in *IEEE Aerospace Conference Proceedings*, IEEE, 2008. IEEEAC paper 1263, Version 4. 4.2.1

[69] Winter, J. E. and Anderson, L. N. C., "Distributed aperture implementation on the techsat 21 satellites," in *Proceddings of Aerospace Conference, 2003*, (Big Sky, MT, USA), IEEE, 2003. 1095-323X. 1.2, 6

[70] Zhang, J., Zhang, K., Grenfell, R., and Deakin, R., "On the relativistic doppler effect for precise velocity determination using gps," *Journal of Geodesy*, vol. 80, 2006. pp. 104-110. 3.1