Optimal Collision Avoidance of Operational Spacecraft in Near-Real Time

Final Report of the Spring 2011 Mathematics Clinic

Sponsored by SpaceNav, LLC

Department of Mathematical and Statistical Sciences

University of Colorado Denver

The Clinic Team

Student Participants

- Kannanut Chamsri
- Jeremy Doan
- Rebeccah Dutcher
- Yonas Getachew
- Pamela Hunter
- Michael Kasko
- Volodymyr Kondratenko
- Anna Le
- Danielle Matazzoni
- Austin Minney
- Mark Mueller
- Brian Wainwright

Faculty Advisor

- Alexander Engau

Sponsor Advisors

- Matthew Duncan
- Joshua Wysack

Sponsor: SpaceNav

SpaceNav LLC is a Colorado-based aerospace engineering firm providing technical solutions in the areas of Space Situational Awareness, Systems Engineering, and Mission Operations. Their process-driven approach to problem-solving enables to deliver on-time, error free products and services in a variety of functional areas including Systems Engineering and Integration, Space Situational Awareness, Orbit Analysis Mission Operations, and Advanced Concepts Research and Development Cell.

The SpaceNav team brings technical expertise and experience from a vast number of programs spanning the U.S. Department of Defense (DoD), National Aeronautics and Space Administration (NASA) and National Oceanic and Atmospheric Administration (NOAA) customer base. Leveraging a passion for space combined with in-depth knowledge of space operations, SpaceNav provides cost-effective solutions to a vast array of technical problems.

Welcome to **SpaceNav**!

SpaceNav is a Colorado-based aerospace engineering firm providing technical solutions in the areas of Space Situational Awareness, Systems Engineering, and Mission Operations. Our process-driven approach to problem-solving enables us to deliver on-time, error free products and services.

The **SpaceNav** team brings technical expertise and experience from a vast number of programs spanning the **DoD**, **NASA** and **NOAA** customer base. Leveraging our passion for space combined with our in depth knowledge of space operations, **SpaceNav** is ready to provide cost effective solutions to a vast array of technical problems.

Figure 1: SpaceNav web site at http://www.space-nav.com

Acknowledgments

The math clinic is a key element in the mathematics education at UC Denver and offers our students a unique experience to collaboratively learn, advance, and apply their knowledge and skills to a challenging, real-life problem. However, we could not do it without the exceptional support by our sponsors who commit resources and often dedicate a tremendous amount of their own time to help us, and to help our students. Participating in a clinic for the first time myself, I am most thankful to Matthew Duncan, President of SpaceNav, and his colleague Joshua Wysack not only for making this clinic possible, but also for taking an outstanding role in getting involved, showing students directions, and spurring motivation by demanding results. Like many of the students, I could learn a lot myself and stay indebted to Matt's and Josh's patience, their encouragement, and our shared leadership throughout the full semester. Needless to say, the problem offered is fantastic and remains an active stimulation for further work or collaboration.

The ability to work with our students in such an open and interactive learning environment is easily one of the best aspects that a clinic offers also to faculty. I therefore like to recognize the wide variety of contributions made by the twelve participating students ranging from literature reviews and presentations to the class over independent studies and mutual tutoring to the development or implementation of software and the compilation of this final report. Despite regular guidance by faculty and sponsor advisers, it is ultimately the students' initiative and decision to take responsibility that make a clinic successful, and I congratulate those for whom meeting expectations was just the beginning but never enough.

Finally, I thank my colleague Stephen Billups for his service as Clinic Director and his substantial contributions to the Math Clinic Program in our department. While Steve's repeated recruitment of partners and sponsors keeps this program alive, his genuine interest and habitual willingness to share his experiences and offer advice truly fills it with life. I also like to thank our Department Chair Michael Jacobson for trusting me with this clinic and allowing me to learn along, and to Angela Beale, Lindsay Hiatt, and Russel Boice for their continuous logistical and technical support.

Alexander Engau
Denver, June 2011

Student Acknowledgments to Sponsor

“Thank you for your organization of such an interesting class. It was a great topic and a good experience of teamwork for me.”

Volodymyr Kondratenko

“I would like to thank you for your time and all the support for the math clinic class. Showing up every single class was very impressive; giving us advice, comments and explanations was creating excellent working conditions. Your insight and your thoughtful explanation made a complex problem easy to understand. Without your guidance and mentorship this class would have been impossible.

For me, working on this problem gave me the opportunity to prepare myself for conducting research and strengthening my experience, accumulating more professional knowledge, proficiency and skills.

I wish you and your families comfort on difficult days, faith so that you can believe, confidence for when you doubt, patience to accept the truth, and I wish your company having excellent distribution to provide great profits to the company and the world.”

Kannanut (Mon) Chamsri

Table of Contents

List of Figures

Chapter 1

Introduction

Scientific discoveries over the last centuries and technological breakthroughs during the last few decades have fundamentally changed the way how we organize and live our lives in the 21st century. Ongoing progress and its resulting new challenges impact practically all areas of human activity including medicine and drugs, public policy and service, transportation and energy, engineering and economics, and new computing technologies. Besides private and public research institutes, governments, and many other commercial or non-profit organizations, one of the most successful and influential players is the aerospace industry. Telecommunication and broadcasting; mobile data exchange and wireless transmission; global positioning, geographic imaging, mapping, navigation, and planning; remote sensing, search, rescue, and defense; weather forecasting and prediction; space exploration and astronomy – the list goes on and on of what we depend on space for, and what particularly satellites provide for us on a routinely and often daily basis.

In consequence of the increasing exploration and exploitation of space, already today there are more than 2,000 active satellites orbiting our planet, about half of which at the relatively small altitude of 500 to 600 miles (for comparison, the Earth's mean radius is approximately 3,958.76 miles, or 6,371 kilometers). Since the launch of the first satellite Sputnik I by the Soviet Union on October 4, 1957, according to the National Space Science Data Center (NSSDC) Master Catalog a total of at least 6,642 spacecrafts were launched into orbit as of May 2011, twenty thereof in 2011 with the last one on April 26 (Progress M-10M) [22]. Soon, on June 10, they will be joined by yet another: a new NASA Aquarius/SAC-D satellite is scheduled for a joint U.S./Argentinian Aquarius/Satélite de Aplicaciones Científicas

(SAC)-D mission that will map the salinity (concentration of dissolved salt) at the ocean surface and improve our understanding of two major components of Earth's climate system: the water cycle and ocean circulation.

Fascinated by such satellite missions and in view of the vastness and perceived emptiness surrounding our planet, it is easy to overlook the question whether there is actually sufficient space for yet another satellite. In fact, although there is still plenty of room for many more, an increasing amount of space debris from previous missions has grown to pose a significant threat to the safety of modern spacecraft operations. In addition to the current estimate of about 22,000 trackable objects with a size of a golf ball or larger, including discarded launch equipment or inactive satellites, there is an estimated number of roughly 350,000 extra pieces of fragments of broken up spacecraft (such as nuts or bolts), equipment lost by astronauts (such as screw drivers or gloves), or other debris from past collisions that are traveling with relative velocities of up to 15 kilometers per second [7, 11]. Hence, because of speed and size, respectively, the avoidance of spacecraft collisions with these objects is of crucial importance but can become tricky and quite challenging. The artwork in Figure 1.1 from the Science Photo Library of the European Space Agency (ESA) alarmingly depicts this situation [12].

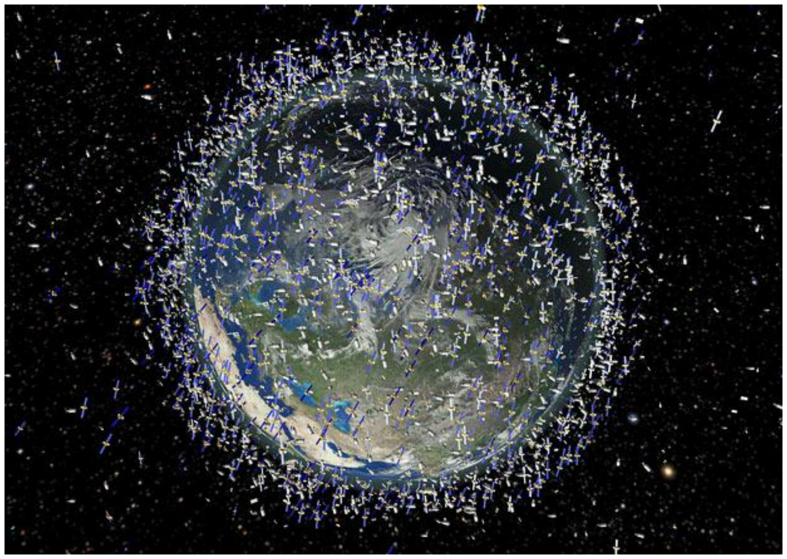


Figure 1.1: Artwork showing space debris in low and geostationary Earth orbit (based on density data, but not to scale)

While this problem is known and discussed among industry practitioners for a long time, two major debris-generating events during the last five years have noticeably increased the concern in both the aerospace industry and national governments. First, during a Chinese Anti-Satellite Missile Test on January 11, 2007, the Chinese government intentionally destroyed one of their own polar-orbiting weather satellites (Fengyun 1C) at an altitude of 537 miles (865 kilometers). Although the test was done in secret, it was later announced publicly and until today remains that single event that produced the largest amount of space debris in history, with a cataloged number of 3,135 pieces of debris as of May 20, 2011 and an estimated number of more than 150,000 pieces larger than 1 cm according to the NASA's Orbital Debris Program Office [16]. Two years later, on February 10, 2009, the U.S. Iridium 33 communication satellite collided with the Russian Cosmos 2251 satellite at an altitude of 490 miles (789 kilometers). Whereas the Iridium satellite was an operational part of a 66 satellite-constellation for the company, the Cosmos satellite had been inactive for nearly 10 years and was non-operational without a working maneuver system. Although both satellites had been tracked without indication of a critically close encounter, at the time of closest approach the Iridium went silent and only after a few minutes a collision became apparent. This was the first time in history that two satellites collided in orbit, and as of May 13, 2011 the U.S. Space Surveillance Network (USSSN) has cataloged 547 pieces of debris (47 pieces of which have already decayed from orbit) associated with Iridium 33 and 1,488 pieces of debris (99 pieces of which have decayed) associated with Cosmos 2251 [15].

Because any debris produced subsequently endangers other pieces of equipment, that often cost billions of dollars to build and launch, there is a clear interest by the aerospace industry and commercial satellite operators in keeping any more of these events from happening. In effect, collision risk management is now standard for all space operations, and close-approach predictions of spacecrafts with other satellites or tracked debris are now published routinely every day. Additional initiatives are also taken on the political stage, and improved data sharing agreements are currently being developed between commercial satellite operators together with the U.S. and other international organizations. In particular, according to the goals laid out in the 2010 U.S. Space Policy, "domestic and international measures to promote safe and responsible operations in space; improved information collection and sharing for space object collision avoidance; protection of critical space systems and supporting infrastructures, with special attention to the critical interdependence of space and information systems; and strengthening measures to mitigate orbital debris" are now a central focus of the U.S. national space programs to strengthen stability in space [23].

From an industrial, operational perspective, collision risk management and collision avoidance (COLA) is a rather complex process that typically involves at least the following four major steps (compare Figure 1.2):

1. Identification of all conjunction events over some future time span;
2. Assessment and quantification of the collision risk for each event;
3. Evaluation and action/no-action decision from satellite operators;
4. Development and execution of collision avoidance maneuvers.

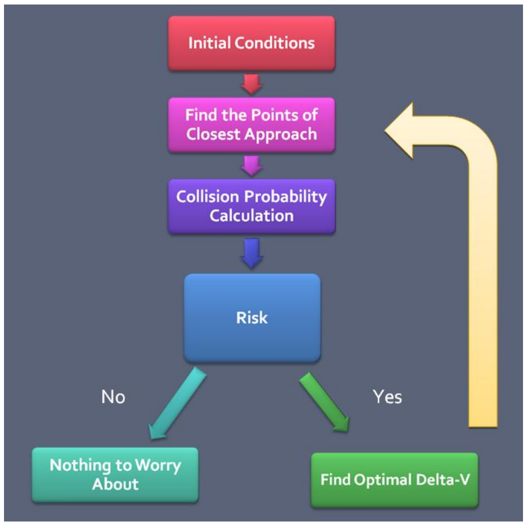


Figure 1.2: Generic decision flow chart for collision risk management

In the above list, we typically refer to a conjunction between two objects when their separation or difference in position violates an a priori specified miss distance tolerance threshold value. Given the high velocities of these objects together with the large distances from where their positions can be measured, and considering the multiplicity of possible conjunction events at any given point of time or time horizon, however, an optimal prediction, evaluation, and reaction is faced by several severe challenges:

- Conjunction events are inherently uncertain and subject to both tracking inaccuracies and numerous other numerical or computational errors.

- The number of possible conjunction events grows exponentially with the size of the space object catalog and the number of tracked objects.
- A successful collision avoidance maneuver for one conjunction may lead to new conjunction events and increase the likelihood of other collisions.

Thus motivated, the goals of this clinic are to mathematically describe and model the process of collision avoidance in an attempt to address the urgent need of better planning or execution tools and methods. Specifically, this report outlines an operational strategy that determines and reduces the risk of collisions between a primary satellite and other spacecrafts or space debris by altering the satellite's trajectory. The ultimate objective of this effort is to assist both researchers and practitioners in formulating tractable optimization models and developing algorithms for this problem, and to make recommendations for collision avoidance maneuvers in the future.

1.1 Overview and Conduct of Clinic

Despite the fact that mathematics clinics at UC Denver are offered as part of the regular curriculum, these courses are typically run as much as possible like real-world projects and focus around activities and initiatives led by the students. As a consequence, most clinics have no or only very few lectures at the beginning of the term, whereas most of the class time is then spent working in individual groups to exchange, develop, and pursue new ideas and learn how to effectively communicate strategies and results to each other. For many students, this is the first opportunity to tackle a challenging and important real-life problem whose solution is not to be found in a textbook but only in their own deep analysis and understanding. Hence, students typically do not only learn new mathematics and other subjects related to the specific clinic topic, but especially successful students will also:

- gain experience in applying math to a challenging real-world problem;
- be exposed or get involved in an interdisciplinary industrial project;
- improve their oral and written (technical) communication skills;
- develop their (soft) project management and teamwork skills.

In addition to these educational objectives, of course the final clinic outcome is equally important and should demonstrate a solid understanding and a clear step toward an acceptable solution of the originally posed problem. To ensure the success of a clinic, this means that students, faculty, and sponsors often need to work together as equal partners in the project activities. It is important to emphasize, however, that a clinic is not intended to compete with the private sector for providing professional consulting services or for producing high-end software applications, but to offer a fresh look paired with mathematical expertise in order to think up innovative solution strategies and possibly prototype a couple of new approaches or ideas.

1.1.1 Project Warmup (January-February)

The final Spring 2011 clinic team consisted of 12 students (8 undergraduates and 4 graduates) who where joined by one member from the UC Denver math faculty and two advising representatives from the clinic sponsor SpaceNav (also compare pages ii and iii). Over a period of sixteen weeks, between the initial meeting on January 19 and the final project presentation on May 10, we met twice a week excluding the week of spring break. During the first two weeks (the last two weeks in January), the students were given an introduction to the class and clinic concept, a problem description by the sponsor, and two faculty lectures on computing collision probabilities using 2D/3D integration and Monte Carlo simulations, and times of closest approaches between moving objects using optimization based on several journal articles and technical reports [1, 2, 3, 6, 17, 29]. At the same time, students started to review other basic literature and were given small writing and two homework assignments (see Appendix B) to independently learn or refresh their working knowledge of the mathematical software system MATLAB [25] and the text-processing system LaTeX [18] which had to be used for all homework assignments and other written materials in the clinic.

1.1.2 Phase 1 Projects (February-March)

Based on sponsor input, at the beginning of February students were separated into five groups to prepare important background topics and acquire certain mathematical prerequisites. Each group was assigned the same set of general goals and objectives:

- to collect, learn, summarize, and explain all major findings to the other students in the class (in both written and oral form);

- to identify aspects with opportunities for improvement and original research (that could later lead to Phase II projects);
- to put their new knowledge and expertise to service for others throughout the rest of the semester and project.

More specific tasks and key take-away points were customized to each individual group by the sponsor:

1. Propagation of Error and Uncertainty (Anna, Danielle, Pam)
   - Understand state errors, acknowledge that representation of errors can be in the form of a covariance matrix.
   - At close approach point, draw two error ellipsoids (note overlap).
   - Generate a covariance time history via Monte Carlo simulation.
   - Convert a covariance matrix from R-I-C frame to Cartesian frame.
2. Analytic Representation of Orbital Trajectories (Volodymyr, Yonas)
   - Conversion of Cartesian to Keplerian element set.
   - Examine time history to observe period behavior.
   - Create Matlab script that takes as an input a Cartesian time history file; output time history of Keplerian file (make plots, etc).
   - Explore different analytic representations for Keplerian element sets. How do we measure the error between the original set and the analytic representation? This needs to be sufficiently close.
3. Computation of Collision Risk Probabilities (Mark)
   - Develop understanding between Monte Carlo, 3-D and 2-D approaches to computing collision probability.
   - Create Matlab script that computes the 2-D collision probability.
   - Draw the combined covariance matrix in the conjunction plane.
4. Trajectory Changes and Delta-V Maneuvers (Austin, Beccah, Mike)
   - Hohmann Transfer describes an in-plane orbit change (assumes no other perturbations to the orbit).
   - Generic orbit change in the V-N-B frame.

- Modify homework M-files to a script that does the propagation, performs a delta-V maneuver, and then does more propagation.
- If you have a range of delta-Vs that you want to apply to a nominal trajectory, modify above to loop ... and then come up with a way to collect and present the output.

5. Finding Points of Closest Approach (Brian, Jeremy, Mon)

   - Create a Matlab script that takes as inputs two time history files and outputs all close approach points for a (user specified) separation distance. Does your algorithm find all minimums and then only report the ones that violate the user specified threshold? Is there a better, more efficient way to do this?
   - How would you modify your script if you had to find separation distances for multiple time history files ... what if you only had analytic expressions (vs time history files)?

The written outcomes from each of these groups are reported in Chapter 2.

1.1.3 Phase 2 Projects (March-April)

Based on those outcomes successfully achieved by the beginning of March, for the second phase we decided to continue with only three groups focusing primarily on mathematical models and optimization strategies to find optimal trajectory changes (Brian, Mark, and Yonas with help by Anna and Danielle), the implementation of a software tool to compute and either manually or automatically change satellite trajectories (Jeremy, Mike, Mon, and Volodymyr), and a writing team who facilitated the collection of materials (Austin and Beccah) and the compilation of the final report (Pam). All teams worked largely independent of each other and received regular input and feedback from faculty and sponsor advisers.

Following the original clinic concept, students were initially left with significant freedom and independence to determine and explore new approaches, take on responsibility, and also develop initiatives on their own. After an introductory faculty lecture on basic principles of optimization models and formulations, the optimization group combined knowledge from several Phase 1 projects (collision probabilities, trajectory changes, and closest approach points) to investigate two existing approaches for trajectory optimization from the literature [21, 24]. The mathematical derivation in Chapter 3 gives a nice account of this study undertaken by these students.

Following feedback from the sponsor, the coding group directed its main focus in Phase 2 on finishing and improving the computational tasks and results from Phase 1 and writing several new program codes in MATLAB, including routines to compute analytic approximations of satellite trajectories, the propagation of space objects based on interpolation between observed states, and the enhanced computation of closest approach points (for a full list, see Appendix C.2). Without a particular emphasis on new mathematical developments, the explanations in Chapter 4 are kept at a relatively general level and give a mostly verbal overview of some of the mathematics underlying the actual code, that is fully given in Appendix C.2.1.

1.1.4 Project Closeout (May)

After an initial draft of this report was delivered by the writing team at the end of April, it was distributed, proofread and collaboratively edited by the full class. All major initiatives were completed during the first week of May, in which two summarizing presentations were given to collect concluding feedback from the sponsor. The final presentation was rehearsed in the morning of Tuesday 10th by Anna, Beccah, Mike, Pam, Volodymyr, and Yonas, and given as part of the Operations Research Seminar on the same day by Austin, Beccah, Brian, Danielle, Jeremy, and Mon. The presentation slides can be requested by e-mail from `alexander.engau@ucdenver.edu`. The final report was subsequently updated and eventually finished in the middle of June.

1.2 Outline of Report

The objective of this report is to provide a detailed account of the materials we learned and the work we accomplished throughout the Spring 2011 clinic. As indicated by the clinic overview in Section 1.1, the basic structure of this report follows the two main phases providing some fundamental background in Chapter 2, and summarizing activities by the optimization and coding groups in Chapters 3 and 4, respectively. Based on some of these results, Chapter 5 presents a small case study using data provided by our sponsor SpaceNav. Concluding remarks are offered in the final Chapter 6 and followed by our references and some supplemental materials in the appendices.

Chapter 2

Background

This section describes the topics we learned throughout the two phases of our project. We begin with a brief overview of satellites and orbits in Section 2.1, followed by three major sections on some orbital mechanics fundamentals in Section 2.2, a description of different coordinate frames and possible trajectory representations in Section 2.3, and a sketch of different ways to represent and deal with the uncertainty in determining exact satellite positions and velocities, and computing collision probabilities in Section 2.4.

2.1 Satellites and Orbits

To better understand the need of collision avoidance in practice, it is useful to address how satellites are distributed in the space around our planet. Obviously, each satellite will be launched at an altitude that matches its intended functionality. For instance, the satellites that operate our cell phones are in so-called geosynchronous or geostationary orbits. Geosynchronous orbits are typically used for satellites at an altitude of approximately 35,800 kilometers that have the same orbital period as the Earth. These circular orbits sit on top of a specific point on the Earth and rotate along with us, which allows the satellite to maintain its fixed relative position and provide constant coverage. There are currently about 300 active satellites in geosynchronous orbit. Because of the associated space limitations of this type of orbit, there are international regulations governing the assignment of individual longitude slots. In addition to communication satellites, this is also a good orbit for many weather satellites like GOES and METEOSAT. From this altitude, one satellite can provide weather coverage for almost an entire hemisphere [20].

The second type of orbits are the so-called Low Earth Orbits (LEO) that correspond to an altitude of about 300 to 1500 kilometers. This altitude has the highest concentration of satellites. Below the altitude of 150 kilometers, an orbit will decay back into the Earth's atmosphere. For example, the Hubble Space Telescope is in a low Earth orbit at 600 kilometers and can operate from an unobstructed view. In low Earth orbits, the period can vary with altitude. For example, a satellite at 900 km will orbit the earth 14 times in a day. The low Earth orbits can be further divided into three sub-categories:

- equatorial orbits near the Earth, where the satellite orbits the equator at various distances and eccentricities (the elliptical shape of the orbit);
- polar orbits, where the satellite orbits the north and south poles at various eccentricities and altitudes;
- specialty orbits, which can include spy satellites and other types of satellites with special functions.

The density distribution of objects in orbit is indicated in Figure 2.1 [4]. Figures 2.2 and 2.3 depict similar information and were provided by Matthew Duncan from our sponsor SpaceNav.

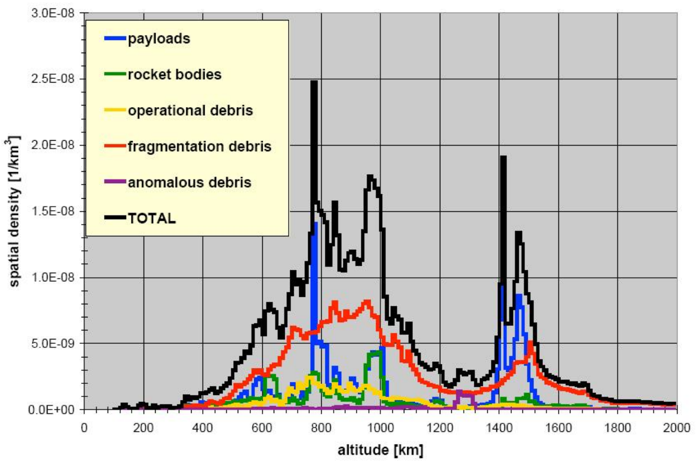


Figure 2.1: Spatial density of equivalent satellite objects in low earth orbits

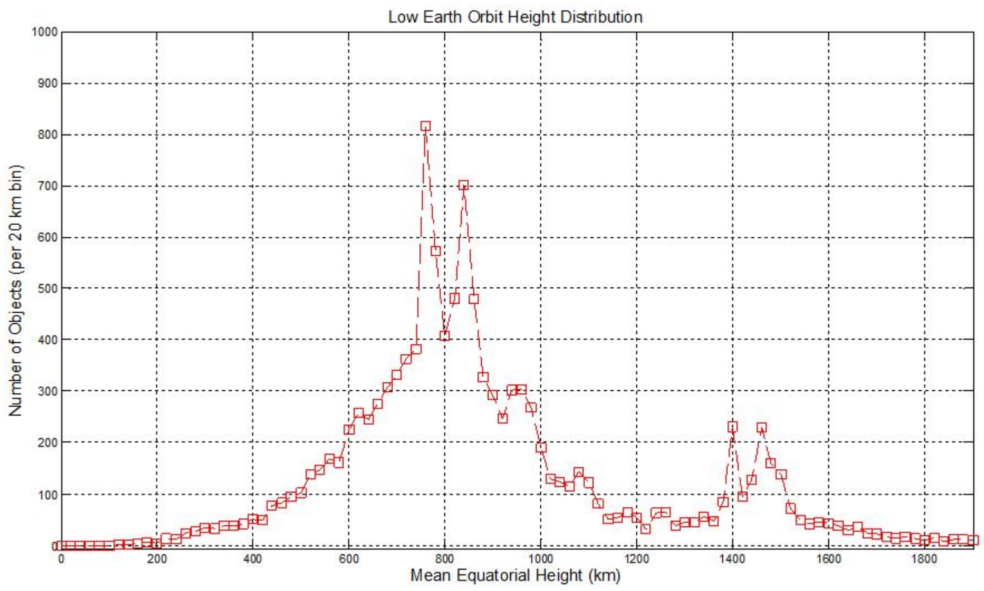


Figure 2.2: Spatial density of cataloged space objects in low earth orbits

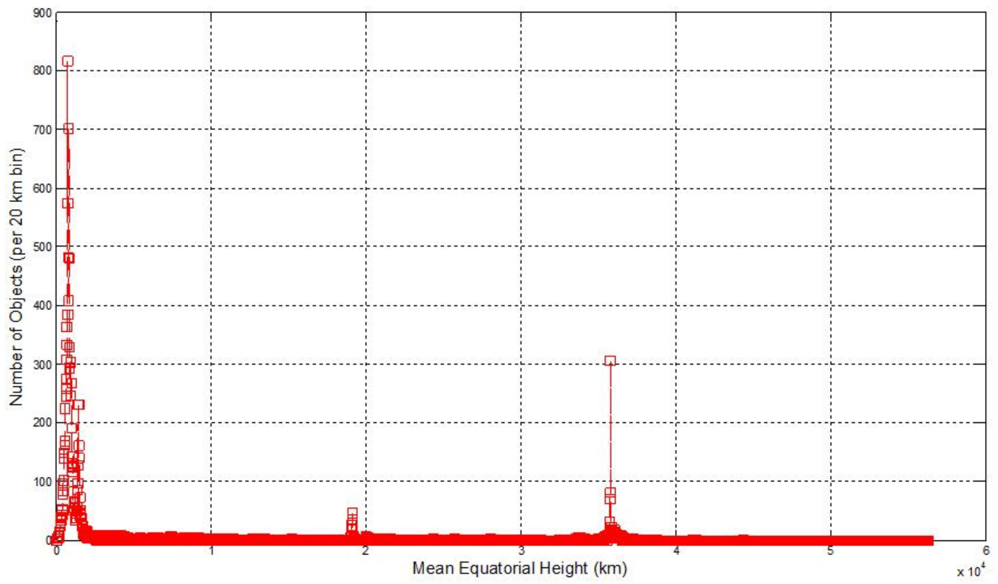


Figure 2.3: Spatial density of cataloged space objects in total

There are several methods for tracking satellites in these different orbits. Because satellite operators usually control satellites from a ground basis, constant contact with the satellites is hugely important. There are a number of tracking systems that can be used to find the instantaneous position of a satellite and its rate of change. The majority of these systems uses radio signals transmitted from a ground antenna. These radio tracking systems perform angle measurements by locating the direction of a radio signal that is transmitted by the satellite. The turn-around delay or Doppler shift of a radio signal that is sent and returned from the satellite can give both distance and velocity information. If the satellites do not have an active transmitter, then a sufficiently powered radar can be used for tracking. Because of complications with ground-based tracking of LEO satellites, geostationary satellites can be used to track another satellite via a relay transponder. It is also possible to use GPS ranging signals to find position measurements on board of a satellite without using a ground station at all. Another way of tracking is to use optical sensors. Imaging telescopes can be used for finding unknown spacecraft and debris as far out as geostationary distances. Satellite laser ranging systems are very useful and obtain very accurate distance measurements by measuring the time it takes for a laser pulse transmitted to the satellite to be returned. These devices are mostly used for scientific and geodetic missions that need very high precision [20].

2.2 Fundamental Orbital Mechanics

This section revisits some orbital mechanics fundamentals and lays out the basics for planetary motion and its relationship to the motion of other, artificial space objects such as spacecrafts and satellites. Classic references for this topic are the monographs by Bate, Mueller and White [5], Prussing and Conway [26], Montenbruck and Gill [20], and Chobotov [10], among many others. We begin our own attempt to explain how satellites move using the laws of motion by Issac Newton and Johannes Kepler.

Newton's laws of motion are the governing ideas behind all of orbital mechanics, specifically the second and third law.

Newton's Second Law: The sum of the forces acting on an object is equal to the mass of the object times the acceleration of the object, which is also equal to the rate of change of the momentum of the object:

$$\vec{F} = m\vec{a} = \frac{d\vec{p}}{dt}. \tag{2.1}$$

This equation is useful for both laws of gravity and for rocket propulsion.

Newton's Third Law: For every action there is an equal and opposite reaction:

$$\vec{F}_{12} = -\vec{F}_{21}. \tag{2.2}$$

Kepler came up with the laws of planetary motion in the 16th century. Kepler mostly concentrated on planetary motion around the sun, but especially his third law is useful for any orbital motion.

Kepler's Third Law: The square of the orbital period of an object in orbit around a much more massive object is proportional to the cube of the distance between the two objects:

$$P^2 = kr^3 \tag{2.3}$$

where k is a constant of proportionality.

The above laws can be combined by a simple matter of substitution and rearrangement. If we write the period in terms of the orbit's circumference and assume a constant velocity of an object with mass m, we get that

$$P = \frac{2\pi r}{v}. \tag{2.4}$$

If we square this value for P and use Kepler's law, we obtain a value that is equal to the centripetal force for circular motion:

$$\frac{4\pi^2 r^2}{v^2} = kr^3. \tag{2.5}$$

Substitution into the former equations gives

$$m\frac{v^2}{r} = \frac{4\pi^2 m}{kr^2} \tag{2.6}$$

which can equivalently be written as

$$F = \frac{4\pi^2 m}{kr^2}. \tag{2.7}$$

This implies that the centripetal force keeping an object of mass m in orbit about a larger object (of mass M) is equal to right side of the last equation. By symmetry, the force must also be the same for the larger object M. Regrouping and combining terms, we find that

$$F = \frac{4\pi^2 Mm}{kr^2}. \tag{2.8}$$

This equation gives a value for the force of gravity between two objects. If we group the constants together, we obtain the universal gravitational constant

$$G = \frac{4\pi^2}{k} \approx 6.673 \times 10^{-11} \text{ N m}^2 \text{ kg}^{-2}. \tag{2.9}$$

Hence, the force of gravity between two objects of masses M and m is

$$F = \frac{GMm}{r^2}. \tag{2.10}$$

In order to apply this derivation to satellites in orbit, we only need to substitute the mass of the satellite and the mass of the earth, and let the distance r equal the radius of the earth plus the altitude of the satellite. The resulting law provides much of the information needed when working with satellites.

2.2.1 Planetary Motion and Two-Body Equation

Writing the scalar equation (2.10) in vector form, we recover Newton's Law of Gravitation:

$$\vec{F} = \frac{-GMm\vec{r}}{r^3} \tag{2.11}$$

where $\vec{F}$ is the force of gravity, G is the universal constant of gravitation, M and m are the mass of the objects, and $\vec{r}$ is the position vector of magnitude $r = \|\vec{r}\|$. The quantity

$$\ddot{r}_s = \frac{-GM_e\vec{r}}{r^3} \tag{2.12}$$

is defined as the acceleration vector for the satellite, and

$$\ddot{r}_e = \frac{-Gm_s\vec{r}}{r^3} \tag{2.13}$$

is the acceleration vector of the Earth. The combined acceleration vector

$$\ddot{r} = \frac{-Gm_s\vec{r}}{r^3} + \frac{-GM_e\vec{r}}{r^3} \tag{2.14}$$

is the pull of the two objects on each other. To write this acceleration as a system of first order equations, first we define the velocity vector r as s. Then we define the derivative s as the second derivative of the position vector r. Now, if we define s in terms of the position vector r, we can write

$$\begin{aligned} \dot{r} &= s \\ \dot{s} &= \frac{-\mu_e\vec{r}}{r^3} \end{aligned} \tag{2.15}$$

Here, the acceleration vector is defined by the position vector times the scalar $-\mu_e$ divided by the cubed radius. This is a system of six first-order differential equations where the first three equations are given in terms of the (x, y, z) components of the velocity vector, and the last three equations are given in terms of the acceleration vectors. Using this system of equations, we can easily solve the above differential equation, say by using the ode45 solver in MATLAB as done for Homework 1 (compare Appendices B and C.2.4). Running this solver produces time, position, and velocity of these objects in space. Effectively, this is an approximation of the trajectory.

2.2.2 Motion in Space: Spacecrafts and Rockets

This section shows a derivation of the governing equation for the motion of a spacecraft or rocket. Rockets work by conservation of linear momentum: if there are no "outside" forces, then linear momentum is conserved and does not change over time. The linear momentum formula is

$$\vec{p} = m \cdot \vec{v} \tag{2.16}$$

where $\vec{p}$ is linear momentum, $\vec{v}$ is velocity, and m is mass. In the context of rockets, gravity is an outside force, but for simplicity we will ignore any gravitational forces on the rocket.

Mass

Rockets burn fuel in order to accelerate, and the rocket's mass decreases as the fuel is burnt. Let

$$m(t) = \text{mass of the rocket, including fuel, at time } t \tag{2.17}$$

and

$$\begin{aligned} dm(t) = {} & \text{change in the rocket's mass during the time interval} \\ & \text{from } t \text{ to } t + dt \text{ due to the decrease in fuel.} \end{aligned}$$

Recalling the definition of change as a new value minus an old value, one can write

$$dm(t) = m(t + dt) - m(t) < 0. \tag{2.18}$$

It is important to remember that $dm(t)$ is always negative.

Velocity of the Rocket

Let $\vec{v}(t)$ and $v(t)$ respectively be the velocity and the speed of the rocket at time t, relative to an observer on the ground.

Velocity of the Rocket's Exhaust

Let w be the speed of the exhaust relative to the rocket. For simplicity, we will make two assumptions about the rocket's exhaust:

1. The direction of the exhausted is opposite the direction of motion of the rocket. This, for example, is the situation when a rocket is first launched.
2. The speed of the exhaust, relative to the rocket, is constant.

Relative Motion

The exhaust's simple motion hides an important but subtle point which we now investigate. If, for example, the rocket is moving straight up, its exhaust moves straight down. To be specific, let's say the rocket is moving upward at 1,000 meters per second relative to an observer on the ground, and its exhaust is moving down at 100 meters per second relative to the rocket. Then the motion of the exhaust relative relative to a observer on the ground is 900 meters per second upward. Even though the exhaust is always pointed down, the rocket and its exhaust are moving upward.

Equation of Motion

The motion of our simple rocket is determined by conservation of linear momentum. The momentum of the rocket at time t is the same same as the momentum of the rocket at time $t + dt$ plus the momentum of the fuel burnt during the time period t to $t + dt$. The momentum of the rocket at time t is

$$m(t) \cdot v(t). \tag{2.19}$$

The momentum of the rocket at time $t + dt$ is

$$(m(t) + dm(t)) \cdot (v(t) + dv(t)) \, . \tag{2.20}$$

Now recall that $dm(t)$ is negative. The momentum of the fuel burnt during the time period t to $t + dt$ is

$$dm(t) \cdot (v(t) - w) \, . \tag{2.21}$$

By conservation of linear momentum

$$\begin{aligned} m(t) \cdot v(t) = \; & (m(t) + dm(t)) \cdot (v(t) + dv(t)) \\ & - dm(t) \, (v(t) - w) \end{aligned} \tag{2.22}$$

yielding

$$\begin{aligned} m(t) \cdot v(t) = \; & m(t) \cdot v(t) + m(t) \cdot dv(t) + dm(t) \cdot v(t) + dm(t) \cdot dv(t) \\ & - dm(t) \cdot v(t) + dm(t) \cdot w. \end{aligned} \tag{2.23}$$

2.2.3 Trajectory Changes by Hohmann Transfers

In order to avoid conjunctions and potential collisions between space objects, at times we may need to re-maneuver spacecrafts or satellites using certain trajectory changes caused by extra fuel burns; see Figure 2.4 [24].

The idea behind Hohmann transfers between elliptical orbits is to move a satellite from its current orbit to another orbit. It is named after Walter Hohmann (1880-1945), a German engineer interested in interplanetary space travel who, after reading about it in science fiction books, began to study orbital dynamics. In his book *The Attainability of the Heavenly Bodies* [13, 14], Hohmann outlined many of his findings about interplanetary space travel including an approach to optimize the increase of energy that a spacecraft

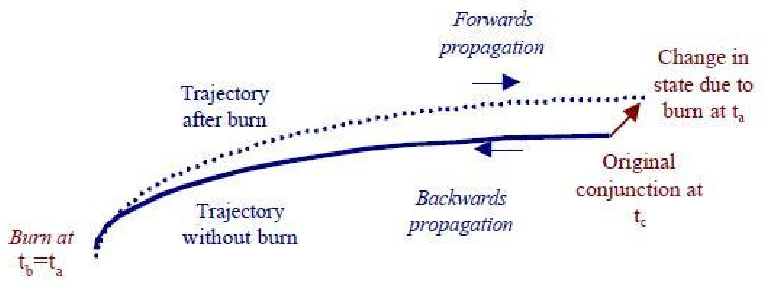


Figure 2.4: Illustration of a single-burn maneuver at $t = t_b$ and the corresponding trajectory change to avoid an original conjunction at $t = t_c$

needs in order to change from its current orbit to another. Because this increase in energy is typically caused by a physical change in velocity, such maneuvers are also called delta-v maneuvers and denoted by Δv. Whereas the motivation to Hohmann was to send a satellite from an orbit around Earth to another planet, our own motivation is to move a satellite from a collision orbit to another, preferably non-collision orbit. Thus, a discussion of delta-v maneuvers is important for addressing possible solution methods investigated in this project.

The Hohmann transfer lays a groundwork for the orbital mechanics of satellites. Because Hohmann was studying orbital mechanics before actual space travel had been attempted (recall that the first satellite in space was Sputnik I in 1957), he imposed several fundamental assumptions that are still standard within the study of celestial mechanics. Before deriving the equations that are necessary to explain a Hohmann transfer, we summarize and explain the three major assumptions that the maneuver is built upon:

1. We assume Newtonian Gravity so that we consider the Earth to be a point-mass system. This is a common assumption within the general study of orbital mechanics and only rarely changed to alternatives systems one might be working within for some other specific problem.
2. We assume that both of our two impulse velocity changes (our satellite "burns") are instantaneous. This assumption is typically satisfied as long as the burns are under a minute, which is often the case. For accuracy within the specific system that one might be studying, the

delta-v impulses become exact. The Hohmann transfer simply makes a generalization about the velocity changes necessary for orbit transfer.

3. We assume that the orbits of the satellites and other space vehicles or debris are co-planar. If the satellites in question are not co-planar, then the equations become much more difficult and a different delta-v maneuver becomes necessary for the satellites in question. Having co-planar orbits, on the other hand, allows for studying orbit transfer without the need of additional dimensions which keeps the algebra relatively simple.

Clearly, we know in many cases that these assumptions are not perfectly true. In view of the first assumption we know that the earth is nowhere near a perfect point mass: it has a varied surface, and we know that there are much more complicated gravity models for the earth. In view of the second assumption, we know that velocity changes cannot be instantaneous, so there is an error involved with this calculation. However, we know that if the velocity transfer is fast, say less than a minute, then the error involved in calculation is close to zero. Finally, in view of the third assumption, our experience with these problems is that satellites are rarely ever co-planar, but rather inclined toward each other which escalates the amount of calculation involved. Some other aspects that violate the above simplifications and may render the computation of Hohmann transfers questionable are continuous thrust, and situations where more than two delta-v maneuvers are needed. More subtle deviations may also arise from other gravitational forces from the moon or sun causing drag or other non-measurable effects. Although such forces may not violate our assumptions completely, over time they may allow for sufficient error to build up and eventually become significant.

Despite these limitations, there is a reason that the Hohmann transfer is still relevant and used to this day. First, it is one of the simplest explanations to understand orbit transfer. Hence, by looking at a rather simple Hohmann transfer we can develop an idea of the orbital mechanics necessary to alter a satellite's trajectory and subsequently avoid satellite collisions. This gives a useful foundation on which further progress in this clinic can be based.

Let us now suppose that the above assumptions are satisfied, and let

$$F = \frac{\mu m}{r^2} \tag{2.24}$$

where F denotes force, $\mu \cdot m$ is the constant for $G \cdot M \cdot m$ which is the product of gravitational constant with mass M of the Earth and the mass m of the satellite, and r is the distance between the two masses. Assumptions 1 and 2 imply that the orbits are Keplerian and correspond to conic orbits. Moreover, under Newtonian gravity which applies to the satellite in question, we know that the total mechanical energy as well as all angular momentum will be conserved. Hence, we can solve for the delta-v maneuvers by manipulating the energy equations.

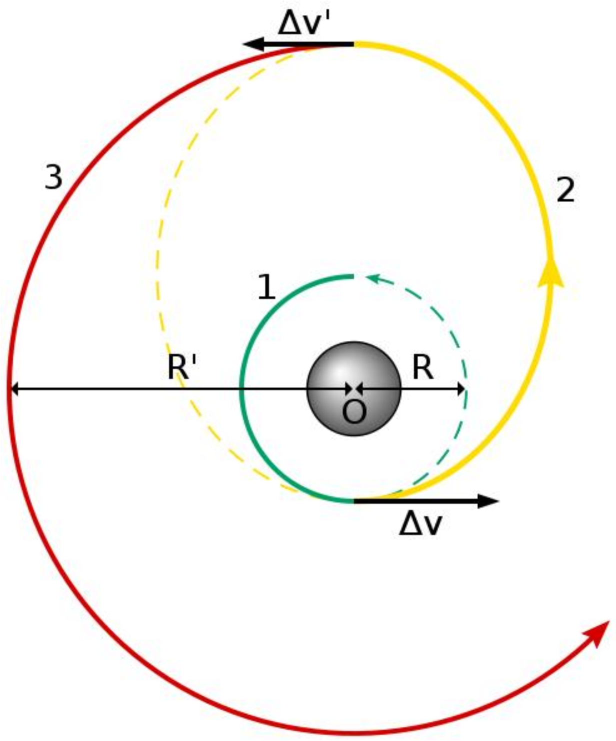


Figure 2.5: Illustration of a Hohmann transfer to change satellite trajectories

Compare Figure 2.5. The Hohmann transfer starts from the initial circular orbit of radius R_1, where we know that its circular speed is given by

$$V_1 = \sqrt{\frac{\mu}{R_1}}. \tag{2.25}$$

Because we want to end on a final circular orbit of radius R_2, we can compute the required circular speed as

$$V_2 = \sqrt{\frac{\mu}{R_2}}. \tag{2.26}$$

Now, to derive the Hohmann transfer equation for elliptical orbits, we let the transfer be achieved by performing a thruster burn at R, for some unknown velocity change ΔV. This maneuver places the spacecraft on new elliptical orbit with a corresponding perigee speed

$$V_p = V + \Delta V. \tag{2.27}$$

Analogously, the resulting apogee speed is

$$V_a = V' + \Delta V'. \tag{2.28}$$

Observe that when the apogee is reached, a second $\Delta V'$ maneuver must be performed to re-circularize the orbit at R' to prevent the satellite from falling back down again.

Now, to solve equations (2.27) and (2.28) for ΔV and $\Delta V'$, we need to determine the transfer orbit. In order to do this we use our assumptions of conservation of angular momentum, and conservation of total mechanical energy.

Conservation of angular momentum

First, by conservation of angular momentum, we know that

$$R \cdot (V + \Delta V) = R' \cdot V' \tag{2.29}$$

which can be solved for V':

$$V' = \frac{R}{R'} \cdot (V + \Delta V). \tag{2.30}$$

Conservation of total mechanical energy

Second, by conservation of total mechanical energy, we know that the term

$$\frac{1}{2}mV^2 - \frac{\mu}{R} \tag{2.31}$$

is constant. Hence, after setting the energy at the apoapsis equal to the energy at the periapsis, we can find the solution for the initial ΔV:

$$\Delta V = \sqrt{\frac{\mu}{R}} \left(\sqrt{\frac{2R'}{R' + R}} - 1 \right). \tag{2.32}$$

Similarly, the $\Delta V'$ solution is

$$\Delta V' = \sqrt{\frac{\mu}{R'}} \left(1 - \sqrt{\frac{2R'}{R' + R}} \right). \tag{2.33}$$

These delta-v equations give us the change in velocity in order to "jump" from one elliptical orbit to another and form one possible basis for our ultimate goal to build a program that can optimally alternate the trajectories of satellites that are on a known, close collision path. In continuation of this idea, the approaches described in Chapter 3 propose techniques to actually define the minimally required delta-v to calculate an in some sense optimal trajectory change when presented with an obstacle.

2.3 Coordinate Frames and Analytic Closed-Form Representations

It is often convenient to represent satellite positions and trajectories in one of several available coordinate frames.

Cartesian/Earth-Centered Inertial (ECI) Frame: The Cartesian system or Earth-centered inertial (ECI) frame has its origin at the center of the Earth. It uses the common (x, y, z) elements of the satellite's position, and the $(\dot{x}, \dot{y}, \dot{z})$ elements of its velocity, where the x-axis points toward the vernal equinox, the y-axis points along the Earth's axis of rotation, and the z-axis is perpendicular to the x and y axes. The term "inertial" means the law of inertia holds. The implication is the coordinate system does not rotate with the Earth, but it is fixed with respect to the stars.

Keplerian Frame: The Keplerian frame also uses sets of six elements that are denoted by $(a, e, i, \Omega, \omega, M)$ and described in more detail in Section 2.3.1 below.

Radial/In-Track/Cross-Track (RIC) Frame: Calculations involving a single satellite often use a coordinate system in which the satellite is the origin. One axis points in the radial direction, that is from the center of the Earth toward the satellite. This is called the radial axis. A second axis, called the in-track axis, points in the general direction of the satellite's velocity vector but perpendicular to the radial axis. (The velocity typically is not perpendicular to the radial axis.) The third axis is orthogonal to these axes and called the cross axis. This coordinate system is called the radial-in-track-cross (RIC) frame.

Encounter Frame: Finally, calculation that involve the encounter of two satellites also use a fourth coordinate system based on the satellites' relative positions and velocities. One of the satellites, the "primary", is chosen as the origin. The y-axis points along the direction of their relative velocity. The plane through the origin, perpendicular to the y-axis is known as the "encounter plane." The secondary satellite will pass through the encounter plane at some point (unless there is a collision). The x-axis points from the origin to that point. The z-axis is orthogonal to the other two axes. This coordinate system is called the encounter frame.

In the following section, we briefly describe how to convert the position and velocity of a satellite given in the ECI (Cartesian) frame to Keplerian element sets and vice versa. A useful reference for this is given by Shapiro [27]. The RIC frame and the encounter frame will become important later, for the discussion of collision probabilities in Section 2.4.2.

2.3.1 Conversion From Cartesian Coordinates to Keplerian Elements

To convert from Cartesian coordinates to Keplerian elements, we first calculate the angular momentum vector $\vec{h}$ and the node $\vec{n}$:

$$\begin{aligned} \vec{h} &= \vec{r} \times \vec{v} \\ \vec{n} &= \hat{k} \times \vec{h}. \end{aligned} \tag{2.34}$$

Since the position and velocity are in the orbital plane of the satellite, $\vec{h}$ points in the normal direction of the orbital plane. The node vector $\vec{n}$ points in the direction of the intersection of the orbital and the equatorial plane, called the ascending node.

Eccentricity (e)

The eccentricity determines the *shape* of the orbit. The eccentricity is a number between 0 and 1: if $e = 0$, then the orbit is circular and if $e \approx 1$, then the orbit is highly elliptical. The eccentricity is calculated from the eccentricity vector point in the direction of perigee (closest approach) and is defined as

$$\vec{e} = \frac{1}{\mu}\left[\left(v^2 - \frac{\mu}{r}\right)\vec{r} - (\vec{r}\cdot\vec{v})\vec{v}\right]. \tag{2.35}$$

The scalar value of the eccentricity is calculated as the norm of the eccentricity vector:

$$e = \|\vec{e}\|. \tag{2.36}$$

Semi-major axis (a)

The semi-major axis of an orbit describes the *size* of the orbit. In an ellipse with the Earth at one focus, the semi-major axis defines the furthest point at which the orbiting object (satellite) orbits the Earth. A large value of a indicates that the object orbits at a large distance from the Earth, whereas a small value of a indicates that the object orbits within a smaller radius. The semi-major axis is calculated as

$$\vec{a} = \frac{\vec{h}\cdot\vec{h}}{\mu(1-e^2)}. \tag{2.37}$$

Inclination Angle (i)

The inclination angle determines the *orientation* of the entire orbit. The inclination angle is defined either as the angle between the equatorial plane and the orbit plane, or as the angle between the North pole (z-axis) and the normal to the orbital plane (direction of $\vec{h}$). The equation to find i is

$$i = \cos^{-1}\left(\frac{\hat{k}\cdot\vec{h}}{h}\right) \tag{2.38}$$

where the inverse cosine is defined so that $0 \le i < \pi$.

Right Ascension of Ascending Node (Ω)

The right ascension Ω describes the location of the ascending node with respect to the x-axis in the ECI frame. Using the node defined in equation (2.34), we get

$$\Omega = \cos^{-1}\left(\frac{\hat{\imath}\cdot\vec{n}}{n}\right). \tag{2.39}$$

Note that because $\cos^{-1}(x)$ gives angles between 0 and π, if the direction of the ascending node has a negative y component then Ω is in the negative y-axis and has value between π and 2π. Hence, we define Ω as follows:

$$\text{If} \quad \vec{n}\cdot\hat{\jmath} < 0 \quad \text{then} \quad \Omega = 2\pi - \left[\cos^{-1}\left(\frac{\hat{\imath}\cdot\vec{n}}{n}\right)\right]. \tag{2.40}$$

Argument of Perigee (ω)

The argument of perigee is the angle between the ascending node and the direction of perigee given by $\vec{e}$:

$$\omega = \cos^{-1}\left(\frac{\vec{n}\cdot\vec{e}}{e}\right). \tag{2.41}$$

For the same reason as above, we set ω as follows:

$$\text{If} \quad \vec{e}\cdot\hat{k} < 0 \quad \text{then} \quad \omega = 2\pi - \left[\cos^{-1}\left(\frac{\vec{n}\cdot\vec{e}}{e}\right)\right]. \tag{2.42}$$

Mean Anomaly (M)

The mean anomaly is also known as Kepler's equation and calculated as

$$M = E - e\sin(E) \tag{2.43}$$

where E is the eccentric anomaly described below.

Eccentric Anomaly (E) The eccentric anomaly is the angle between the semi-major axis and the satellite in the orbit plane.

$$E = \cos^{-1}\left(\frac{e+\cos(\theta)}{1+e\cos(\theta)}\right) \tag{2.44}$$

where θ is the true anomaly described below.

True Anomaly (θ) The true anomaly is simply the angle between the perigee, the eccentricity vector, and the position vector of the satellite. Using the dot product, it is given as

$$\theta = \cos^{-1}\left(\frac{\vec{e}\cdot\vec{r}}{re}\right). \tag{2.45}$$

For the same reason as described for equations (2.40) and (2.42), it is modified if necessary:

$$\text{If} \quad \vec{r}\cdot\vec{v} < 0 \quad \text{then} \quad \theta = 2\pi - \left[\cos^{-1}\left(\frac{\vec{e}\cdot\vec{r}}{re}\right)\right]. \tag{2.46}$$

2.3.2 Conversion From Keplerian Elements to Cartesian Coordinates

The next step is to go from a Keplerian element set to ECI (Cartesian) coordinates given only the six elements a, e, i, Ω, ω, and M. The first parameter to be calculated is the eccentric anomaly by solving equation (2.43) for E. Since it is easier to do this numerically, the algorithm shown as Algorithm 1 is usually used instead. The final eccentric anomaly from the loop is the value we use for E.

```
Algorithm 1:  Eccentric Anomaly
    i = 1
    $E_i$ = M
    tolerance = 1e-15   % Note:  value can be changed by user.
    difference = 1
    while difference $\geq$ tolerance do
      $E_{i+1}$ =   $E_i - \frac{E_i - e\sin E_i - M}{1 - e\cos E_i}$
      difference = $|E_i - E_{i-1}|$
      if difference $\leq$ tolerance  then
        Exit the loop
      end if
      i = i + 1
    end while
```

Next, we use ω and Ω to find the unit vectors of the perifocal coordinate system. The perifocal coordinates are based on the motion of the satellite in that two of its three components are inside its orbital plane. The vector $\vec{P}$ points along the periapsis (closest approach) and the vector $\vec{Q}$ is rotated $90°$ in the direction of the orbital motion. The third component that is determined by $\vec{P} \times \vec{Q}$ points orthogonal to the orbital plane into the same direction as $\vec{h}$.

Hence, after some algebra and by writing $\vec{P}$ and $\vec{Q}$ in terms of the Keplerian elements we find that

$$\begin{aligned}
P_x &= \cos(\omega)\cos(\Omega) - \sin(\omega)\cos(i)\sin(\Omega) \\
P_y &= \cos(\omega)\sin(\Omega) + \sin(\omega)\cos(i)\cos(\Omega) \\
P_z &= \sin(\omega)\sin(i) \\
Q_x &= -\sin(\omega)\cos(\Omega) - \cos(\omega)\cos(i)\sin(\Omega) \\
Q_y &= -\sin(\omega)\sin(\Omega) + \cos(\omega)\cos(i)\cos(\Omega) \\
Q_z &= \sin(i)\cos(\omega).
\end{aligned} \tag{2.47}$$

These six components make up the vectors $\vec{P}$ and $\vec{Q}$ as shown below:

$$\begin{aligned}
\vec{P} &= P_x\hat{\imath} + P_y\hat{\jmath} + P_z\hat{k} \\
\vec{Q} &= Q_x\hat{\imath} + Q_y\hat{\jmath} + Q_z\hat{k}.
\end{aligned} \tag{2.48}$$

Finally, from equation (2.48) we see that $\vec{r}$ and $\vec{v}$ are defined as

$$\begin{aligned}
\vec{r} &= a(\cos(E) - e)\vec{P} + a\sqrt{1-e^2}\,(\sin(E))\,\vec{Q} \\
\vec{v} &= -a(\sin(E))E'\vec{P} + a\sqrt{1-e^2}\,(\cos(E))\,E'\vec{Q},
\end{aligned} \tag{2.49}$$

where

$$E' = \frac{1}{(1 - e\cos(E))}\sqrt{\frac{\mu}{a^3}}. \tag{2.50}$$

2.3.3 Closed-Form Analytic Representation

The next task after converting Cartesian coordinates from the ECI frame to Keplerian elements is to determine or approximate their analytical expression. The main source we used to solve this problem was the analysis offered by Liu and Segrest [19] who discuss the frequency domain using Fast Fourier Transformations (FFT). More specifically, they outline a method to represent the orbital elements with general analytical expressions of the form

$$X_j = \sum_{i=0}^{\ell} X_{ji}^{s} t^i + \sum_{k=1}^{m} X_{jk}^{p} \cos(\dot{\theta}_{jk} t + \phi_{jk}) \tag{2.51}$$

where the index $j = 1, 2, \ldots, 6$ stands for the six Keplerian elements. The X_{j0} terms represent the initial values of the elements without the periodic terms, and the superscripts s and p represent the polynomial and periodic terms, respectively. The first term on the right hand side of equation (2.51) is the polynomial term where the X^s_{ji} terms are the coefficients of the non-periodic (polynomial terms) terms. The second term on the right hand side of equation (2.51) is the periodic term where the terms X^p_{jk}, $\dot{\theta}_{jk}$, and ϕ_{jk} are the amplitude, frequency, and phase angle of the periodic term, respectively.

The authors continue to describe the following technique to find the analytical expressions of the Keplerian elements.

1. First, compute the terms X^p_{jk}, $\dot{\theta}_{jk}$, and ϕ_{jk} using a Fast Fourier Transform (FFT) for a specific periodic motion of a Keplerian element. Then remove the entire periodic component form X_j by adding the out-of-phase term
$$X^p_{jk} \cos(\dot{\theta}_{jk} t + \phi_{jk} + \pi) \tag{2.52}$$
to the periodic term $X^p_{jk} \cos(\dot{\theta}_{jk} t + \phi_{jk})$. The result of this step is that the periodic term in equation (2.51) is 180° out of phase with equation (2.52) and therefore cancel outs.
2. Next, after all the periodic terms are canceled out, only the polynomial terms will remain. Hence, it is now possible to find the analytical expression and to determine the coefficients X^s_{ji} of the polynomial terms using interpolation techniques.
3. Finally, combine the polynomial terms determined in the previous step with the periodic terms from the first step, by adding the appropriate terms to produce a time dependent expression for the particular Keplerian element X_j.

Below we describe the work involved in three three steps in a little bit more detail. Whereas the authors' original discussion contains an analysis using their own data of Keplerian elements, in our analysis we use data provided by our sponsor SpaceNav formatted as a column of equally spaced time stamps with corresponding position and velocity measurements in the (x, y, z) ECI frame. Hence, we first need to convert the ECI coordinates into Keplerian elements as described earlier in Section 2.3.1.

As mentioned for the first step above, the primary technique to find the amplitude, frequency, and phase is the FFT method which involves the transformation of a time signal into a frequency signal to better or more efficiently describe some of the parameters involved. The FFT is the transformation of N time dependent signals into a frequency dependent signal. The result is a complex number of length N. We can weigh out the results of the FFT signals by N and then find the (power) spectrum of the signal by taking only half ($N/2$) of the complex weighted signals and squaring them.

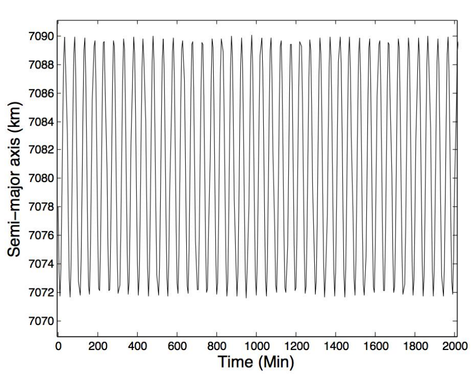
The plot of the Keplerian elements before we find the spectrum is shown as a function of time in Figure 2.6. There are several things to note:

1. The entire range of the data is not shown, but we zoomed in to about the first fourth of the entire data span.
2. The periodicity of the Keplerian elements depends somewhat on the data, and for different types of orbits the shape of some of the parameters could be slightly different.

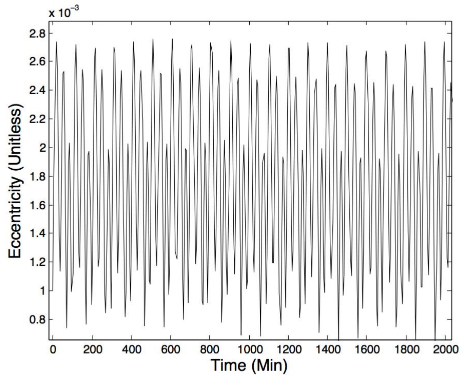
To restate the purpose of our analysis, recall that we are interested to simplify these seemingly busy, cumbersome, and quite complex plots of the time dependent signal in order to allow their easy interpretation once we find the power spectrum. Using the FFT function in MATLAB, we can directly determined the spectrum of the Keplerian elements. The results we got are shown in Figure 2.7 with logarithmic scales on both axes. Note here that since we are moving to the frequency domain, it is natural to adjust the axes to log scale to see the entire spectrum, and to re-adjust (go back to linear scale) only later as needed. It is also important to notice that the first decade of the data is irrelevant in determining frequency because it is a contribution from low or zero frequency measurements.

The graphs shown in Figure 2.7 should be interpreted carefully because they do not necessarily represent the actual amplitude (power) spectrum. This is mainly because a low frequency component of the signals (DC in analysis) is present in the first decade, from 10^{-6} to 10^{-5} on the x-axis. This low frequency part does not contribute to the periodicity of the signal but rather offsets the entire spectrum by a particular amount. Moreover, the DC component usually has a large magnitude which subsequently has an effect of "swallowing-up" or "leaking into" near-by measurements. The resulting effect of this is to raise (or lower) close-by measurements.

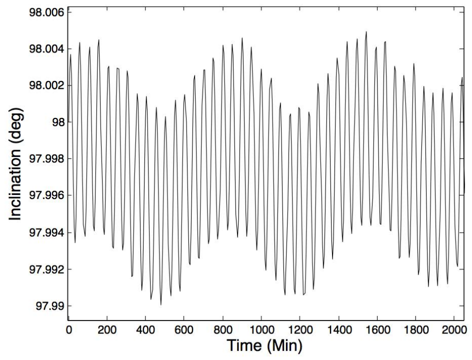
There is a common method in signal analysis called off-setting or zeroing out the DC term, which involves subtracting the average of the entire signal from each component of that same signal. For example, suppose that we have a time dependent signal $f(t)$ with N components given by $(f_1, f_2, \ldots, f_N)$



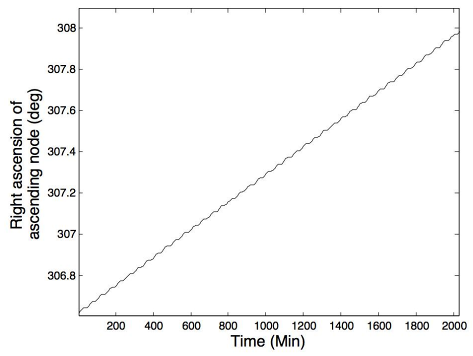
(a) Semi-MajorAxis vs. time.



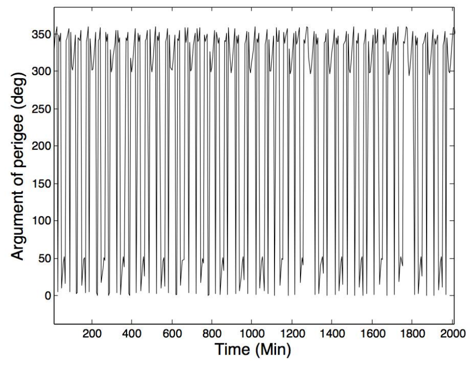
(b) Eccentricity vs. time.



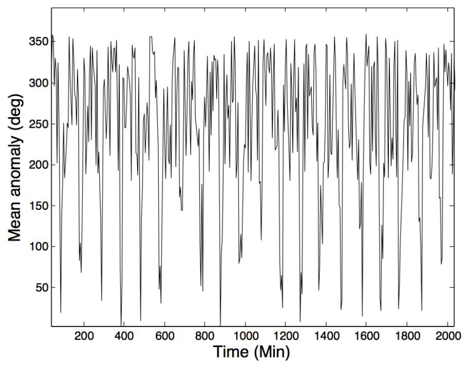
(c) Inclination vs. time.



(d) Right ascension of ascending node vs. time.

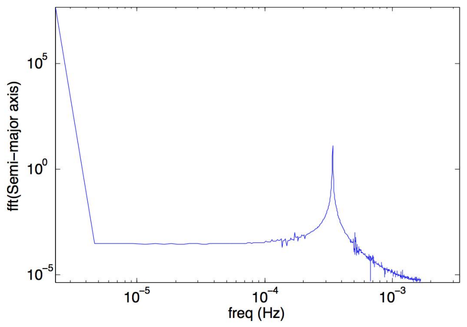


(e) Argument of perigee vs. time.

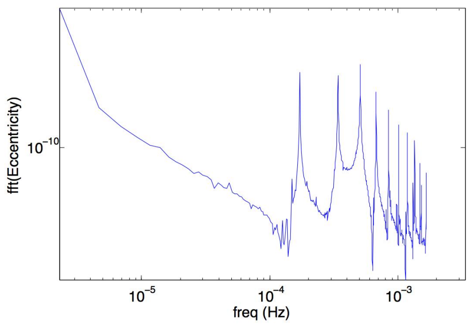


(f) Mean anomaly vs. time.

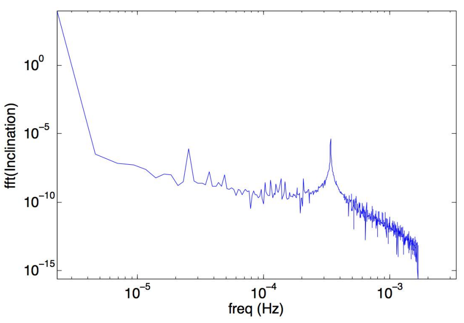
Figure 2.6: Output of the `Kepler` function showing the Keplerian elements as a function of time in minutes, before pre-processing



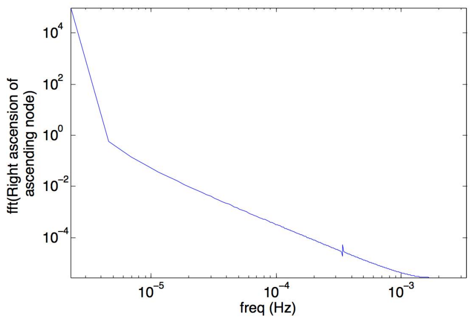
(a) Spectrum of the semi-MajorAxis on a log-log plot.



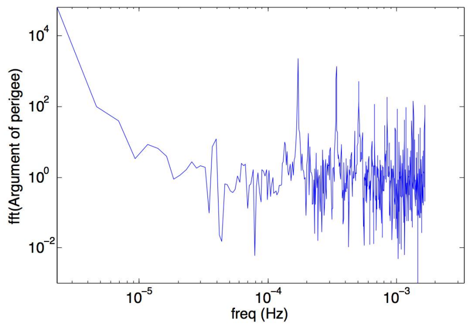
(b) Spectrum of the eccentricity vs. frequency.



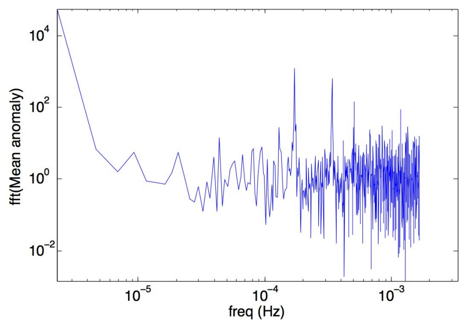
(c) Spectrum of the inclination angle.



(d) Spectrum of right ascension of ascending node.



(e) Spectrum of the argument of perigee.



(f) Spectrum of the mean anomaly.

Figure 2.7: Output of the `Periodic_Test` function showing the power spectrum of the Keplerian elements in the frequency domain

whose average is denoted by $\bar{f} = (\sum_{i=1}^{N} f_i)/N$. The new signal looks like $(f_1 - \bar{f}, f_2 - \bar{f}, \ldots, f_N - \bar{f})$. This is done to bring the magnitude of the signal close to the zero on the y-axis which enables the signal to be represented as much as a purely sinusoidal signal as possible. However this method only works for functions that are oscillating between two constant values like the spectrum in Figures 2.6(a), 2.6(b), 2.6(e), and 2.6(f).

The other issue to consider when interpreting log plots is that equally spaced peaks in the spectrum may appear to be bunched up on the right side of the decade. This is shown in all the graphs in Figure 2.7 but 2.7(d). So we improve this effect by plotting the power spectrum with a linear x-axis and a logarithmic y-axis. Figure 2.8 shows the combined result of the zeroing out the low frequency and adjusting the log x-axis to a linear scale. Note again that the entire range of the data is not shown and that the periodicity of the Keplerian elements depends somewhat on the data.

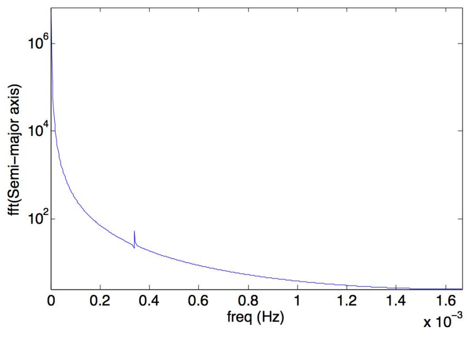
There are two important frequencies in a time signal to check in every equally spaced signal, the sampling frequency, and the Nyquist frequency:

- The sampling frequency $f_s = 1/(\Delta t) = \frac{1}{(t_i - t_{i+1})}$ is the number of samples per second taken, and a particular value of the sample represents the magnitude of the input signal at the sampling frequency.
- The Nyquist frequency $f_N = \frac{1}{2} f_s$ is defined such that for the signal analysis to have any meaning the sampling frequency should be at least twice the Nyquist frequency. Another way to state this is that the sampling frequency should be at least two times the highest frequency contained naturally in the signal.

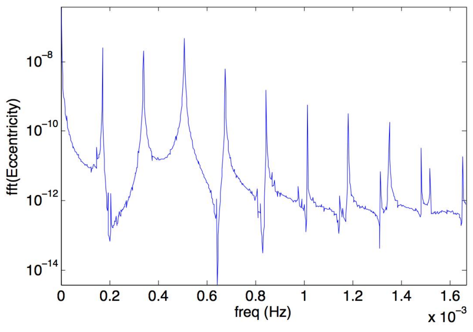
It is important to mention that there are other sources that define these two frequencies differently; for example, sometimes the sample frequency is used instead of the Nyquist frequency. In our case, up to this point the frequencies in Figures 2.7 and 2.8 were found by weighting the $N/2$ bins that were constrained by the Nyquist frequency. Hence, the frequencies have the form

$$f = \frac{k f_N}{N/2} \text{ for } k = 1, 2, \ldots, N/2. \tag{2.53}$$

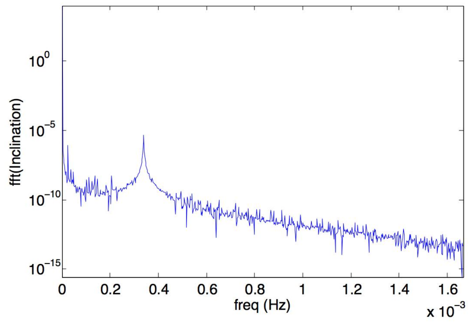
The first parameters to be calculated from the FFT are the frequencies $\dot{\theta}_{jk}$ from equation (2.51). So naturally the next step is to describe the important features of the FFT so that the numerical methods give reasonable results.



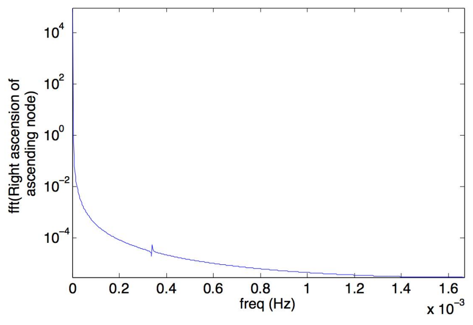
(a) Spectrum of the semi-MajorAxis on log-linear scale after DC offset.



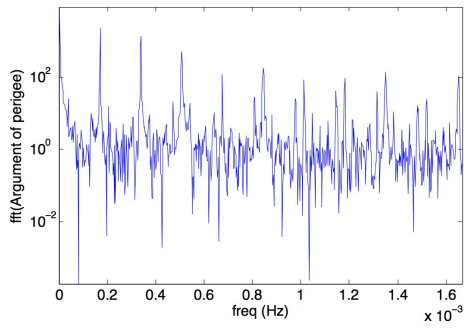
(b) Spectrum of the eccentricity vs. frequency on log-linear scale after DC offset.



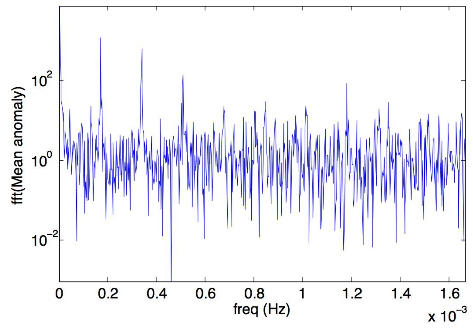
(c) Spectrum of the inclination angle on log-linear scale without DC offset.



(d) Spectrum of right ascension of ascending node on log-linear scale without DC offset.



(e) Spectrum of the argument of perigee on log-linear scale after DC offset.



(f) Spectrum of the mean anomaly on log-linear after DC offset.

Figure 2.8: Output of the `Kepler` function showing the Keplerian elements as a function of time in minutes, after pre-processing

In order to use FFT for performing a Discrete Fourier Transform (DFT), the input to the FFT needs to be equally spaced data [19].

Moreover, if the sampling frequency has a low rate, then the frequency components with values greater than $f_s/2$ will be merged with, or 'fold back', into the analyzed spectrum and alter the result in a phenomenon known as 'aliasing' in signal analysis. For now, we are not sure if the data we get has a low sampling frequency. The common solution to 'aliasing' is to take a different set of data with a different sampling rate and find the spectrum. Unfortunately, we are only provided with data with a fixed sampling rate.

Several other aspects were mentioned in the original paper [19], including the importance of a correct interpretation of frequency and phase angles for which the authors propose the Blackman amplitude weighting methods together with frequency centering techniques from signal analysis. In view of the above difficulties, these techniques were not studied any further.

2.3.4 Discussion of Implementation

Based on the graphs in Figure 2.6 we can observe that the Keplerian elements have a decaying periodic form that can be modeled using periodic and polynomial functions. Hence, under the additional assumption that the decaying term goes down linearly, we expect to be able to find analytic expressions for the elements by decomposing the time signal into linear and trigonometric terms.

Analytic_form.m

Given periodic data, we construct a MATLAB function `Analytic_form` to compute the coefficients of an analytic expression of the form

$$F(x) = T(x) + P(x), \tag{2.54}$$

where $T(x)$ represents the periodicity by trigonometric functions and $P(x)$ contains the linear offset of the data as a polynomial. We assume that $F(x)$ is given as discrete data, for example, as time history of a Keplerian element set based on which the analytic form needs to be formulated.

- **Input:** There are four inputs into the function `Analytic_form` each of which is described below:

data: the vector from periodic data of Keplerian elements needed to be transformed to analytic form;

period: the period of the data;

start_pnt: the number of the vector element, that is starting point of the first cycle;

crcl_num: the number of total cycle.

- **Output:** The outputs labeled T and P are arrays that contain all the coefficients needed to construct the trigonometric and polynomial form.

First, we interpolate on one cycle where we we take a first cycle from `start_pnt` of length `period`, and find the function $T(x)$. Next, we interpolate for the polynomial expression of the data by subtracting the periodic part from the general Keplerian element data $F(x)$, simply yielding $P(x) = F(x) - T(x)$. To find the linear function that starts at the beginning of the first cycle, we label the first point where the periodicity starts as A and let the last point of the last given cycle be the second point denoted by B. Each cycle is part of the same data history with a total number of elements equal to its period, so that ideally all elements are equal to the elements of any other cycle. After we found points A and B, we build $P(x)$ as that function that goes through this two points by the slope formula

$$x - \frac{A(x)}{B(x)} - A(x) = y - \frac{A(y)}{B(y)} - A(y). \tag{2.55}$$

Here, the coefficients for the polynomial function are found by evaluating the trigonometric function at the points A and B.

Run_analytic_form.m

The function `run_analytic_form` creates a sample of bigger size from a given sample of Cartesian coordinates. This routine can be used in cases where our own two data files (input data of time history of two objects) are of different size. In this case, to make them of the same size, we need to increase the size of smaller file by increasing its data history.

- **Input:** The inputs to the function `run_analytic_form` are as follows:

data: the first column contains time in seconds, columns two to four contain position of in the ECI fame, and columns five to seven contain the corresponding velocity;

period: an array of six periods for the six Keplerian elements a, e, i, Ω, ω, and M;

sample_num: the size of the sample for the output data;

dt: the time step for output data in seconds.

- **Output:** A sample of Cartesian coordinates for time history of an object for a needed size.

The way this function works can be separated in several steps:

1. The time history of the satellite is converted into Keplerian elements.
2. The analytic form of each Keplerian element is determined as a sum of trigonometric and polynomial form.
3. Using these analytic form expression we create a sample of bigger size for each set of the Keplerian elements.
4. We convert the set of Keplerian elements back to the Cartesian history, which now has a bigger size.

Form_calc.m

Finally, the function `form_calc.m` takes all the coefficients for trigonometric and polynomial components that we get from `analytic_form` as input and calculates the value of a particular Keplerian element at any given time t. This function is a subfunction of `run_analytic_form` and can be used in the case that we have two input files of different sizes.

2.4 Error Propagation and Computation of Collision Probabilities

In addition to the computational challenge of finding good representations of satellite trajectories with inevitable numerical errors and inaccuracies, another key source of error is the difficulty to accurately track objects in space and determine their current, actual positions and velocities. Clearly, to identify an optimal collision avoidance maneuver, it therefore is important to understand how error in trajectories can be represented and dealt with. Let us begin to define an error in trajectory as the uncertainty in the position of an orbiting object. Due to many constraints and technological limitations, it is generally impossible for us to know the exact location of a satellite at an arbitrary time t. We may have an idea of where a satellite is, but we are not certain of the exact position. If the error in a satellite's trajectory is neither known nor sufficiently understood, there is little hope that we can find a reasonable solution to our problem. Mathematically, error propagation is a complex topic that has numerous pieces to it, as described by Bush [8] who addresses the main constraints seen when trying to estimate error in orbital trajectories. He also provides a covariance method to propagate into the future. Tsuda [28] addresses the state transition matrix for orbiting objects around a single mass, explains how the state transition matrix is created, and and discusses its role in error propagation.

2.4.1 Representation of Uncertainty Errors

Three common mathematical quantities used to represent errors from uncertainty are covariance matrices, state error transition matrices, and error ellipsoids that are briefly described below.

Covariance Matrices

To begin, error analysis of objects in orbit have what is called a covariance matrix. This matrix has entries of the variance in distance where a satellite can be positioned in reference to its actual position. In a multi-dimensional system, the covariance is a symmetric matrix as exemplified below:

$$\begin{bmatrix} 4.44 & 2.73 & 2.23 & -0.27 & -1.91 & -1.05 & -1.24 \\ 2.73 & 3.92 & 2.64 & -0.17 & -1.59 & -1.29 & -1.61 \\ 2.23 & 2.64 & 2.35 & 0.00 & -1.28 & -1.01 & -1.36 \\ -0.27 & -0.17 & 0.00 & 0.23 & 0.13 & 0.01 & -0.11 \\ -1.91 & -1.59 & -1.28 & 0.13 & 1.63 & 1.12 & 1.28 \\ -1.05 & -1.29 & -1.01 & -0.01 & 1.12 & 1.20 & 1.26 \\ -1.24 & -1.61 & -1.36 & -0.11 & 1.28 & 1.26 & 1.95 \end{bmatrix}. \tag{2.56}$$

The diagonal contains the covariances in distance, and the off-diagonal terms are the cross covariances. The off-diagonal terms can be transformed into error correlations. The covariance is estimated and then propagated using the linear state transition matrix.

State Transition Matrices

The state transition matrix is a Jacobian matrix (i.e. a set of partial derivatives) that relates changes in a state vector from one point in time to another:

$$J = \begin{bmatrix} \dfrac{\partial y_1}{\partial x_1} & \cdots & \dfrac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial y_m}{\partial x_1} & \cdots & \dfrac{\partial y_m}{\partial x_n} \end{bmatrix}. \tag{2.57}$$

It is formulated using the position vector in Cartesian or Keplerian coordinates. A matrix derivation is explained in detail by Tsuda [28]. The state transition matrix is used to find the probability of collision of two objects in orbit based on their associated error correlations.

Error Ellipses

The different positions of an object in orbit can also be represented in a cluster around the actual position of the object that is usually in the shape of an ellipse. This is what is called an error ellipse, or a state uncertainty as explained by Campbell [9]. If these ellipsoids overlap we have a probability of collision. The magnitude of this probability is related to the percentage of overlap of these ellipses. Figures 2.9 and 2.10 illustrate these relationships between covariance-based error ellipses and their overlaps, and the resulting collision risks for two objects with low and high covariance interactions, respectively.

2.4.2 Calculation of Collision Probabilities

Let us now consider two objects and suppose that there are uncertainties in the positions of both objects. These uncertainties are handled by combing them into a single larger uncertainty in the position of one object. We then assume complete certainty in the position of the other object, but treat its size as the combined sizes of the two objects. This is known as the combined hard-body radius. As an encounter unfolds, the latter object passes through a

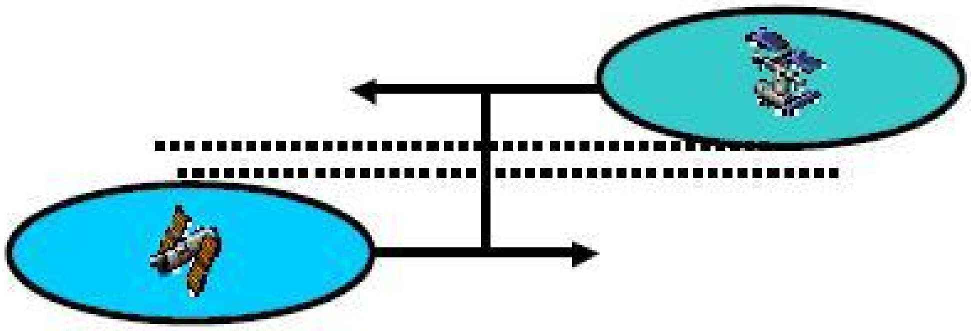


Figure 2.9: Two objects with low covariance interaction and collision risk

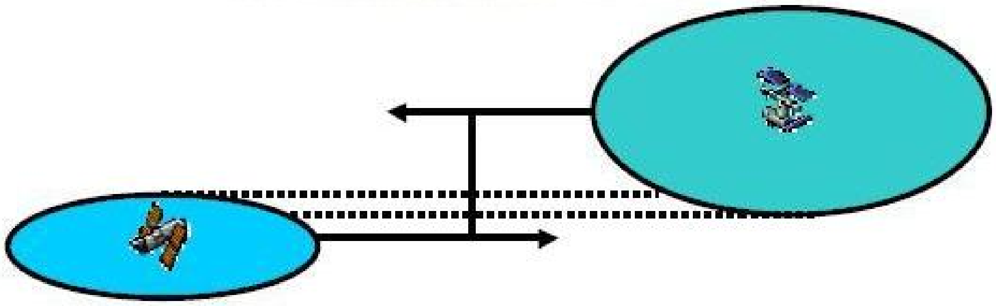


Figure 2.10: Two objects with larger covariance interactions and significant ellipsoidal overlap resulting in a larger risk of collision

certain volume of the error ellipsoid representing the probability distribution of the former object. Ultimately, we want to measure and control this volume.

Let us further assume that the primary object of concern is also the object that we can control, and that we simply refer to as the *primary*. Analogously, the other object whose trajectory passes through the collision volume with a positive probability is referred to as the *secondary*. To simplify this preliminary discussion and the mathematics involved, we typically treat the primary as a point mass and the secondary as if it had the physical radii of both objects combined. This combined radius is called *hard-body radius*. The time of closest approach corresponds to that point in time at which the probability of collision is greatest, and we will call this event our (critical) *encounter*. The *encounter plane* is the plane perpendicular to the relative

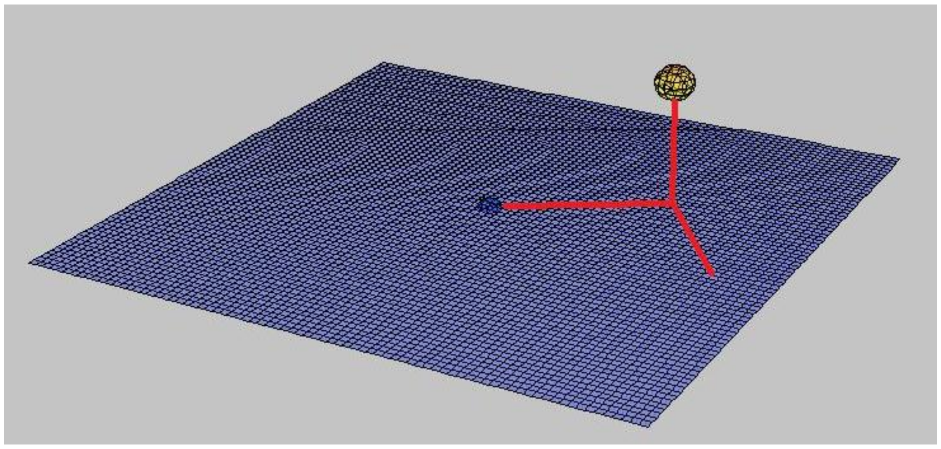


Figure 2.11: Illustration of the encounter plane and its basis vectors

velocity vector near the time of closest approach, with our primary object at the center. The *miss vector* is defined as the vector that stretches *from* our primary *to* our secondary at the actual time of closest approach, when the secondary is intersecting the encounter plane. The cross-product of these two vectors gives a third basis vector for this encounter frame, shown in Figure 2.11.

As mentioned earlier, we suppose that the position of each object is subject to uncertainty, where the covariances are assumed to have an approximately normal distribution. In an effort to maintain a common standard by which to measure this uncertainty, Air Force personnel at the Joint Space Operations Center (JSpOC) at the Vandenberg Air Force Base in California, and the North American Aerospace Defense Command (NORAD) at the Peterson Air Force Base in Colorado Springs generates these covariances and includes them with the ephemeris data that is provided to satellite operators. We can therefore use the combined uncertainty of the primary and secondary to generate a probability surface, centered at our primary. The general equation to find the probability of collision between two passing objects is then

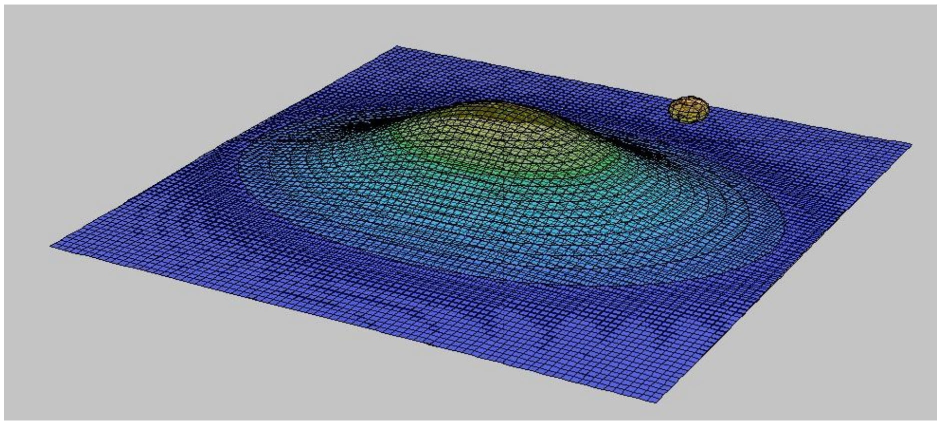


Figure 2.12: Illustration of a probability surface with a secondary object

given as follows [24]:

$$P = \frac{r_a^2}{2\pi\sigma_x\sigma_z\sqrt{1-\rho_{xz}^2}} \iint\limits_{A_a} \exp\left[-\frac{1}{2(1-\rho_{xz}^2)} \cdot \left(\left(\frac{x}{\sigma_x}\right)^2 - 2\rho_{xz}\left(\frac{xz}{\sigma_x\sigma_z}\right) + \left(\frac{z}{\sigma_z}\right)^2\right)\right] \tag{2.58}$$

where

$$\begin{aligned} P &= \text{probability of collision;} \\ r_a &= \text{combined hard-body radius;} \\ A_a &= \text{cross-sectional area of the combined hard-body radius}(= \pi r_a^2); \\ x, z &= \text{components of the hard-body radius in the encounter plane;} \\ \sigma_x, \sigma_z, \rho_{xz} &= \text{parameters in the combined covariance in the encounter plane.} \end{aligned}$$

Here, the z-direction is from the primary object to the secondary object at the time of closest approach. The x-direction is perpendicular to both the z-direction and the relative velocity of the objects. Furthermore, if the secondary (now defined with the hard-body radius) is much smaller than the primary covariance ellipse, it turns out that the integrand does not need to be evaluated but can be simply replaced by the value at the center of the hard body [24].

Now take a look at Figure 2.12 which shows a probability surface overlaid on the encounter plane, with a secondary object near the time of closest approach. Here it is worth making special note of the level equiprobability curves, where the curve through which the secondary is passing corresponds to the calculated probability of collision. Hence, we can now see that there is a direct mapping between the probability space and the physical space. Later, when we move our primary as part of a delta-v maneuver, this will also effect its corresponding probability surface thus causing the secondary to pass through some other, and preferably less critical equiprobability curve.

In practice, it is typical the role of the satellite operator to determine that probability of collision that is deemed low enough to be safe. This is referred to as the threshold probability P_T. The equation for the threshold probability contour is [24]:

$$\left(\frac{x_T}{\sigma_x}\right)^2 - 2\rho_{xz}\left(\frac{x_T z_T}{\sigma_x \sigma_z}\right) + \left(\frac{z_T}{\sigma_z}\right)^2 = \left(\frac{x_C}{\sigma_x}\right)^2 - 2(1-\rho_{xz}^2)\ln\left(\frac{P_T}{P_C}\right). \quad (2.59)$$

Note that all the terms on the right hand side are known, and that this is the equation of an ellipse on which the probability of collision P_C equals the threshold probability P_T.

2.4.3 Discussion of Implementation

Following the previous discussion, we have developed a program to determine if a possibility of collision is substantial, or can safely be ignored. The information we are given consists of positions, times of closest approach and a miss distance of two objects. The inputs for the program are:

1. the position vector of the first object;
2. the position vector of the second object;
3. the velocity vector of the first object;
4. the velocity vector of the second object;
5. the covariance of uncertainty of first object;
6. the covariance of uncertainty of second object;
7. the hard body radius of the two objects, or miss distance.

From this information, we determine a probability of collision between the primary and secondary satellite around the particular time specified. If the probability is high enough for there to be concern, we move on to the next stage of the collision avoidance process (review the decision chart in Figure 1.2).

The first thing that is handled by the program is a change of coordinate systems; compare the beginning of Section 2.3. Three right-handed coordinate systems are used.

- The inputs use the Earth-centered inertial (ECI) frame with the origin at the center of the Earth.
- After converting each satellite's ECI coordinates to a radial(R), in-track (I), and cross-track (C) component, all calculations involving a single satellite the corresponding (RIC) frame.
- Finally, calculations involving the encounter of two satellites are based on the satellites' relative positions and velocities, and therefore use the frame associated with the corresponding encounter plane as depicted in Figure 2.11.

We convert between these frames by using orthogonal matrices. The relative position of the secondary object with respect to the primary object is converted from the ECI frame to the encounter frame. The covariance matrix of each satellite is converted from the ECI basis to the satellite's RIC frame. It adds the resulting covariance matrices and the combined matrix is then transformed into the encounter frame.

Now we can actually calculate the probability of collision. We have the relative position of the secondary object, along with its matrix of covariance representing the uncertainty in its position in the encounter frame. This allow us to define a two-dimensional normal distribution centered on the secondary object. The primary satellite is assumed to be spherical with its radius equal to the hard-body radius input parameter. The primary satellite covers a circular area in the encounter plane. This area has a probability associated with it, which can be calculated by integrating the two-dimensional Gaussian distribution over the circular area. This explains the output of our program.

Chapter 3

Optimization

Combining some of our new knowledge about the geometry of satellite trajectories, the possibility to influence them using delta-v maneuvers, and the existence of an underlying probability or encounter plane, we studied two relevant strategies for trajectory optimization described by Peterson [24]. These two strategies, that we refer to as "Miss Vector Approach" and the "Probability Gradient Approach" for reasons outlined below, determine and calculate orbital maneuvers that lower the risk of collision when two objects are about to have a close encounter. In comparison to other methods that often rely on numerical solutions that can be computationally expensive, the methods we chose for implementation provide approximate analytical solutions of the optimal delta-v maneuver, and hence are relatively easy to code and quick to compute. In each case, our maneuver will consist of time, magnitude and direction of an associated burn of fuel.

3.1 Terminology and Model Assumptions

Let us begin by speaking about the objects and their encounter. Analogous to Section 2.4.2, we refer to the two objects of interest as primary and secondary. As before, it is convenient to treat the primary object as a single point mass and to add its actual radius to that of the secondary yielding its new, hard-body radius. The encounter plane is defined as the plane perpendicular to the relative velocity vector of the two objects near the time of closest approach, with our primary object at the center. The miss vector is defined as the vector that stretches from our primary to the secondary at the actual time of closest approach, when the secondary is intersecting the encounter plane. The cross-product of these two vectors gives a third basis vector for this encounter frame, that is described in more detail in Section 2.3 and depicted in Figures 2.11 and 3.1.

We now assume that the operator determines a probability of collision that is deemed low enough to be safe. This is referred to as the threshold probability and set by the owners or operators of the satellite depending on its age or particular mission. Using the probability of collision and the time of closest approach codes, we can calculate the chance of there being a collision. If the resulting probability exceeds the allotted threshold of risk that we are willing to take, then we need perform a delta-v maneuver to redirect the primary in order to lower a probability of collision. The time of this maneuver is also decided by the operator. Hence, for a given time and threshold probability, we need to find the direction and magnitude of the optimal burn.

As indicated above, the goal of our maneuver is to change and effectively increase the distance of the objects at the time of their closest approach:

- In the miss vector approach illustrated in Figures 3.2 and 3.3, we move the objects directly away from each other. "Directly" means that we move the primary satellite along the actual miss vector in direction opposite to the intersection of secondary object and the physical encounter plane. This assumes that the objects actually occupy the positions that the tracking system indicates.
- In the probability gradient approach illustrated in Figures 3.2 and 3.4, we more directly target to lower the probability of collision by moving the primary in the direction of the negative gradient of the underlying probability distribution. This reduces the probability of collision as quickly as possible, resulting in the shortest relative path for the secondary out and away from the predicted collision ellipsoid of the primary. This approach assumes that the relative position of the objects is a normally distributed random variable.

The following section analyzes the type of maneuvers that need to be performed at an a priori specified time to effect one of these changes at the time of closest approach. We emphasize that the optimization problems underlying these methods have analytic solutions, so that they can be calculated in real or near-real time.

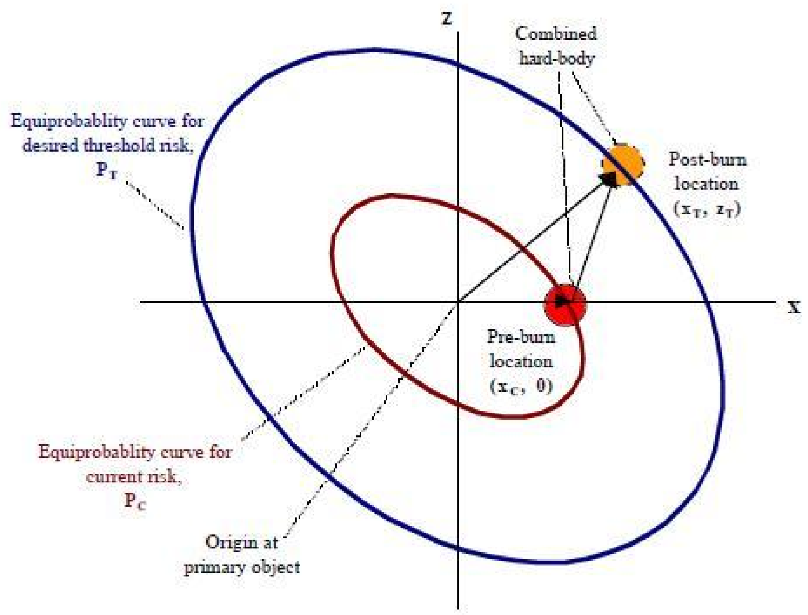

z
Combined hard-body
Equiprobablity curve for desired threshold risk, P_T
Post-burn location (x_T, z_T)
x
Pre-burn location $(x_C, 0)$
Equiprobablity curve for current risk, P_C
Origin at primary object

Figure 3.1: Encounter plane with the primary object at its origin and two equiprobability curves for current risk and target risk

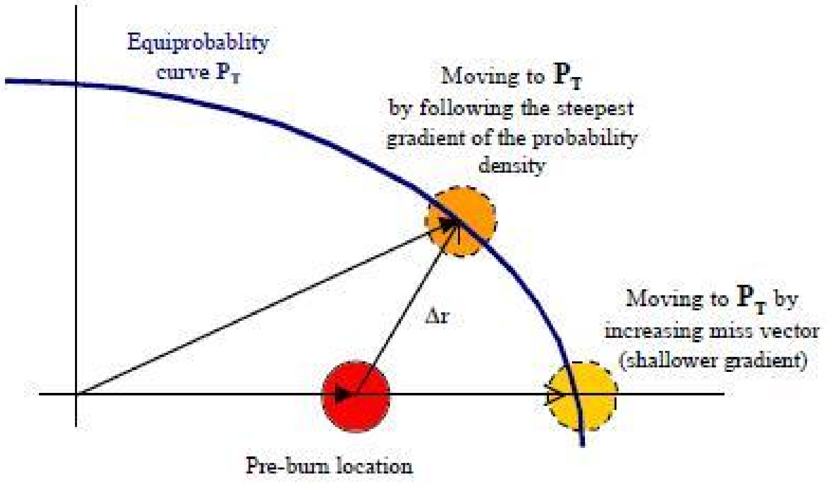

Equiprobablity curve P_T
Moving to P_T by following the steepest gradient of the probability density
Δr
Moving to P_T by increasing miss vector (shallower gradient)
Pre-burn location

Figure 3.2: Two maneuver strategies for lowering the collision probability and equiprobability curves from current to target risk

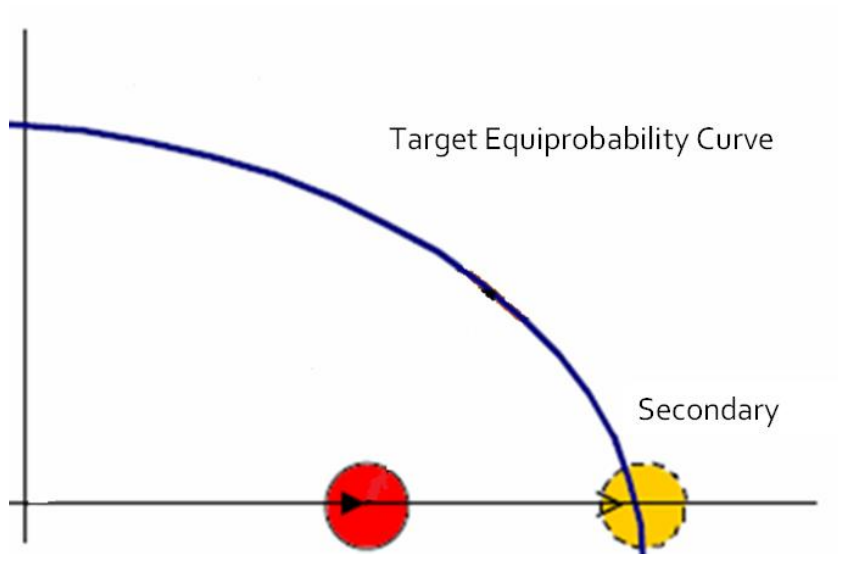


Figure 3.3: Illustration of the miss vector approach: the primary moves directly along the miss vector in direction opposite to the intersection of secondary and the physical encounter plane

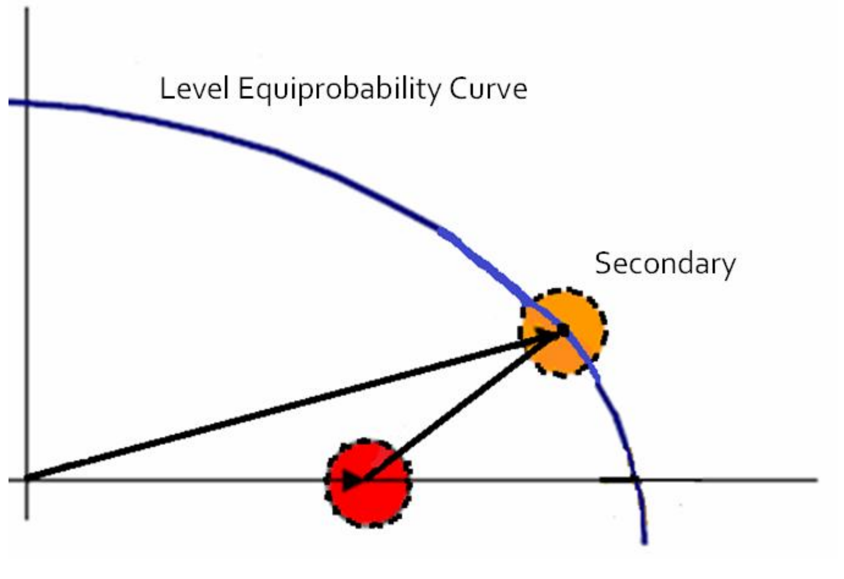


Figure 3.4: Illustration of the probability gradient approach: the primary moves along the negative gradient in the probability space, resulting in the shortest path out and away from the predicted collision ellipsoid

3.2 Strategies and Closed-Form Solutions

The given inputs are the estimated positions and velocities of the primary and secondary object at time t, and the covariance matrices of the two objects. We combine the individual covariance matrices to get σ_x and σ_z where the variables x and z represent the Keplerian planes in which our objects are located. Although our data usually comes in ECI coordinates and our final result is reported in ECI coordinates as well, for our analysis and probability calculations it is more convenient to convert all coordinates into RTN form (see Section 2.3).

3.2.1 Miss Vector Approach

We begin to discuss the first strategy that we call the miss vector approach. Compare Figure 3.3 and recall that the miss vector is that vector that stretches from the primary to the secondary at the time of closest approach. In this method, we seek to increase the magnitude of this vector and and hence solve the maneuver problem under the constraint that the change in the miss vector due to the burn must be collinear with the original miss vector. To accomplish this, we perform a delta-v maneuver that directs the primary in the negative miss vector direction with the effect of placing the secondary on a lower-level equiprobability curve as it intersects the encounter plane. In effect, this thereby reduces our probability of collision.

Angle of Burn to Increase the Miss Vector

In order to burn in the negative miss vector direction, we must first find the angles for the burn. Let

$$\hat{r} = (\hat{r}_R, \hat{r}_T, \hat{r}_N) \tag{3.1}$$

be the unitized ECI vectors for the primary at the time of closest approach in the RTN frame (using Radial, Transverse, and Normal coordinates), and

$$(\hat{r}_{rel}, \hat{v}_{rel}, \hat{h}_{rel}) \tag{3.2}$$

be the unitized ECI vectors for the relative miss vector, relative velocity vector, and their cross product (the previously defined basis vectors for our encounter frame). Let θ and ϕ respectively be the in-plane and out-of-plane angle of burn (for our purposes, $\phi = 0$). The following terms are the orbit elements for the primary object, where the subscript b refers to the trajectory *before* the burn whereas the subscript c refers to the desired time of the solution (i.e., the time of the original conjunction):

$$\begin{aligned}
P_b &= \text{semi-latus rectum before burn;}\\
u_b &= \text{argument of latitude before burn;}\\
\xi_b &= \text{eccentricity} \cdot \text{cos(arg. of perigee) before burn;}\\
\eta_b &= \text{eccentricity} \cdot \text{sin(arg. of perigee) before burn;}\\
n_b &= \text{mean motion before burn;}\\
h_b &= \text{angular momentum magnitude before burn;}\\
r_{bc} &= \text{magnitude of position vector before burn at the time of conjunction;}\\
t &= \text{time before conjunction when burn is applied;}
\end{aligned}$$

We also let C_{ubc} and S_{ubc} be the cosine and sine of the argument of latitude at the time of conjunction before the burn is applied, and we let C_{ub} and S_{ub} be the cosine and sine of the argument of latitude at the time of the burn whose angle can be found through a rotation backwards in time:

$$\begin{aligned}
S_{ub} &= S_{ubc}\cos(n_b t) - C_{ubc}\sin(n_b t)\\
C_{ub} &= C_{ubc}\cos(n_b t) + S_{ubc}\sin(n_b t).
\end{aligned} \tag{3.3}$$

Finally, we let

$$\begin{aligned}
a =& \left[\frac{r_{bc}^2}{h_b}\sin(n_b t)\right]\\
b =& -3\left(\xi_b S_{ub} - \eta_b C_{ub}\right)t\\
c =& \frac{r_{bc}^2}{h_b}\left[2 - \xi_b C_{ubc} - \eta_b S_{ubc} - \left(2 - \xi_b C_{ub} - \eta_b S_{ubc}\right)\cos(n_b t)\right]\\
&- 3(\xi_b S_{ubc} - \eta_b C_{ubc})t\\
d =& -3t\left(\frac{1 - \xi_b C_{ubc} - \eta_b S_{ubc}}{1 - \xi_b C_{ub} - \eta_b S_{ub}}\right)
\end{aligned} \tag{3.4}$$

be a group of functions that depend on the orbital elements of the primary at some specific time before we initiate our delta-v maneuver. It can be shown that this setup is sufficient to find the angles of our maneuver [24]:

$$\begin{aligned}
\sin\theta_{mv} &= \pm\frac{c(\hat{r}\cdot\hat{r}_{rel}) + d(\hat{r}_T\cdot\hat{r}_{rel})}{[K_{mv}]^{1/2}} \\
\cos\theta_{mv} &= \pm\frac{a(\hat{r}\cdot\hat{r}_{rel}) + b(\hat{r}_T\cdot\hat{r}_{rel})}{[K_{mv}]^{1/2}} \\
\tan\phi_{mv} &= \pm\frac{e(\hat{r}_N\cdot\hat{r}_{rel})}{[K_{mv}]^{1/2}}
\end{aligned} \tag{3.5}$$

where

$$K_{mv} = (a^2+c^2)(\hat{r}\cdot\hat{r}_{rel})^2+(b^2+d^2)(\hat{r}_T\cdot\hat{r}_{rel})^2+2(ab+cd)(\hat{r}_T\cdot\hat{r}_{rel})(\hat{r}\cdot\hat{r}_{rel}). \tag{3.6}$$

This in turn gives us our maneuver vector as

$$\Delta\vec{V} = (\Delta V_R, \Delta V_T, \Delta V_N) \tag{3.7}$$

where

$$\begin{aligned}
\Delta V_R &= \Delta V\cos\theta\cos\phi \\
\Delta V_T &= \Delta V\sin\theta\cos\phi \\
\Delta V_N &= \Delta V\sin\phi
\end{aligned} \tag{3.8}$$

are the radial, transverse and normal components of the maneuver.

Magnitude of Burn in Miss Vector Approach

Now that we know the direction in which we move the primary, we must determine how far we need to move. Ideally, to minimize the magnitude of the burn and thus limit the amount of fuel necessary, we would like to go just far enough to achieve the maximally permissible probability of collision threshold. The key idea by Peterson [24] is the observation that the coordinates of the miss vector change in the encounter plane due to the burn must match the boundaries of the equiprobability ellipse, which allows us to find the magnitude of the maneuver burn as follows.

First, we set the equation for our equiprobability ellipse equal the coordinates of the miss vector change in the encounter plane:

$$(x_T, z_T) = (x_C, 0) + (\Delta\dot{\vec{r}}r_{rel}, \Delta\dot{\vec{r}}h_{rel}). \tag{3.9}$$

Then we can substitute this into our equation for the desired equiprobability curve:

$$\begin{aligned}&\left(\frac{x_T}{\sigma_x}\right)^2 - 2\rho_{xz}\left(\frac{x_T z_T}{\sigma_x \sigma_z}\right) + \left(\frac{z_T}{\sigma_z}\right)^2 \\ &= \left(\frac{x_C}{\sigma_x}\right)^2 - 2(1-\rho_{xz}^2)\ln\left(\frac{P_T}{P_C}\right) \equiv K_C.\end{aligned} \tag{3.10}$$

This yields the quadratic equation

$$A\Delta V^2 + B\Delta V + C = 0 \tag{3.11}$$

where

$$\begin{aligned} A &= \frac{(\Delta\tilde{r}\cdot\hat{r}_{rel})^2}{\sigma_x^2} + \frac{(\Delta\tilde{r}\cdot\hat{h}_{rel})^2}{\sigma_z^2} - \frac{2\rho_{xz}(\Delta\tilde{r}\cdot\hat{r}_{rel})(\Delta\tilde{r}\cdot\hat{h}_{rel})}{\sigma_x\sigma_z} \\ B &= \frac{2x_C}{\sigma_x}\left[\frac{(\Delta\tilde{r}\cdot\hat{r}_{rel})}{\sigma_x} - \frac{\rho_{xz}(\Delta\tilde{r}\cdot\hat{h}_{rel})}{\sigma_z}\right] \\ C &= \frac{x_C^2}{\sigma_x^2} - K_C = 2(1-\rho_{xz}^2)\ln\left(\frac{P_T}{P_C}\right). \end{aligned} \tag{3.12}$$

From here, we can use any suitable numerical method such as the secant method to find the solution.

3.2.2 Probability Gradient Approach

The second strategy already indicated in Figure 3.4 is the probability gradient approach. In this case we move the primary in the direction of the probability gradient in order to place the secondary onto the desired lower level equiprobability curve in the most direct way possible. Clearly, this maneuver will typically be different than the one resulting from the miss vector approach. The direction in which we choose to thrust is determined by maximizing the dot product of the change in the miss vector resulting from the burn, and the gradient vector.

Angle of Burn in Probability Gradient Approach

Using the same notation as before, the negative gradient of the collision probability in equation (2.58) at the original closest approach is given by the following expression [24]:

$$\Delta P_c(x_c, 0) = \frac{-P_c x_c}{\sigma_x^2(1-\rho_{xz}^2)} \left(\hat{r}_{rel} - \rho_{xz}\frac{\sigma_x}{\sigma_z}\hat{h}_{rel} \right). \tag{3.13}$$

To force a change in this direction, the angle of the required burn is:

$$\sin\theta_{sg} = \frac{\pm D_2}{[{D_1}^2 + {D_2}^2]^{1/2}} \tag{3.14}$$

where

$$\begin{aligned} D_1 &= (a\hat{r} + b\hat{r}_T)\cdot(\hat{r}_{rel} - \rho_{xz}\frac{\sigma_x}{\sigma_z}\hat{h}_{rel}) \\ D_2 &= (c\hat{r} + d\hat{r}_T)\cdot(\hat{r}_{rel} - \rho_{xz}\frac{\sigma_x}{\sigma_z}\hat{h}_{rel}). \end{aligned} \tag{3.15}$$

Magnitude of Burn in Probability Gradient Approach

The size of the burn is again found so that the point of closest approach is at the threshold probability. This magnitude is the solution of the equation

$$A\Delta V^2 + B\Delta V + C = 0 \tag{3.16}$$

where

$$\begin{aligned} A &= \frac{(\Delta\tilde{r}\cdot\hat{r}_{rel})^2}{\sigma_x^2} + \frac{(\Delta\tilde{r}\cdot\hat{h}_{rel})^2}{\sigma_z^2} - \frac{2\rho_{xz}(\Delta\tilde{r}\cdot\hat{r}_{rel})(\Delta\tilde{r}\cdot\hat{h}_{rel})}{\sigma_x\sigma_z} \\ B &= \frac{2x_C}{\sigma_x}\left[\frac{(\Delta\tilde{r}\cdot\hat{r}_{rel})}{\sigma_x} - \frac{\rho_{xz}(\Delta\tilde{r}\cdot\hat{h}_{rel})}{\sigma_z}\right] \\ C &= 2(1-\rho_{xz}^2)\ \ln\left(\frac{P_T}{P_C}\right) \end{aligned} \tag{3.17}$$

and

$$\Delta\tilde{r} = (a\cos\theta' + c\sin\theta')\hat{r} + (b\cos\theta' + d\sin\theta')\hat{r}_T. \tag{3.18}$$

Here note that equations (3.17) are identical to (3.12) and differ only in the value of θ', that is either the angle from the steepest gradient in equation (3.14), or the angle from increasing the miss vector in equation (3.5) of the previous approach, respectively.

3.3 Discussion of Implementation

We ran two simulations with our data titled Opt3 and TestOpt3. Opt3 is the original implementation by Peterson [24] and uses real data supplied by a satellite operator. This includes basic and important pieces of data such as the current position and velocity. The code that we have written ourselves, called TestOpt3, is our own test case and was used to ensure we are getting reasonable numbers. It uses a variety of other codes written for this project, including the Kepler conversion, programs to switch between ECI and RTN frames, the probability of collision calculator, and the time of closest approach finder. These codes are included in Appendix C.2.2.

To run a simulation in TestOpt3 we need to make several assumptions to get us started. Without real data supplied by a satellite tracker, we create our own data such as the position and velocity of the two satellites. We assume that the position and velocity of the primary and secondary are similar because then there is a higher probability of collision. However, we act as if that we do not know the time of closest approach because it will be given, but we chose the time span that we wish to monitor. Once we have assigned a time period we run our simulation backwards in time. This allows us to look at a large spans of time in which the two objects have passed by each other while orbiting.

The resulting output we are looking at in Figures 3.5, 3.6 and 3.7 are the size of the delta-v and the angle in which we should thrust our satellite. These two quantities are basically the magnitude and the radial direction of where we like to move the primary, out of the path of the approaching secondary.

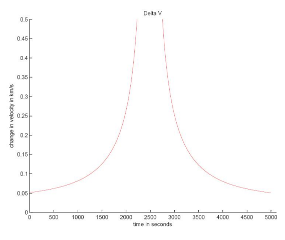


Figure 3.5: Magnitude of the optimal delta-v maneuver versus time of burn

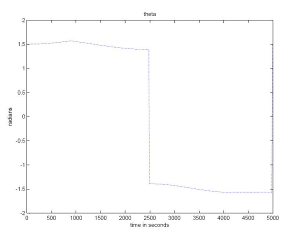


Figure 3.6: Angle of the optimal delta-v maneuver versus time of burn

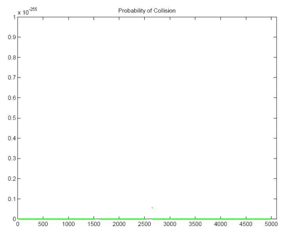


Figure 3.7: Reduced collision probability after the optimal delta-v maneuver

Chapter 4

Implementation

To validate the optimization strategies developed in Chapter 3, a significant amount of time was spent to implement a basic "trade space tool" that can compare a specific delta-v maneuver against an enumerative collection of alternative maneuvers. Given an initial, high probability of collision between two objects, we assume that only one of the objects (the primary) is under the control of a satellite operator whereas the secondary object, possibly a piece of space junk, is under no such control. The goal of the trade space tool is to simulate the effects that various choices of Δv applied to the primary object at different times have on the probability of collision.

In order to design an actual implementation of the given problem, several mathematical principles have to be understood and utilized. As described in some detail in the previous chapters, we decided to break the problem into different pieces where each group had to prepare and translate relevant information into mathematical formulas. Clearly, decomposing real-world situations into a mathematical formulation does not only require a good working knowledge of basic mathematical principles and modeling techniques, but it also requires a solid understanding of the assumptions and boundaries that make these models work or fail. Specifically, our program employs three basic concepts:

- properties of vectors from linear algebra and analytic geometry;
- first-order systems of ordinary differential equations;
- local linearization and Lagrange interpolation.

Before we explain the tools and structure of the code it is important to reemphasize some of the adopted assumptions. Hence, let us now turn our attention to the reasoning behind the code and only later discuss how the code actually works.

4.1 General Assumptions

To find a practical set of solutions certain assumptions must be made or met. For the most part these assumptions are not directly mathematical in nature but rather based on the information that is given in order to yield a relevant output.

1. The first major assumption is that we only have control of one object, the primary object. Some cases could occur where communication with another party would yield a solution, but it is not practical to make that assumption. Many collisions happen between operating satellites and space junk. For example, the 2009 Cosmos-Iridium collision occurred between an operable and a non-operable satellite. In that case, the only way to avoid a collision is by moving the primary object.
2. The second major assumption is that after a maneuver the primary object must stay inside its mission box. The mission box is defined as the allowable range of new orbits in order to maintain the utility of the satellite. An extension of this idea is that we assume that all solutions are coplanar. In other words, any new orbit must share the same plane as the old orbit. There can be instances where a non-coplanar maneuver is optimal but it is considered as a last resort because non-coplanar maneuvers involve more unknowns.
3. The third major assumption is that the covariances of the objects are available. In practice, these are usually calculated by the Air Force and included in the time-history data of the objects. When the Air Force releases time histories the covariance is part of the information in order to maintain a high standard and deliver good-quality, workable data.

4.2 The Trade Space Tool

For every given time t and every value of Δv included in our trade space, we need to compute the resulting new times of closest approach and the associated new collision probabilities. In order to do this, the trajectory of the primary object must be altered. This is done by changing the velocity. Once a given time for applying some delta-v maneuver has been established, the changes in velocity are similar to Hohmann transfers with the exception that there is only one burn. This is justified because our changes in velocity are very small, in the order of centimeters per second, whereas the velocity of the object that is measured in kilometers per second is typically much larger. Hence, if Δv is small, then the shape of the original orbit will usually not change dramatically and thus no correction is needed later for the satellite to stay within its mission box. As discussed in Section 2.2.3, it is also assumed that the changes in velocities occur instantaneously so that no additional adjustment for acceleration becomes necessary (with the exception of acceleration due to pull of gravity). In summary, only one very small burst or burn is applied virtually instantaneously to change the velocity at a given point.

Following the above discussion together with the background on Hohmann transfers provided in Section 2.2.3 and on trajectory optimization in Chapter 3, it follows that each delta-v maneuver is uniquely characterized as a combination of the following three quantities:

- the time at which the delta-v maneuver is executed;
- the angle in which the delta-v maneuver is applied;
- the magnitude or size of the delta-v maneuver burn.

The trade space tool that we describe here allows to select a finite set of possible times and a finite set of possible values for ΔV, and then compute the resulting collision probabilities in search of an acceptable combination. Moreover, at any given point of time it also allows to validate the optimization strategies from Chapter 3, by including those values of ΔV corresponding to the miss vector approach or the probability gradient approach.

4.2.1 Data Flow

Note that the above search space is four-dimensional in both time and space, in principle, but can be reduced to merely two dimensions if we require that the new orbit stays co-planar with the old orbit. To do this, we need to ensure that the direction of our delta-v points into the (tangential) direction of velocity. Therefore, we switch all given position data from Cartesian vectors into the VNB frame. Similar to the RIC frame, the VNB frame takes

Cartesian coordinates and changes them to vectors in a tangential direction (V), the normal direction (N), and the bi-normal direction (B).

- The (V) component is computed directly from the given velocities of the object.
- The (B) component is obtained as the cross product of the V component with the position vector. By definition, it is orthogonal to any other vector that lies in the plane of the velocity and position vectors.
- The (N) component is defined as the cross product of the velocity vector (V) and the bi-normal vector (B).

The velocity vector and the bi-normal vector define a new plane which is orthogonal to velocity and position plane and runs along the velocity direction. Hence, a change in the angle of the tangential direction can be represented by a right triangle in the (V,N) plane. The directions corresponding to the x-axis and the y-axis are defined as the tangential velocity of the object and the normal direction, respectively. The resulting Cartesian coordinates are in the same plane as the original orbit, and any addition of two vectors in the same plane will also be in this plane.

Let us define the set of possible delta-v maneuvers as a set of pairs

$$(t_i, \Delta v_j) \text{ for } i = 1, 2, \ldots, m \text{ and } j = 1, 2, \ldots, n \tag{4.1}$$

where $t_i \in [t_{\min}, t_{\max}]$ are given points of time in the trade space time interval $[t_{\min}, t_{\max}]$ indexed over i, and $v_j \in [\Delta v_{\min}, \Delta V_{\max}]$ are given scalar values in the trade space velocity interval $[\Delta v_{\min}, \Delta v_{\max}]$ that can be applied to the velocity vector associated with nominal state of the object at that time. By default, the program divides the specified trade range intervals into $m = n$ equidistant values for t and Δv, where the value of n can be chosen by the user. The corresponding values for delta-v are then added to the velocity of the object at this time, resulting in a new velocity and thus affecting both the trajectory of the primary object, and the probability of collision with any secondary objects.

Specifically, for each trade space pair $(t_i, \Delta v_j)$, we can then compute the associated critical encounters $k = 1, 2, \ldots$ with their corresponding times, miss distances, and collision probabilities p_{ijk}. To combine all probabilities into a single risk measure, we define the aggregated probability of each trade space pair as the probability of at least one collision:

$$P_{ij}^{\text{agg}} = 1 - \prod_k (1 - p_{ijk}) \text{ for } i = 1, 2 \ldots, m \text{ and } j = 1, 2, \ldots, n. \tag{4.2}$$

We briefly summarize the inputs and outputs of this trade space tool in Section 4.2.2, and then focus in a little bit more detail on the propagation of the modified trajectories in Section 4.2.3. The computation of encounters and collision probabilities is handled by the separate `collision module` subroutine that we review in additional detail in Section 4.3.

4.2.2 Inputs and Outputs

As indicated above, the initial input data consists of the states of the objects, and the trade space intervals of time and velocity.

- The states of the objects are described by time histories of their positions and velocities, together with covariance matrices that describe their errors in position.
- The trade space for the primary object is a range of possible values for Δv and a set of times at which these maneuvers can be applied.

All of this data can either be entered manually or read into the program from a set of files.

The goal of the program is to compare several different time and changes in velocity and establish a family of solutions to reach an acceptable level of risk. Hence, the output of the program gives a list of the aggregated probabilities P_{ij}^{agg} defined in Equation (4.2) whereas the full results are not displayed by default but stored separately for later retrieval, if desired. In particular, these full results also include the individual encounters $k = 1, 2, \ldots$ with their corresponding time and Δv as well as all individual TCAs, miss distances, and individual probabilities p_{ijk}. The aggregated probabilities of collision are also visualized in a two-dimensional contour to facilitate an intuitive interpretation and their further analysis.

4.2.3 Propagation Versus Interpolation

After the initial data input, the time-history files are continuously used to either look up or compute the expected states of the two objects at the time of the proposed Δv. Since the time history file only include observations at discrete times, it is clear that we need to find intermediate observations using different means. This can be done by one of three different methods, in principle, including propagation from a single close-by observation or by using some sort of interpolation.

Numerical Propagation using Differential Equation Solvers

An easily implemented approach to propagate an object out (or back) in time is to use the two-body differential equation derived in equation (2.13). This equation relates acceleration of an object with the position of that object at a given time and gives a simplified description of the orbit of an object around the Earth. The benefit of this method is that it is quite simple to implement, as explained for Homework Assignment 1 in Appendix B. On the negative side, this method is often not very accurate:

- The orbit of an actual object is nearly always much different than what is obtained from this simplified equation.
- Whereas Newton's two-body equation assumes spherical objects with uniform densities, the shape of the Earth is not a perfect sphere but an an oblate spheroid that is flattened near the poles and fattened around the equator. Because of this, and due to the rotation of the Earth more mass centers exist around the equator so that the gravitational force at each point is not constant.
- The gravitational force exerted by the Earth onto each point in space is also not constant. This will cause the trajectory of the satellite to vary slightly in both velocity and position.
- Atmospheric drag is another factor that the two-body equation does not account for. Very small quantities of particles are being released by the Earth's atmosphere. The collision of these particles with the satellite will cause a small amount of drag. Although the drag has a very insignificant effect on the velocity and position of the satellite, it will slightly alter the actual position.

Hence, a numerical propagation using the two-body equation is only acceptable for propagating very small time spans, or when no other, more accurate methods are available or computationally feasible.

Numerical Propagation using Lagrangean Interpolation

As indicated above, a numerical propagation technique using the two-body equation and the resulting set of differential equations is not one of the most accurate methods. Because the two-body equation simplifies the problem greatly and the new trajectory is estimated from only a single other point, this method is often very sensitive and alternative, more robust methods of estimating trajectories are generally preferably.

Interpolation is a method of estimating intermediate data points from a set of given observations or control points. The method of interpolation we use is Lagrangean interpolation. Lagrangean interpolation uses Lagrangean polynomials that form a least-degree polynomial that will run though the control points of the given data set. The interpolation is the sum of basis polynomials. A basis polynomial is a polynomial of smallest degree that runs through a control point with the property of being equal to zero at any of other given control points. For example, an interpolation between four points would have four scaled basis polynomials. Each of these basis polynomial would run through their own single control point and be a polynomial of degree three. In general, a basis polynomial would be of degree one less than the number of points, because it equals zero at any other control point of the interpolation. The final polynomial is the sum of all these polynomials and will constitute the least-degree fit for the given points in the data set.

Not difficult to implement, the default method used in our program is the interpolation method. Using the time histories of two different objects we can interpolate the data to estimate the state of the object at any given point using the points surrounding that point. This is more favorable than an ODE solver:

- The estimation is based on several points rather than just one.
- The estimation is better localized based on points in close proximity to the actual time of interest.

Using the time history files of the objects, the first step is to calculate the orbital periods. These periods are not consistent and can vary by a couple of minutes each orbit. Therefore, the program calculates several of these periods and then takes their average. The next step is the interpolate the data. Normally, the time history includes readings (states of the objects) at specific time intervals, which can be anywhere from 5 to 20 minutes. Interpolation produces new points between these existing readings, essentially fitting a curve to the data. This enlarged set of data combined with the calculated orbital period results in an estimation of the orbital trajectory of the two satellites. Now, given a future time, the program can estimate the states of the two objects at that time.

Important Note: Despite the advantages of interpolation over propagation, it is important to note that in order to estimate the new trajectory of the primary object after a delta-v maneuver, there is no other choice but to use the ODE propagation in order to find approximate positions and velocities of the primary object. Because this is a new trajectory, a time-history does not exist and interpolation is not possible. Hence, in order to calculate the probability of collision between the secondary object and the new trajectory of the primary object, an ODE solver must be used. Specifically, the approach we implemented uses Newton's two-body equation (2.15) and MATLAB's ode45 solver.

Analytic Propagation using Closed-Form Representations

One last method that we briefly consider is the use of an analytic closed-form expression to represent the trajectories in their entirety. For this method Keplerian elements can be used, in principle, as discussed in Section 2.3.3. Unfortunately, although this methods may produce the most accurate results, analytic expression are typically very difficult to obtain. The major challenge lies in the large amount of data required and the long time histories of satellite observations needed to calculate the periodicity of the orbit in terms of Keplerian elements. If there is not sufficient data, then this calculation is unreliable. Furthermore, because in our case the size of the time history files varies from one satellite to the other, an analytic propagation is unlikely to give a practical method and is not used in our implementation.

4.3 The Collision Module

As discussed in Section 4.2, we simulate a delta-v maneuver applied to the primary object by computing a new time history using a trajectory propagation based on the two-body equation. We spent a significant amount of time in writing a working code that compares two given time histories in order to find all times and points of closest approach together with their corresponding collision probabilities. Here we give a verbal overview of this major piece of code implemented. The full code of this routine, together with all its subroutines, is given in Appendix C.

4.3.1 Data Flow

As outlined above, the two key steps in the collision module are the computation of the times of closest approach, and the calculation of the new collision probabilities. The generic data flow is indicated in Figure 4.1.

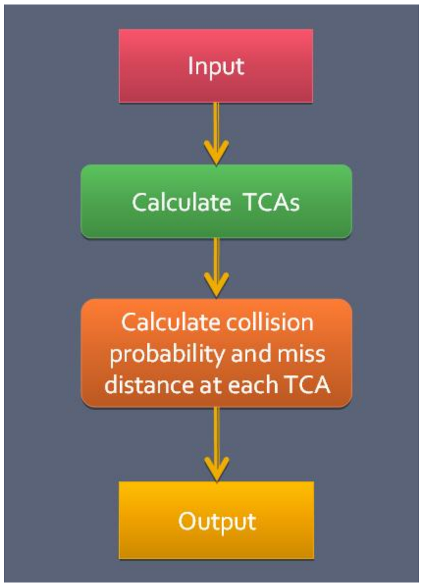

Input
Calculate TCAs
Calculate collision probability and miss distance at each TCA
Output

Figure 4.1: Generic data flow within the collision module

4.3.2 Inputs and Outputs

The initial input data consists primarily of a nominal time of closest approach and the new states of the primary object after a given delta-v maneuver $(t_i, \Delta v_j)$, as well as the the original time histories of the secondary objects and possibly a covariance matrix for each object that describes the error in position. The main output of this routine is the single aggregated probability P_{ij}^{agg} defined in equation (4.2), whereas the full results are stored separately for later retrieval, if desired. As before, these full results also include the individual encounters $k = 1, 2, \ldots$ with their corresponding time and Δv as well as all individual TCAs, miss distances, and individual probabilities p_{ijk} for fixed values of i and j.

As an optional input parameter we can also consider an additional probability threshold that allows to ignore especially small probabilities for which an encounter is not deemed critical. Note, however, that conceptually the computed aggregated probability is largely independent of this threshold as small (or zero) probabilities $p_{ijk} \approx 0$ do only very minor (or not) contribute to the aggregated probability value P_{ij}^{agg}.

4.3.3 Calculation of TCA and Collision Probabilities

The new time of closest approach (TCA) is computed using an iterative scheme that is successively refined using a sequential local linearization technique. It is based on the assumption that at times when two objects are close their relative trajectories can be estimated accurately linearly. The justification for this assumption is that these objects are moving very fast:

- Because of the speed of these objects moving in space the window of time for approaching a collision is very short.
- Because the window of time is very short, the only points we are dealing with are values that are close to the original position.

TCA Calculation

Since the Δv has changed the trajectory, it is now necessary to re-calculate the known nominal TCA for a new, actual TCA. The program first calculates the states of the two objects at the nominal TCA. While the interpolation method still works for the secondary object since its trajectory has remained unchanged, the velocity of the primary object is now different so that the interpolation method will no longer work without a valid time history. Hence, our only choice is to use the two-body equation and an ODE solver.

Once the states of objects at the nominal TCA are known, the new TCA can be obtained by finding the minimum of their new, relative distance. Let $r_p(t)$ and $r_s(t)$ be the positions of the primary and secondary object:

$$r_p(t) = r_{p0} + v_p t \text{ and } r_s(t) = r_{s0} + v_s t \tag{4.3}$$

where r_{p0} and r_{s0} be the initial positions of the two objects (in this case their position at the nominal TCA), and v_p and v_s be their velocities. Let $p(t)$ represent their miss distance that is defined as the distance between the two objects as a function of time:

$$\begin{aligned} p(t) &= r_p(t) - r_s(t) \\ &= r_{p0} + v_p t - r_{s0} + v_s t \\ &= (r_{p0} - r_{s0}) + (v_p - v_s)t \\ &= p_r + v_r t. \end{aligned} \tag{4.4}$$

Note that $(r_{p0} - r_{s0})$ is the relative position, denoted by p_r, and $(r_{p0} - r_{s0})$ is the relative position, denoted by v_r.

To find the time of closest approach, we need to minimize the norm, or equivalently the squared norm of the relative position vector. This is an unconstrained optimization problem in a single variable and can be solved easily by computing the gradient derivative, setting it equal to zero, and then solving it for time t. First, we note that

$$\|p(t)\|^2 = p^T p = (p_r + v_r t)^T (p_r + v_r t) = p_r^T p_r + 2p_r^T v_r t + t^2 v_r^T v_r. \tag{4.5}$$

The minimum of this function is where the derivative with respect to time vanishes:

$$\frac{d}{dt}\|p(t)\|^2 = 2p_r^T v_r + 2t v_r^T v_r = 0. \tag{4.6}$$

Solving this expression for t yields the desired actual TCA:

$$t = -\frac{p_r^T v_r}{v_r^T v_r}. \tag{4.7}$$

Because our goal is to find that time t when the derivative of the relative position function is zero, the only thing that matters is the numerator. If $p_r^T v_r = 0$, then we know that the relative position vector is orthogonal to the relative velocity vector, which therefore defines the time of closest approach. This suggests the following iterative two-step procedure:

1. To find the new TCA, the value for t in the above formula is calculated at the (current) nominal TCA.
2. If the value is not 0, then it is added to the nominal TCA and the calculation is re-done at that new time.

The program repeats this procedure until the value for t is less than some specified threshold ϵ. At this point, the TCA as well as the states of the objects can be computed.

Figures 4.2, 4.3, and 4.4 illustrate the trajectories, distances and miss distances between two randomly selected objects. As to be expected, we can see that the intersections of the graphs in Figure 4.3 coincide with those times in Figure 4.4 where the respective miss distances are approaching zero.

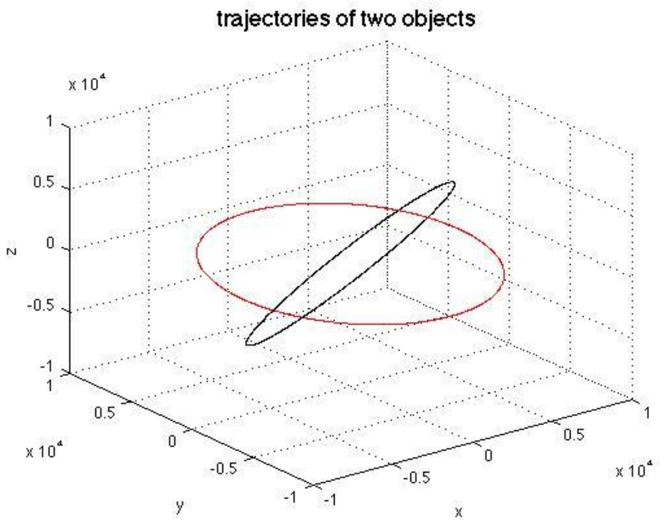


Figure 4.2: Illustration of miss distances: two object trajectories

Calculation of Collision Probabilities

Finally, given the TCA and the corresponding states of the two objects from the close approach finder, our program calculates the probability of collision as already explained in Section 2.4.2. Starting from the time at which the Δv is applied, a separate function propagates the covariance matrix of the primary object to the new TCA. For simplicity, in our implementation the covariance of the secondary object is simply taken to be a scalar multiple of the new covariance of the primary object. These two covariance matrices, along with the states of the objects at the new TCA, form the input for our probability calculator. This calculator numerically estimates the value of the probability density function over a specified hard body radius. It also calculates the miss vector as the difference between the position vectors of the two objects in the RIC frame of the primary object, and returns its magnitude as the corresponding miss distance.

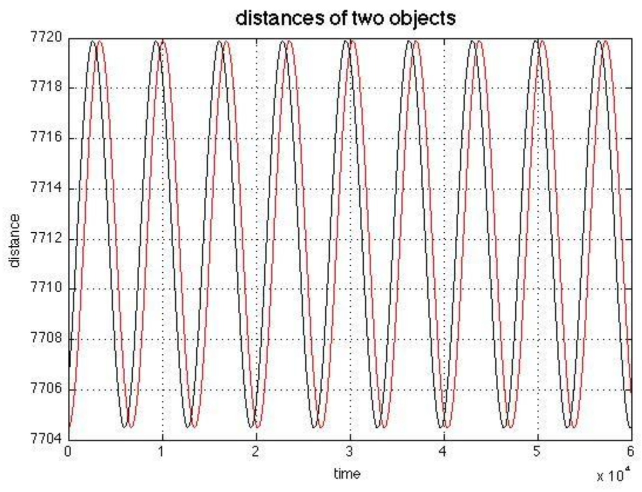


Figure 4.3: Illustration of the distances between two objects over time

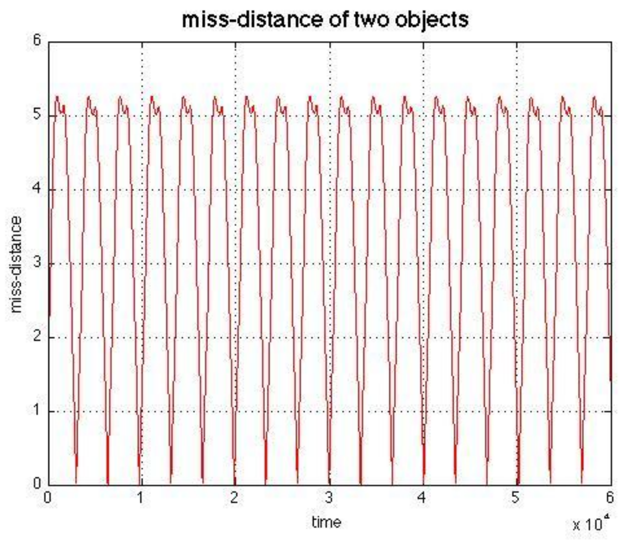


Figure 4.4: Illustration of the miss distances between two objects over time

Chapter 5

Experiment

In this chapter, we demonstrate how to apply and utilize our collision function to analyze the time history files for three satellites (primary, secondary-1, and secondary-2). The data is courtesy to Joshua Wysack from our sponsor SpaceNav and shown in Appendix C.3. For each data set, the starting date was 16-Jan-2005 02:14:37.212. The time history files contains ephemeris data at 5-minute intervals. Running the core function on the primary and secondary-1, we obtain the following output:

TCA	**Collision Probability**	**Miss Distance (km)**
17-Jan-2005 02:14:37.272	1.67E-05	0.192

The TCA is just over 1 day after the start of the file. The miss distance of 192 meters is just over the hard-body radius used in the probability calculator (which is 100 meters). Observe that if there were no uncertainty regarding the positions, the two objects would miss each other by approximately 92 meters. Yet, because of the uncertainty, we obtain a collision probability of 1.02×10^{-6}. Next we have the output for the primary and secondary-2:

TCA	**Collision Probability**	**Miss Distance (km)**
16-Jan-2005 04:49:25.410	3.37E-05	1.99E-04
16-Jan-2005 14:54:44.266	3.40E-05	1.05E-04
16-Jan-2005 15:44:36.136	2.22E-05	1.31E-05
17-Jan-2005 09:24:28.860	1.78E-05	1.39E-04
17-Jan-2005 19:29:47.745	1.95E-05	1.54E-04
17-Jan-2005 20:19:39.617	5.86E-05	2.93E-05

There is much more risk for these two objects. For each of the six close approach points, the calculated miss distance is less than 1 meter. Note again that the uncertainty in position is reflected in the collision probability, which hovers between 1.7×10^{-5} and 5.7×10^{-6}. The first TCA is just over 2 hours after the start of the data. This will make applying a Δv to avoid this collision difficult.

5.1 Trade Space Comparison

We want to generate some cases using values from a trade space. We set the trade-space interval for Δv to $[-5 \times 10^{-5}, 5 \times 10^{-5}]$, which is in cm/sec, and we set the trade-space range of times to $[.1, 1.1]$, which represents a fraction of a day from the beginning of the files. Figure 5.1 is a graph of the resulting aggregate probabilities:

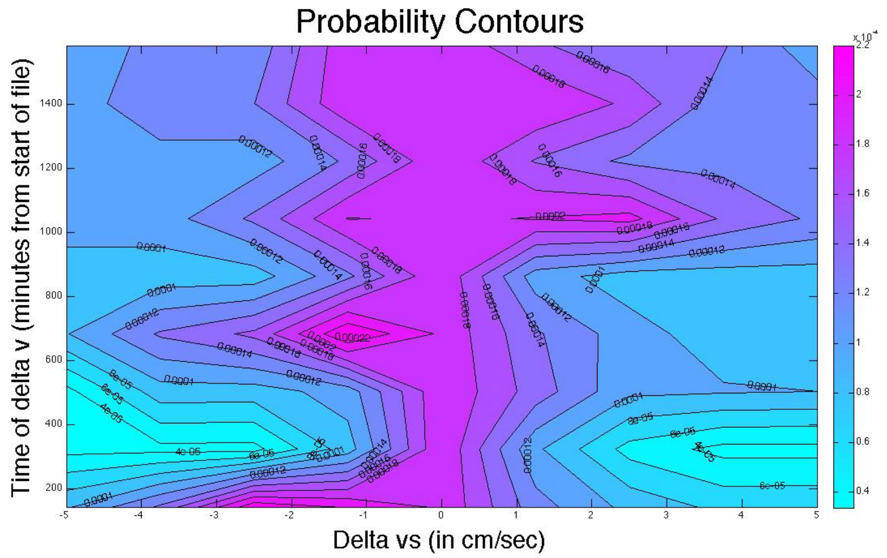


Figure 5.1: 2D contours of aggregate probabilities in full Δv trade space

The graph is somewhat symmetric about the point where no Δv is applied. Intuitively, this is how it should be. The lower probabilities are where the larger Δv is applied earlier. The later the Δv is applied, the more TCAs it misses, so the probability is higher.

5.2 Discussion of Results

Figure 5.2 is a closer look at the left side of the graph.

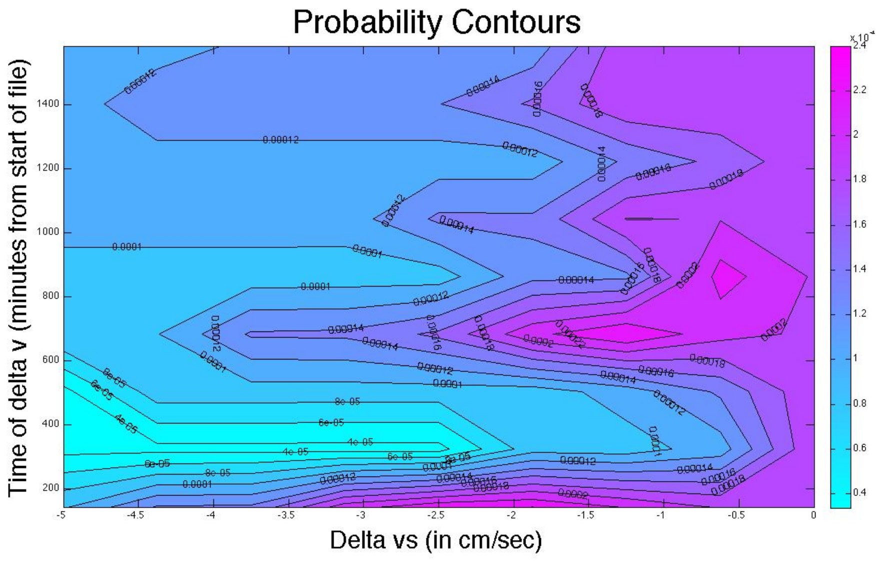


Figure 5.2: 2D contours of aggregate probabilities in negative Δv space

This closer inspection helps us see more specifically what is happening with these probabilities. We see to the left, where Δv's of -5 cm/sec are applied, that the probability starts higher and then jumps down a bit. Since the first TCA is right around this point, one would guess that the probability would raise once the Δv was applied after the TCA. The graph seems to suggest something else. Looking closely at this case, we obtain the following output, when a Δv of -5 cm/sec is applied at 144 minutes:

TCA	Collision Probability	Miss Distance (km)
16-Jan-2005 04:49:25.287	3.03E-05	0.067
16-Jan-2005 14:54:44.213	2.02E-05	0.943
16-Jan-2005 15:44:36.080	4.75E-05	1.037

We see that the first Δv managed to avoid the TCA with secondary-1 and the last three TCAs with secondary-2. It also changed the probabilities and miss distances with the first three TCAs with secondary-2. Now we look a bit later, where a Δv of -5 cm/sec was applied at 334 minutes:

TCA	**Collision Probability**	**Miss Distance (km)**
16-Jan-2005 04:49:25.410	3.37E-05	1.99E-04

This Δv was applied before the first TCA, and therefore did not change any of the associated numbers. It did avoid all the subsequent TCAs. Looking back at Figure 5.1) we see that the bottom right side is not the same as the bottom left side. We can compare the output from these points. Following is the output from when a Δv of 5 cm/sec was applied at 144 minutes:

TCA	**Collision Probability**	**Miss Distance (km)**
16-Jan-2005 04:49:25.533	1.78E-05	0.065
16-Jan-2005 14:54:44.309	3.51E-05	1.18
16-Jan-2005 15:44:36.158	2.24E-05	1.31

These results are very similar to when a Δv of -5 cm/sec was applied at the same time. The TCAs are all a few milliseconds different, and the miss distances are all a bit larger. The probabilities are all a tad smaller. This demonstrates some of the complexity of the problem. By simply changing the direction of the Δv, we obtain quite different output. We also see how fast the objects are moving. A difference in just a couple of milliseconds in TCA can result in many meters (even a kilometer) in miss distance.

Chapter 6

Conclusion

Collision avoidance of operational spacecrafts is a major concern for the aerospace industry, and a fascinating problem for learning and applying a broad array of mathematical areas and related disciplines:

1. Fundamentals of Orbital Mechanics and Astrodynamics
    - equations of motion in space (Newton laws of motion and gravity)
    - changing trajectories of space objects using delta-v maneuvers
2. Modeling and Propagation of Tracking Uncertainties
    - state error covariance matrices and state transition matrices
    - stochastic approximations based on Monte Carlo simulations
3. Elements of Numerical Analysis and Scientific Computing
    - numerical integration methods to calculate collision probabilities
    - analytic approximations of orbital elements and space trajectories
4. Overview of Optimization Techniques and Optimal Control
    - computing time and separation distances at close-approach points
    - finding optimal trajectory changes and making maneuver decisions

This report summarizes the work of 12 undergraduate and graduate students, that were advised by one faculty member and two industry representatives to learn about the problem, understand its implications, and begin the development and implementation of strategies that may eventually contribute to a solution of this challenging problem. Combining small pieces from each of the above areas, this project centered around determining strategies and finding maneuvers that minimize the risk of collision between space objects by altering their trajectories. The final goal of this clinic was to assist our sponsor SpaceNav in using optimization models and developing software to implement a practical collision avoidance method.

As undergraduate and graduate students in mathematics and physics, we were provided with the opportunity to tackle this problem and to design a system both for analyzing the risk of collision of a satellite with other objects in space, and for determining one or several trajectory maneuvers for the satellite to lower that risk. In this report, we describe our method that allows to analyze incoming data with satellite information to detect and subsequently provides a collection of maneuvers able to reduce an initial high risk of collision.

In order to accomplish this goal, we divided the problem into a number of parts and initially tackled these components separately. The preliminary information that we were given or that was assumed to be known was a table providing a primary satellite's position and velocity information, and a covariance matrix to account for the uncertainty in position that exists when tracking a satellite in orbit. There is also a list of all those satellites within a certain distance (a close encounter), together with their positions and covariance matrices.

Using this information, we apply a number of steps to find suitable maneuvers (review Figure 1.2):

1. First we convert the Cartesian coordinates into Keplerian coordinates.
2. Then we take the transforms and calculate the probability of collision between the primary satellite and the ones in close proximity.
3. If any probability exceeds a certain threshold, we specify a window of possible trajectory maneuvers to reduce the risk of collision between the two satellites. This is again done by a number of steps:

    (a) First we apply several changes in velocity from a trade space at several times before the time of closest approach.

    (b) Then we calculate a new time of closest approach for each velocity change and chosen maneuver time.

(c) With these new times of closest approach, we calculate new probabilities of collision with the primary satellite.

4. Now it is easy to find the best trajectory change for reducing the risk and recommend which delta-v maneuver should be implemented.

While the above strategy guarantees to find a best maneuver among those taken into consideration, it is inherently a search method and not an optimization method. Hence, as alternative, two additional strategies are described that are based on actual optimization:

- The first method is the "Miss Vector Approach" in which we move the primary satellite directly along the miss vector in direction opposite to the intersection of secondary object and the physical encounter plane.
- The second method is the "Probability Gradient Approach" in which we move the primary satellite along the negative gradient in the probability space, resulting in the shortest and presumably quickest path out and away from the predicted collision ellipsoid.

With closed-form solutions available, each approach is relatively quick in response which makes it possible to use all three strategies and compare their results, in principle. However, it remains hard to declare a winner and say which one is better because it seems to vary in different situations. A deeper analysis and more computational experiments are necessary to arrive at any additional conclusions.

List of References

[1] M. R. Akella and K. T. Alfriend. Probability of collision between space objects. *Journal of Guidance, Control and Dynamics*, 23(5):769–772, 2000.

[2] S. Alfano. Satellite conjunction Monte Carlo analysis. In *AIAA Space Flight Mechanics Meeting*, Paper AAS 09-233, February 2009.

[3] K. T. Alfriend, M. R. Akella, J. Frisbee, J. L. Foster, D.-J. Lee, and M. Wilkins. Probability of collision error analysis. *Space Debris*, 1(1):21–35, 1999.

[4] P. D. Anz-Meador. International guidelines for the preservation of space as a unique resource. *Online Journal of Space Communication*, 6, Winter 2004.

[5] R. R. Bate, D. D. Mueller, and J. E. White. *Fundamentals of Astrodynamics*. Dover Publications, 1971.

[6] N. Bérend. Estimation of the probability of collision between two catalogued orbiting objects. *Advances in Space Research*, 23(1):243–247, 1999.

[7] A. M. Bradley and L. M. Wein. Space debris: Assessing risk and responsibility. *Advances in Space Research*, 43(9):1372–1390, 2009.

[8] N. Bush. Unmodeled error analysis on trajectory and orbital estimation. *Technometrics*, 13(2):303–314, May 1971.

[9] M. Campbell. Collision monitoring within satellite clusters. *IEEE Transactions on Control Systems Technology*, 13(1):42–55, 2005.

[10] V. A. Chobotov. *Orbital Mechanics, Third Edition.* AIAA Education Series. The Aerospace Corporation, third edition, 2002.

[11] L. David. Space debris: a growing challenge. *Aerospace America*, 47(9):30–36, 2009.

[12] Euopean Space Agency. Multimedia Gallery (topic: operations, subtopic: space debris). Http://www.esa.int/esa-mmg/mmghome.pl, 2011.

[13] W. Hohmann. *Die Erreichbarkeit der Himmelskörper.* R. Oldenbourg, München, Berlin, 1925.

[14] W. Hohmann. The Attainability of Heavenly Bodies. [Translation of "Die Erreichbarkeit der Himmelskörper"]. NASA Technical Translation F-44, November 1960.

[15] T. S. Kelso. Iridium 33/Cosmos 2251 Collision (Coverage started March 5, 2009). Http://celestrak.com/events/collision, updated May 13, 2011.

[16] T. S. Kelso. Chinese ASAT Test (Coverage started January 19, 2007). Http://celestrak.com/events/asat.asp, updated May 20, 2011.

[17] A. Lambert, G. Saint-Pierre, and D. Gruyer. A Monte Carlo approach for collision probability computation. In *Workshop IROS 2008 on Perception, Planning and Navigation for Intelligent vehicles*, Nice, France, 26 September 2008.

[18] L. Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley Professional, second edition, 1994.

[19] J. F. Liu and J. F. Segrest. An analysis of orbit motion in frequency domain. Technical report, Directorate of Astrodynamics DCS/Operations, Peterson AFB, Colorado, March 1986.

[20] O. Montenbruck and E. Gill. *Satellite Orbits: Models, Methods and Applications.* Springer, 2000.

[21] J. B. Mueller and R. Larsson. Collision avoidance maneuver planning with robust optimization. In *GNC 2008, 7th International ESA Conference on Guidance, Navigation & Control Systems*, Tralee, County Kerry, Ireland, 2-5 June 2008.

[22] NASA National Space Sciece Data Center. NSSDC Master Catalog. Http://nssdc.gsfc.nasa.gov/nmc, version 4.0.16, 26 April 2011.

[23] B. Obama. National Space Policy of the United States of America. Http://www.whitehouse.gov/sites/default/files/national_space_policy_6-28-10.pdf, June 28, 2010.

[24] G. E. Peterson. Maneuver selection for probability reduction of near-circular orbit conjunctions. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Paper AIAA-2002-4630, Monterey, CA, 5-8 August 2002.

[25] R. Pratap. *Getting Started with MATLAB: A Quick Introduction for Scientists and Engineers*. Oxford University Press (USA), 2009.

[26] J. E. Prussing and B. A. Conway. *Orbital Mechanics*. Oxford University Press, USA, September 1993.

[27] B. Shapiro. Conversion between kepler and cartesian elements. Technical report, California State University CSUN/JPL PAIR Program, 2001.

[28] Y. Tsuda. State transition matrix approximation with geometry preservation for general perturbed orbits. *Acta Astronautica*, 68(7-8):1051–1061, April-May 2011.

[29] W. H. de Vries and D. W. Phillion. Monte Carlo method for collision probability calculations using 3D satellite models. Technical Report LLNL-CONF-454474, Lawrence Livermore National Laboratory, 2010.

Further Readings

[30] J. R. Alarcon, H. Klinkrad, J. Cuesta, and F. M. Martinez. Independent orbit determination for collision avoidance. In *Proceedings of 4th European Conference on Space Debris*, Darmstadt, Germany, 18-20 April 2005.

[31] S. Alfano. Method for determining maximum conjunction probability of rectangular-shaped objects. United States Patent, Patent No.: US 7,383,153 B2, Jun. 3, 2008.

[32] S. Alfano. Satellite collision probability enhancements. *Journal of guidance, control, and dynamics*, 29(3):588–592, 2006.

[33] K. Assmann, J. Berger, and S. Grothkopp. The COLA collision avoidance method. In *Proceedings of 5th European Conference on Space Debris*, Darmstadt, Germany, 30 March - 2 April 2009.

[34] R. W. Beard and T. W. McLain. Multiple UAV cooperative search under collision avoidance and limited range communication constraints. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, Maui, Hawaii, USA, December 2003.

[35] H. Blake. Space so full of junk that a satellite collision could destroy communications on earth. The Telegraph, 3 February 2011. Retrieved from http://www.telegraph.co.uk/science/space/8295546/Space-so-full-of-junk-that-a-satellite-collision-could-destroy-communications-on-Earth.html.

[36] F. K. Chan. Spacecraft maneuvers to mitigate potential collision threats. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Paper AIAA-2002-4629, Monterey, CA, 5-8 August 2002.

[37] F. K. Chan. *Spacecraft Collision Probability.* AIAA/Aerospace Press, 2008.

[38] K. Chan. Collision probability analysis for earth orbiting satellites. In *Proceedings of the 7th International Space Conference of Pacific-Basin Societies*, pages 1033–1048, Nagasaki, Japan, 15-18 July 1997.

[39] V. A. Chobotov. Classification of orbits with regard to collision hazard in space. *Journal of Spacecraft and Rockets*, 20:484–490, Oct 1983.

[40] J. C. Clements. The optimal control of collision avoidance trajectories in air traffic management. *Transportation Research Part B: Methodological*, 33(4):265–280, 1999.

[41] Commission on Engineering and Technical Systems (CETS). Collision avoidance. In *Protecting the Space Shuttle from Meteoroids and Orbital Debris*, chapter 5, pages 36–41. The National Academies Press, 1997.

[42] P. B. de Selding. Satellite collision avoidance methods questioned after space crash. Space News, 27 February 2009. Retrieved from http://www.space.com/2386-satellite-collision-avoidance-methods-questioned-space-crash.html on January 20, 2011.

[43] N. Fulton and T. Tarnopolskaya. Optimal cooperative collision avoidance strategy for coplanar encounter: Merz's solution revisited. *Journal of Optimization Theory and Applications*, 140(2):355–375, 2009.

[44] I. Garcia and J. P. How. Trajectory optimization for satellite reconfiguration maneuvers with position and attitude constraints. In *2005 American Control Conference*, Portland, OR, USA, 8-10 June 2005.

[45] D. Gaylor, R. Gottlieb, and S. Sponaugle. Satellite collision prediction and avoidance system. Prior Art Database, 1 May 2000. Retrieved from http://priorartdatabase.com/IPCOM/000009823.

[46] R. G. Gottlieb, S. J. Sponaugle, and D. E. Gaylor. Orbit determination accuracy requirements for collision avoidance. In *11th Annual AAS/AIAA Space Flight Mechanics Meeting*, Paper AAS-01-181, Santa Barbara, CA, February 2001.

[47] F. Y. Hadaegh, Y. Kim, and M. Mesbahi. Multiple-spacecraft reconfiguration through collision avoidance, bouncing, and stalemate. *Journal of Optimization Theory and Applications*, 122(2):323–343, 2004.

[48] International Space University. Space university report offers ways to avoid further satellite collisions. Retrieved from http://www.isunet.edu/index.php/news-mediacenter/566-space-university-report-offers-ways-to-avoid-further-satellite-collisions.

[49] D. J. Kessler. Derivation of the collision probability between orbiting objects: the lifetimes of Jupiter's outer moons. *Icarus*, 48(1):39–48, 1981.

[50] D. J. Kessler. Collision probability at low altitudes resulting from elliptical orbits. *Advances in Space Research*, 10(3-4):393–396, 1990.

[51] D. Kestenbaum. Chinese missile destroys satellite in 500-mile orbit. NPR, January 19, 2007. Retrieved from http://www.npr.org/templates/story/story.php?storyId=6923805.

[52] I. Klotz. A traffic cop for satellites. Discovery News, Space News, 1 September 2010. Retrieved from http://news.discovery.com/space/satellites-traffic-cop.html.

[53] E. Lalish. *Distributed Reactive Collision Avoidance*. PhD thesis, Department of Aeronautics and Astronautics, University of Washington, 2009.

[54] M. R. Lehto, J. D. Papastavrou, T. A. Ranney, and L. A. Simmons. An experimental comparison versus optimal collision avoidance warning system thresholds. *Safety Science*, 36(3):185–209, 2000.

[55] T. Malik. Debris from space collision poses threat to other satellites. Space News, 12 February 2009. Retrieved from http://www.space.com/5540-debris-space-collision-poses-threat-satellites.html on January 20, 2011.

[56] F. A. Marcos, M. J. Kendra, J. M. Griffin, J. N. Bass, D. R. Larson, and J. J. F. Liu. Precision low earth orbit determination using atmospheric density calibration. In *Proceedings of the AAS/AIAA Astrodynamics Conference*, pages 501–513, Sun Valley, ID, USA, 4-7 August 1997.

[57] F. A. Marcos, M. J. Kendra, J. M. Griffin, J. N. Bass, D. R. Larson, and J. J. F. Liu. Precision low earth orbit determination using atmospheric density calibration. *Journal of the Astronautical Sciences*, 46(4):395–409, 1998.

[58] J. B. Mueller. Onboard planning of collision avoidance maneuvers using robust optimization. In *AIAA Infotech at Aerospace Conference and Exhibit and AIAA Unmanned ... Unlimited Conference*, Seattle, WA, USA, 2009.

[59] K. Muinonen, J. Virtanen, and E. Bowell. Earth-crossing asteroids using orbital ranging. *Physics and astronomy celestial mechanics and dynamical astronomy*, 81(1-2):93–101, 2001.

[60] R. P. Patera. Satellite collision probability for non-linear relative motion. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Monterey, CA, 5-8 August 2002.

[61] A. Platzer and E. M. Clarke. Formal verification of curved flight collision avoidance maneuvers: A case study. Technical Report CMU-CS-09-147, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2009.

[62] A. Richards, T. Schouwenaars, J. P. How, and E. Feron. Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. *Journal of Guidance, Control, and Dynamics*, 25(4):755–764, 2002.

[63] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo Method.* A John Wiley & Sons, Inc., 2007.

[64] N. Sanchez-Ortiz, M. Bello-Mora, and H. Klinkrad. Collision avoidance manoeuvres during spacecraft mission lifetime: Risk reduction and required Δv. *Advances in Space Research*, 38(9):2107–2116, 2006.

[65] D. P. Scharf, A. B. Acikmese, S. R. Ploen, and F. Y. Hadaegh. A direct solution for fuel-optimal reactive collision avoidance of collaborating spacecraft. In *Proceedings of the American Control Conference*, pages 5201–5206, 2006.

[66] J. A. Snyman. Introduction. In *Practical Mathematical Optimization*, chapter 1, pages 1–31. Springer, 2005.

[67] Q. Tang, B. J. Pang, and W. Zhang. *Collision risk assessment for a spacecraft in space debris environment*, pages 721–724. Number 587 in (Special Publication) ESA SP. European Space Agency, 2005.

[68] R. J. Tayler. The rapid computation of Earth-satellite ephemerides by a method using Fourier series. Technical Note Space 61, Royal Aircraft Establishment Farnborough (UK), March 1964.

[69] M. Tillerson, G. Inalhan, and J. P. How. Coordination and control of distributed spacecraft systems using convex optimization techniques. *International Journal of Robust and Nonlinear Control*, 12(2-3):207–242, 2002.

[70] R. H. Vassar and R. B. Sherwood. Formation keeping for a pair of satellites in a circular orbit. *Journal of Guidance, Control, and Dynamics*, 8(2):235–242, 1985.

[71] S. Wang and H. Schaub. Spacecraft collision avoidance using Coulomb forces with separation distance and rate feedback. *Journal of Guidance, Control, and Dynamics*, 31(3):740–750, 2008.

[72] S. Wang and H. Schaub. Electrostatic spacecraft collision avoidance using piecewise-constant charges. *Journal of Guidance, Control, and Dynamics*, 33(2):510–520, 2010.

[73] Wikipedia. 2009 satellite collision. Retrieved from http://en.wikipedia.org/wiki/2009_satellite_collision.

[74] Wikipedia. Collision avoidance (spacecraft). Retrieved from http://en.wikipedia.org/wiki/Collision_avoidance_(spacecraft).

[75] Wikipedia. Delta-v. 15 November 2010. Retrieved from http://en.wikipedia.org/wiki/Delta-v on January 20, 2011.

[76] Wikipedia. Orbital maneuver. 19 January 2011. Retrieved from http://en.wikipedia.org/wiki/Orbitali_maneuver on January 20, 2011.

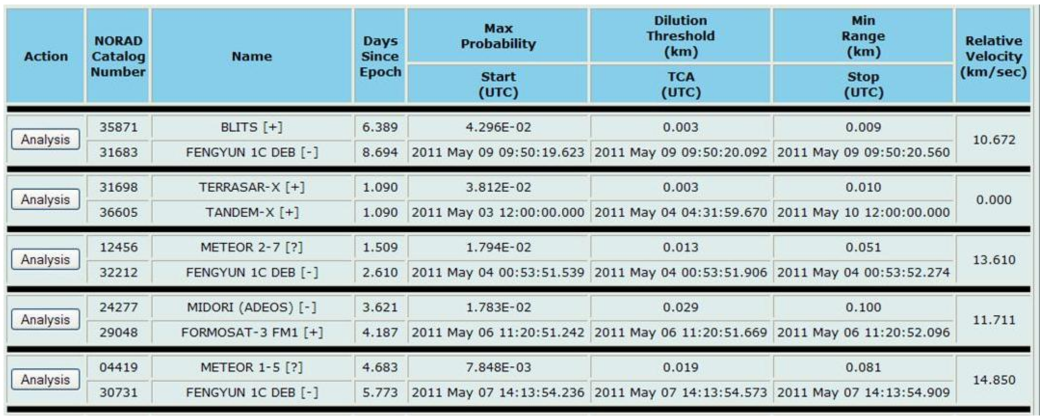
Appendix A

Other Web Resources

This appendix includes a brief list of additional resources that are available online and were explored or used for some additional background information.

GMAT (General Mission Analysis Tool) This software package seems to be one of the better examples of current technology that deals with satellite collision avoidance. It is a continuing, collaborative team project between NASA and the open industry. The system can be downloaded for free from the GMAT homepage at http://gmat.gsfc.nasa.gov/index.html. According to its mission statement, "the goal of the GMAT project is to develop new space trajectory optimization and mission design technology by working inclusively with ordinary people, universities, businesses, and other government organizations, and to share that technology in an open and unhindered way." Currently, the system can be used to analyze flight trajectories of satellites and to find the optimal change of trajectory given a certain set of parameters.

CelesTrak/SOCRATES (Satellite Orbital Conjunction Reports Assessing Threatening Encounters in Space) This service is provided by the Center for Space Standards & Innovation (CSSI) and accessible through http://celestrak.com/SOCRATES. It runs a list of all satellite payloads in orbit against a list of objects in orbit using the catalog of unclassified NORAD data and looks for weekly satellite conjunctions. The goal of SOCRATES is to provide satellite operators with a tool to assess the possible risks of collision between their payloads and other objects in orbit. The data is updated twice a day. Any data that could compromise U.S. security is omitted. SOCRATES takes data sorted for all conjunctions that will come within 1 km at the time of closest approach and reports the minimum distance and the maximum probability for each conjunction. A top ten risk sheet is then released, sorted by maximum probability and showing position, velocity, miss distance and TCA information. For example, the top five collision predictions reported at the beginning of May 2011 are shown in Figure A.1.



Action	NORAD Catalog Number	Name	Days Since Epoch	Max Probability	Dilution Threshold (km)	Min Range (km)	Relative Velocity (km/sec)
				Start (UTC)	TCA (UTC)	Stop (UTC)	
Analysis	35871	BLITS [+]	6.389	4.296E-02	0.003	0.009	10.672
	31683	FENGYUN 1C DEB [-]	8.694	2011 May 09 09:50:19.623	2011 May 09 09:50:20.092	2011 May 09 09:50:20.560	
Analysis	31698	TERRASAR-X [+]	1.090	3.812E-02	0.003	0.010	0.000
	36605	TANDEM-X [+]	1.090	2011 May 03 12:00:00.000	2011 May 04 04:31:59.670	2011 May 10 12:00:00.000	
Analysis	12456	METEOR 2-7 [?]	1.509	1.794E-02	0.013	0.051	13.610
	32212	FENGYUN 1C DEB [-]	2.610	2011 May 04 00:53:51.539	2011 May 04 00:53:51.906	2011 May 04 00:53:52.274	
Analysis	24277	MIDORI (ADEOS) [-]	3.621	1.783E-02	0.029	0.100	11.711
	29048	FORMOSAT-3 FM1 [+]	4.187	2011 May 06 11:20:51.242	2011 May 06 11:20:51.669	2011 May 06 11:20:52.096	
Analysis	04419	METEOR 1-5 [?]	4.683	7.848E-03	0.019	0.081	14.850
	30731	FENGYUN 1C DEB [-]	5.773	2011 May 07 14:13:54.236	2011 May 07 14:13:54.573	2011 May 07 14:13:54.909	

Figure A.1: Top 5 collision predictions by SOCRATES in early May 2011

Http://www.space-track.org The space-track website (which interested parties can apply for and be accepted as members) has plenty of data, although the site warns that it should not be used for conjunction probabilities. Nevertheless, this was one of the most useful of the various websites for finding data on satellite time histories.

Http://www.shatters.net/celestia This page provides a free space simulation software package to look at the universe from perspectives other than just Earth. There are many extra, add-on downloads as well. The website does a pretty good job explaining what it does, and although not particularly useful for our project it is still kind of cool.

Http://www.satobs.org/orbsoft.html This page offers several satellite-related software descriptions and links for many different operating systems.

Http://www.satobs.org/tletools.html Similar to the former, this service deals more with tracking software and other two-line element resources.

Appendix B

Homework Assignments

Math 4779/5779 Homework Assignment 1

Numerical Solution of ODE using Matlab

Due Tuesday, 1 February 2011

Newton's law of gravitation states that any two objects attract one another with a force proportional to their combined mass and inversely proportional to the square of the distance between them. This law can be expressed mathematically as

$$\boldsymbol{F} = -\frac{GMm}{r^2}\frac{\vec{\boldsymbol{r}}}{\|\vec{\boldsymbol{r}}\|} \equiv -\frac{GMm\vec{\boldsymbol{r}}}{r^3} \tag{1}$$

where $\boldsymbol{F}$ is the force on mass m due to mass M and G is a gravitational constant. Newton's law of gravity and Newton's second law can be combined to develop the so-called 'two-body' equation of relative motion. The two-body equation is given to be

$$\ddot{\vec{\boldsymbol{r}}} = -\frac{G(M+m)\vec{\boldsymbol{r}}}{r^3}. \tag{2}$$

For a space object orbiting the Earth, the equation above can further reduced to

$$\ddot{\vec{\boldsymbol{r}}} + \frac{\mu_E\vec{\boldsymbol{r}}}{r^3} = \boldsymbol{0}. \tag{3}$$

1. Discuss how this formulation could have been produced. What assumptions were made? Can you determine the definition of μ_E?

2. Equation (3) is a second-order vector differential equation. Express this equation as a first-order system.

3. Given the following initial conditions, numerically solve the differential system using Matlab's ODE45 solver for $t = 0$ to $t = 1000$ minutes. Produce a plot showing total position and total velocity as a function of time. ($\mu_E = 398600.5\,\text{km}^3/\text{sec}^2$, $[x, y, z] = [-2436.45, -2436.45, 6891.037]\,\text{km}$, $[\dot{x}, \dot{y}, \dot{z}] = [5.088611, -5.088611, 0.0]\,\text{km/sec}$)

Type your solutions using LaTeX and submit a zip file with all tex and graphics files needed for its compilation to the Drop Box on Blackboard. Please also bring a hard copy of the compiled pdf to class on Tuesday, February 1. Please ask if you have any questions!

Math 4779/5779 Homework Assignment 2

Uncertainty Propagation and Collision Probabilities using Monte Carlo Simulations

Due Tuesday, 8 February 2011

Recall from Assignment 1 that the 'two-body equation' for space objects orbiting Earth is given by

$$\ddot{\boldsymbol{r}} + \frac{\mu_E \boldsymbol{r}}{r^3} = \boldsymbol{0} \tag{1}$$

where the constant $\mu_E = 398600.5\,\mathrm{km}^3/\,\mathrm{sec}^2$ is the product of reduced mass and gravitational force of the Earth, $\boldsymbol{r} = [x, y, z]$ is the Cartesian position vector, $r = \|\boldsymbol{r}\|$ is the distance between the center of the Earth and the object, and $\dot{\boldsymbol{r}} = [\dot{x}, \dot{y}, \dot{z}]$ and $\ddot{\boldsymbol{r}} = [\ddot{x}, \ddot{y}, \ddot{z}]$ denote velocity and acceleration, respectively. Using the time span from $t = 0$ to $t = 1000$ minutes with initial position vector $\boldsymbol{r}(0) = [x(0), y(0), z(0)] = [-2436.45, -2436.45, 6891.037]\,\mathrm{km}$ and initial velocity vector $\dot{\boldsymbol{r}}(0) = [\dot{x}(0), \dot{y}(0), \dot{z}(0)] = [5.088611, -5.088611, 0.0]\,\mathrm{km}/\,\mathrm{sec}$, at $t = 1000$ minutes you should have found the new position and velocity vectors $\boldsymbol{r}(1000) = [-5145.24, 1174.72, 5614.94]\,\mathrm{km}$ and $\dot{\boldsymbol{r}}(1000) = [2.827993, -5.464932, 3.729041]\,\mathrm{km}/\,\mathrm{sec}$ (please check and – if necessary – correct your code first).

1. To reflect uncertainty and inevitable tracking inaccuracies of the initial position of the object (its velocity be known for certain), let $\hat{\boldsymbol{r}} = \boldsymbol{r}(0)$ be the mean (in Cartesian coordinates) and

   $$\Sigma = \mathbb{E}(\boldsymbol{r} - \hat{\boldsymbol{r}})(\boldsymbol{r} - \hat{\boldsymbol{r}}) = \begin{pmatrix} 0.613 & -0.271 & -0.018 \\ -0.271 & 0.613 & -0.144 \\ -0.018 & -0.144 & 0.312 \end{pmatrix}$$

   be the covariance matrix of a trivariate Gaussian (normal) distribution. Draw random samples from this distribution (hint: search the web for "*sampling from multivariate Gaussian distribution*" for help if needed) and use your code from Assignment 1 to compute and plot all perturbed trajectories, positions, and velocities from $t = 0$ to $t = 1000$ minutes. Use a single plot for all sample trajectories and (if necessary) zoom in around $t = 1000$. Describe the effects of the initial state's uncertainty propagation using your Monte Carlo Simulation.

2. Consider a second object with (nominal) initial position $\bar{\boldsymbol{r}} = [-5351.66, 1596.76, 5310.02]\,\mathrm{km}$ and initial velocity $\dot{\boldsymbol{r}}(0) = [\dot{x}(0), \dot{y}(0), \dot{z}(0)] = [-2.471876, 5.370908, -4.099681]\,\mathrm{km}/\,\mathrm{sec}$ (the state error covariance matrix be the same as before). If a total (Euclidean) distance of 100 meters or less between the two objects is considered a collision, verify that the two objects collide after $t = 50$ minutes on their *nominal* trajectories and estimate the *actual* collision probability after $t = 50$ minutes using a Monte Carlo Approximation (hint: disable any plotting to gain speed and crank up your sample size – expect probabilities to be very small!).

3. As discussed in class, in the presence of uncertainty the time of reaching the closest point of approach may not anymore be at $t = 50$ minutes. Improve your estimate of the collision probability taking into account the actual time of closest approach between the objects (you may try to use the Matlab functions "`fminsearch`" or "`fminunc`" for unconstrained minimization).

Type your solutions using LaTeX and submit a zip file with all tex and graphics files needed for its compilation to the Drop Box on Blackboard. Please also bring a hard copy of the compiled pdf to class on Tuesday, February 8. Please ask if you have any questions!

Appendix C

MATLAB Implementation

We describe our two main functions written in MATLAB to model satellite collisions. The first function is the collision module, which compares two data history files and finds all points of closest approach. It also calculates the miss distances and the collision probabilities at these points. The second function is the case generator, used for the experiment in Chapter 5. The inputs to this function are three data history files, a range of delta-Vs, a range of times for the delta-v, and the number of points into which the ranges will be divided. Section C.1 explains these inputs a little bit better and describes how to execute the code. Figure C.1 shows a conceptual diagram of the above functions together with all their subfunctions and dependencies, that are also listed in Section C.2. The full code and some supplemental routines are given in Section C.2.1, and Sections C.2.3 and C.2.4, respectively.

C.1 Running The Code

To execute the program, the first program to run is `collision_model.m`. On the command line, input

```
output = collision_model('datafile1','datafile2')
```

where `'data file 1'` and `'data file 2'` correspond to the names of the two data files. The output will be an array containing the times of closest approach in date number format for the two objects, along with the corresponding collision probabilities and miss distances.

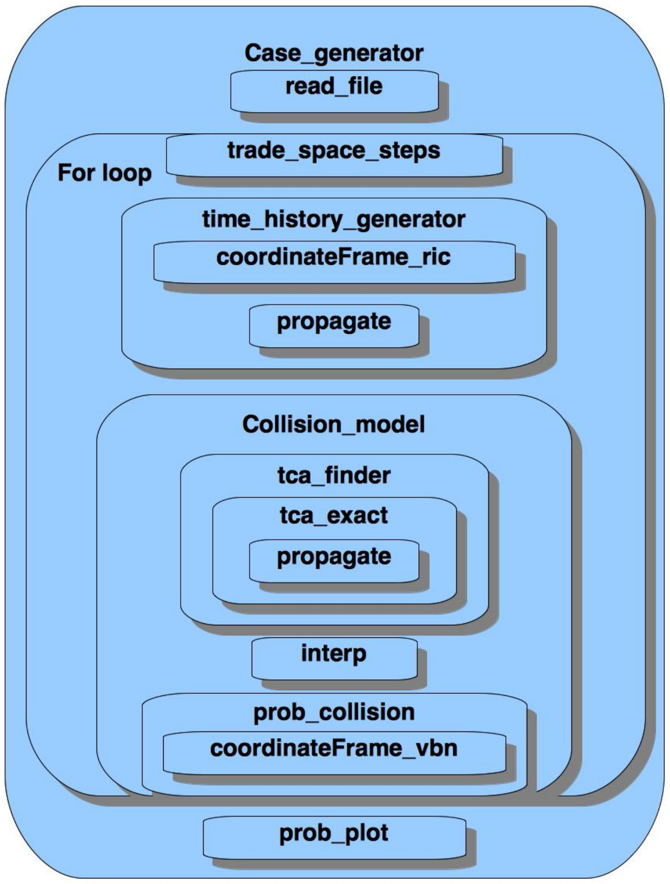


Figure C.1: Overview of functions in final MATLAB implementation

The second main function is `case_generator.m`. On the command line, input

$$\begin{aligned}\texttt{Cases} = \texttt{case\_generator}(&'\texttt{datafile1}','\texttt{datafile2}','\texttt{datafile3}',\ldots\\ &[\texttt{rangeofdelta\_Vs}],[\texttt{rangeoftimefordelta\_v}],\ldots\\ &\texttt{numberofdivisionsforthetworanges})\end{aligned}$$

where the output will be returned in n^2 cell arrays with n being the specified number of divisions. The extension from three to an arbitrary number of data files is not yet implemented but will require an only minor modification of the current code.

C.2 List of Subfunctions

Figure C.1 gives a general overview of the different functions implemented as well as their dependencies: all the functions used by some other function are contained in the box of the calling function. The following is an alphabetical listing the various functions together with a brief description of each:

Case_generator.m

generates a series of cases in which a delta-v is applied

- **Input:** 3 data history files, a range of delta-Vs, a range of times for the delta-V, and the number of points into which the ranges will be divided
- **Output:** A $n^2 \times 6$ cell array containing: Case number, [delta-v, time], aggregate probability, TCAs, miss distances, collision probabilities

Collision_model.m

calculates TCA, miss distances, and collision probabilities (used in case generator)

- **Input:** 2 sets of data files
- **Output:** an array containing TCAs, miss distances, and collision probabilities

coordinateFrame_ric.m

calculates a matrix to transform coordinates from ECI to RIC frame (used in collision model)

- **Input:** position and velocity of an object
- **Output:** a transformation matrix

coordinateFrame_vbn.m

calculates a matrix to transform coordinates from ECI to VBN frame (used in prob collision)

- **Input:** position and velocity of an object
- **Output:** a transformation matrix

Covariance_LEO.m

propagates covariance matrix to a given time (used in collision model)

- **Input:** a time range
- **Output:** a covariance matrix

interp.m

takes 6 data points and interpolates to produce new data points within the range, then finds the points at a specific time (used in collision model)

- **Input:** an array of time, position, and velocity for an object, along with a tea (time at which new points are needed)
- **Output:** a vector with states at TCA

prob_collision.m

calculates the probability of collision between two objects (used in collision model)

- **Input:** states of two objects at TCA
- **Output:** probability of collision

propagate.m

propagates the state of an object using the two-body differential equation and an ode solver (used in tca exact)

- **Input:** initial position and velocity of an object, and a time span
- **Output:** a new array of states

read_file.m

reads data from a file and places into an array (used in case generator)

- **Input:** a data file
- **Output:** a data array

tca_exact.m

calculates the exact TCA near the minimum relative position points (used in tca finder)

- **Input:** states of two objects at minimum relative position
- **Output:** exact time of close approach

tca_finder.m

calculates all the times of close approach when the relative position vector is below a certain threshold (used in collision model)

- **Input:** two data arrays containing times and corresponding states (position and velocity)
- **Output:** an array of TCAs (in date number format)

time_history_generator.m

generates a new set of data that includes a delta-v (used in case generator)

- **Input:** a matrix containing time history data for one object, a delta-v, and a time of delta-v (which refers the row of the data array)
- **Output:** a new matrix containing the altered time history data

trade_space_steps.m

used to create vectors of evenly spaced points from two ranges of values (used in case generator)

- **Input:** a range of delta-Vs and a range of delta-Ts
- **Output:** two vectors of evenly spaced points within the range

C.2.1 Main Code: Satellite Model

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                           Collision_model.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Cases=case_generator(thf1, thf2, thf3, range_dv, range_t,num_dvs)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function completes the following:
%   !. Generates new time history files that contain a delta_v
%   2. Computes the new set of close approach data.
%   3. Store each output in a cell array
%   4. Graphs the aggregate probabilities over the trade-space.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

data1=read_file(thf1,1);
data2=read_file(thf2,1);
data3=read_file(thf3,1);


% We divided in the trade space into n sepments.
[trade_space_dv, stepsize_dv, trade_space_t, stepsize_t] = ...
    trade_space_steps(range_dv, range_t, num_dvs);


% Counter for the total case
counter=1;
% Counter for the different delta_vs
counter_v=1;
% Counter for the time of the delta_vs
counter_t=1;

for i=trade_space_dv(1):stepsize_dv:trade_space_dv(end)
    for j = trade_space_t(1):stepsize_t:trade_space_t(end)

        % We need the delta_v to be at a specific time node.  We assume the
        % data is in 5 minute intervals, and there are 288 such intervals
        % per day.
        delta_t=floor(j*288);

        % Only generate new data if the delva_v does not equal 0
        if i ~= 0
            data1_v=time_history_generator(data1,i, delta_t);
        else
            data1_v = data1;
        end

        % Output here will be TCAs and corresponding probability and miss
        % distance
        output_1=collision_model(data1_v, data2);
        output_2=collision_model(data1_v, data3);

        % The next few lines concatentate the data in single cell arrays.
        % At this point, the data is arranged by object--that is, all the
        % data from one object is together.
        probabilities=[output_1{2}; output_2{2}];
        tcas=[output_1{1}; output_2{1}];
        miss_distances=[output_1{3};output_2{3}];

        aggregate_prob=1-prod(1-probabilities);

        % Now we combine the various cells from above into one array.
        Cases(counter,:)= {counter,[i,j],aggregate_prob, tcas,
            miss_distances, probabilities};
        counter=counter+1;

        % We use this to create the probability plot
        prob_matrix(counter_t,counter_v)=aggregate_prob;
        counter_t=counter_t+1;


    end
    counter_v=counter_v+1;
    counter_t=1;
end
% We plot the aggregate probabilites over the trade space.
prob_plot(trade_space_dv,trade_space_t,prob_matrix);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                    coordinateFrame_ric.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [M_ric]=coordinateFrame_ric(pos_1, vel_1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This produces a matrix that transforms coordinates from ECI to RIC for a
% specific object
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

R=pos_1/norm(pos_1); V=vel_1/norm(vel_1);
h=cross(R,V);      C=h/norm(h);      I=cross(C,R);
%
M_ric=zeros(3,3); M_ric(1,:)=R; M_ric(2,:)=I; M_ric(3,:)=C;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                    coordinateFrame_vbn.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [M_vbn] = coordinateFrame_vbn(pos_1, vel_1, pos_2, vel_2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This produces a matrix that tranforms coordinates to an orthogonal
% relatvie velocity frame
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

vel_r=vel_2-vel_1; pos_r=pos_2-pos_1;
R=pos_r/norm(pos_r); V=vel_r/norm(vel_r);
h=cross(R,V);      C=h/norm(h);      I=cross(C,V);
%
M_vbn=zeros(3,3); M_vbn(:,1)=C; M_vbn(:,2)=V; M_vbn(:,3)=I;
end
```

```
% File to create Covariance Time History File
% Single Time History to be used for all secondaries.

function [cov_new] = Covariance_LEO(t0,t_end)
 tVec = [t0 t_end]; % t represented in days

 % Input Coefficients for Error Profiles (assumes quadratic error
 % functions) - units are meters

 %----------- Position Error, Radial InTrack CrossTrack ---------

    %Radial Error Coefficients
        PosErrorCoeff_R = [-.3,2.5,2];
    %In-Track Error Coefficients
        PosErrorCoeff_IT = [6.2,30,6];
    % CrossTrack Error Coefficients
        PosErrorCoeff_CT = [-.3,4.5,2];
 %---------------------------------------------------------------

 % Input Correlation Terms:

 Corr21 = -.8;
 Corr31 = 0.5;
 Corr32 = -.45;


    % Evaluate Polynominals
        PosError_Radial = polyval(PosErrorCoeff_R,tVec);
        PosError_InTrack = polyval(PosErrorCoeff_IT,tVec);
        PosError_CrossTrack = polyval(PosErrorCoeff_CT,tVec);

        PosError_Total = sqrt(PosError_Radial.^2+ PosError_InTrack.^2+
            PosError_CrossTrack.^2);

        P21 = Corr21*PosError_Radial.*PosError_InTrack;
        P31 = Corr31*PosError_CrossTrack.*PosError_Radial;
        P32 = Corr32*PosError_CrossTrack.*PosError_InTrack;


        CovMatrix = zeros(length(tVec),7);

        CovMatrix(:,1) = tVec;
        CovMatrix(:,2) = PosError_Radial.^2;
        CovMatrix(:,3) = P21;
        CovMatrix(:,4) = PosError_InTrack.^2;
        CovMatrix(:,5) = P31;
        CovMatrix(:,6) = P32;
        CovMatrix(:,7) = PosError_CrossTrack.^2;

cov_new=zeros(3);
cov_new(1,1)=CovMatrix(end,2);
cov_new(1,2)=CovMatrix(end,3);
cov_new(1,3)=CovMatrix(end,5);
cov_new(2,1)=CovMatrix(end,3);
cov_new(2,2)=CovMatrix(end,4);
cov_new(2,3)=CovMatrix(end,6);
cov_new(3,1)=CovMatrix(end,5);
cov_new(3,2)=CovMatrix(end,6);
cov_new(3,3)=CovMatrix(end,7);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                                interp.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function pos_vel=interp(data,tca)
% Using Lagrange interpolation finds position and velocity out of 6 points
% of data1, that are around tca
data_size=size(data);
if (data_size(2)==7)
time=data(:,1);
size_time=size(time);
x=data(:,2);
y=data(:,3);
z=data(:,4);
x1=data(:,5);
y1=data(:,6);
z1=data(:,7);
pos_vel(1)=tca;
% 1) find the closest point to the tca in the data1
time1=time-tca;
min_abs=min(abs(time1));
index=find(time1==min_abs);
size_index=size(index);
if (size_index(1)==0) % so we look for negative
   index=find(time1==-min_abs);
   size_index=size(index);
end
if (size_index(1)==0)
    display('Error in interp.m, can not find the closest point to the tca');
end
pos_vel(1)=tca;

if (index(1)>=size_time(1)-3) % take 5 indexes from the left
    pos_vel(2)=interp1(time(index(1)-5:index(1)),x(index(1)-5:index(1)),tca,
        'spline');
    pos_vel(3)=interp1(time(index(1)-5:index(1)),y(index(1)-5:index(1)),tca,
        'spline');
    pos_vel(4)=interp1(time(index(1)-5:index(1)),z(index(1)-5:index(1)),tca,
        'spline');
    pos_vel(5)=interp1(time(index(1)-5:index(1)),x1(index(1)-5:index(1)),tca
        ,'spline');
    pos_vel(6)=interp1(time(index(1)-5:index(1)),y1(index(1)-5:index(1)),tca
        ,'spline');
    pos_vel(7)=interp1(time(index(1)-5:index(1)),z1(index(1)-5:index(1)),tca
        ,'spline');
elseif (index(1)<3) % take 5 indexes from the right

    pos_vel(2)=interp1(time(index(1):index(1)+5),x(index(1):index(1)+5),tca,
        'spline');
    pos_vel(3)=interp1(time(index(1):index(1)+5),y(index(1):index(1)+5),tca,
        'spline');
    pos_vel(4)=interp1(time(index(1):index(1)+5),z(index(1):index(1)+5),tca,
        'spline');
    pos_vel(5)=interp1(time(index(1):index(1)+5),x1(index(1):index(1)+5),tca
        ,'spline');
    pos_vel(6)=interp1(time(index(1):index(1)+5),y1(index(1):index(1)+5),tca
        ,'spline');
    pos_vel(7)=interp1(time(index(1):index(1)+5),z1(index(1):index(1)+5),tca
        ,'spline');

else % take 2 from the left and 3 from the right

    pos_vel(2)=interp1(time(index(1)-2:index(1)+3),x(index(1)-2:index(1)+3),
        tca,'spline');
    pos_vel(3)=interp1(time(index(1)-2:index(1)+3),y(index(1)-2:index(1)+3),
        tca,'spline');
    pos_vel(4)=interp1(time(index(1)-2:index(1)+3),z(index(1)-2:index(1)+3),
        tca,'spline');
    pos_vel(5)=interp1(time(index(1)-2:index(1)+3),x1(index(1)-2:index(1)+3)
        ,tca,'spline');
    pos_vel(6)=interp1(time(index(1)-2:index(1)+3),y1(index(1)-2:index(1)+3)
        ,tca,'spline');
    pos_vel(7)=interp1(time(index(1)-2:index(1)+3),z1(index(1)-2:index(1)+3)
        ,tca,'spline');

end
else
display('data should contain 7 columns')
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                              prob_plot.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function prob_plot(trade_space_dv,trade_space_t,prob_matrix)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Creates a contour plot over a m by n matrix
% Specifically used to plot probabilities over trade space
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% x-axis is the range of delta_vs
X=trade_space_dv.*1e+5;
% y-axix is the range of times
Y=trade_space_t.*1440;
% contours relate to the probabilities
Z=prob_matrix;
[C,h]=contourf(X,Y,Z);
xlabel('Delta vs (in cm/sec)','Fontsize',24);
ylabel('Time of delta v (minutes from start of file)','Fontsize',24);
clabel(C,h);
colormap(cool);
colorbar('EastOutside');
title('Probability Contours','Fontsize',32);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                         Prob_collision.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [p_int] = prob_collision(pos_1, vel_1,cov_1, pos_2, vel_2, cov_2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This computes the probability of collision using a two dimensional
    integral.
% We first numerically evaluate the double integral.
% We then estimate the integral by a simple sum per Gottlieb, Sponaugle,
    Gaylord
% code by mark mueller and jeremy doan
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% inital data, which will be inputs:
% 2 covariances, 2 positions, 2 velocities, hard body radius


% hard body radius
dist = .100;


% Combined covariance matrix in an orthogonal relative velocity frame
[M_vbn]=coordinateFrame_vbn(pos_1,vel_1,pos_2,vel_2);
pos_r2=M_vbn'*(pos_2-pos_1)'; mu=pos_r2(1:2:3);
G=M_vbn*(cov_1+cov_2)*M_vbn';
g=G(1:2:3,1:2:3); ginv=inv(g);



% This will give us the probability
[p_int] = prob_collision_int(g, mu, ginv, dist);
end

 %                              Subfunctions

function [p_int] = prob_collision_int(g, mu, ginv, dist)
% This numerically evaluates the double integral
% limits of integration
 n=dist; m=-n;
 f2 = 1/(2*pi*sqrt(det(g))); tol = 1.0e-9;
 a = ginv(1,1); b = ginv(1,2); c = ginv(2,1); d = ginv(2,2);
 int1 = @(x,y)exp(-.5*(a*(x-mu(1)).^2 + (c+b)*(x-mu(1)).*(y-mu(2))+ d*(y-mu
     (2))^2)).*(x.^2+y.^2<n^2);
 out2=dblquad(int1, m, n, m, n, tol);
 p_int = f2*out2;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
```

```
%                          propogate.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [X] = propagate(pos_0, vel_0, tspan)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   This function uses ode45 (an ordinary differential equation solver) to
%   propagate the states of two satellites using the famed two-body
%   differential equation.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% These are options that we would use for the ode solver.
options = odeset('RelTol',1e-9,'AbsTol',1e-9);


% The initial state must be one vector containing position and velocity.
x0=[pos_0, vel_0];

% The output X will be the propogated states of the objects.
% The output tp1 is the time in seconds.
[tp1,X] = ode45(@orbit,tspan,x0,options);
end

function f = orbit(t,y)
%%This function turns our second order two body problem equation
%%into a system of first order differential equations.
%%Inputs: t = time
%%Output: used in ode45.
mu = 398600.5;
f = zeros(6,1);
f(1) = y(4);
f(2) = y(5);
f(3) = y(6);
f(4) = -(mu*y(1))/((y(1))^2+(y(2))^2+(y(3))^2)^(3/2);
f(5) = -(mu*y(2))/((y(1))^2+(y(2))^2+(y(3))^2)^(3/2);
f(6) = -(mu*y(3))/((y(1))^2+(y(2))^2+(y(3))^2)^(3/2);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                          read_file.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [S]= read_file(thf,num)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%       This function reads in date from a time history file.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This will come from a time history file
fid = fopen(thf,'r');
thd =textscan(fid, '%24c %f %f %f %f %f %f');
fclose(fid);
```

```
% The first 24 places are the date string.  This is what is found in thd{1}
comb=thd{1}; %time (dd mmm yyyy HH:MM:SS.FFF)

% We can use this function whether the date in the time history file is in
% Julian format (a date number) or in Gregorian (a date string).
if num ==1
   T= datenum(comb);
end

if num ==2
   T= datenum(comb);
   T = datestr(T,'dd-mmm-yyyy HH:MM:SS.FFF');
end

% We place the vaules into an array.
% T is the time.
% x, y, z are the position components.
% vx. vy, vz are the velocity components.
x = thd{2};          vx = thd{5};
y = thd{3};          vy = thd{6};
z = thd{4};          vz = thd{7};
S = [T, x, y, z, vx, vy, vz];
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                         tca_exact.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [tca] = tca_exact(pos_primary,vel_primary,pos_secondary,
    vel_secondary)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function calculates the time of closest approach (TCA).
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% To begin, we set t_diff, which is the time difference between the
% positions at the initial time and the positions at the actual tca.  We
% iterate until t_diff is less than some small epsilon.
% Initially, we set t_diff equal to 1 (so that the loop will run).
% We also set tca = 0.;

t_diff = 1;
tca=0;

% This loop will calculate t_diff, and then make the new tca the former tca
% plus t_diff.  This loop runs until t_diff is less than the specificed
% epsilon.
while abs(t_diff) > 1e-10

    % The first time the loop runs, we will use the states supplied as
    % imputs.  For each time afterward, we use the states at the new tca.

    % We calculate the relative states, and then use these to calculate
    % t_diff, the difference in time between the nomical tca and the actual
    % tca.
    pr = pos_primary-pos_secondary;
    vr = vel_primary-vel_secondary;


    % This equation is derived from the formula p =p_o + vt, where:
    %   p is the position at the tca,
    %   p_0 is the intitial relative position,
    %   vt represents the difference between the two.
    % We calculate (p*p), take the derivative, and then solve for t.
    % The t that we obtain is t_diff.  The relative position is at a
    % minimum when t_diff =0;
    t_diff = -dot(pr,vr)/norm(vr)^2;

    % Our new tca is the old one plus the calcuated difference in time.  As
    % we iterate this t_diff becomes smaller and eventually we will have
    % our new tca as well as the states at that time.
    tca = tca + t_diff;

    % We need to propagate to the new tca.  tspan is the time span over
    % which we propagate.
    tspan = [0 t_diff];


    % We use the ode solver to propogate the states to the necessary times
    [X] = propagate(pos_primary, vel_primary, tspan);
    [Y] = propagate(pos_secondary, vel_secondary, tspan);

    % We identify the states at this new tca.
    pos_primary = X(end,1:3);
    vel_primary = X(end,4:6);

    pos_secondary = Y(end,1:3);
    vel_secondary = Y(end,4:6);


end


end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                         tca_finder.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [tca_s]= tca_finder(thf1, thf2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function finds all the times of close approach when the relative
% position is below a specified threshold, the dist_sep value.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dist_sep = 2; % in kilometers
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% INPUTS: thf1 = time history file of object 1
%         thf2 = time history file of object 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% If we need to read files directly by this program, we can.
% Otherwise, we will just use the data arrays.
%thf1 = read_file(thf1,1); % read in time history data 1
%thf2 = read_file(thf2,1); % read in time history data 2


% We assume that the time data files have the same length.  So we can use
% the same value for both data sets.
stop = length(thf1(:,1));

% This counter is used to organize the output
counter=1;

% Our loop first calculates the relative position vector.  If the magnitude
% of the vector is below dist_sep (which can be set arbitrarily or for a
% specific value), then the function finds the local minimum data point.
% At that point it calculates the each tca (time of close approach) then
% then uses interpolation to discover the states at that time.
for i = 1:stop
    pos_1(i,1:3)=thf1(i,2:4);
    pos_2(i,1:3)=thf2(i,2:4);
    vel_1(i,1:3)=thf1(i,5:7);
    vel_2(i,1:3)=thf2(i,5:7);

    % We calculate the relative position vector.  We place it into an array
    % so that it is indexed and we can refer to it later
    pos_rel(i,1:3) = pos_1(i,1:3)-pos_2(i,1:3);
end
tca_s=[0];
% This loop calculates the local minimums--the points from the time history
% files that are lower than the ones around it.  We assume that the graph
% of the magnitude of the relative position vectors will be sinusoidal-ish,
% and thus there will exist several local minimums.
for j=2:stop-1

    % This filters out relative positions whose magnitude is above
    % dist_sep.
    if norm(pos_rel(j,1:3)) < dist_sep

        % This conditional looks for the point at which the slope is
        % changing from negative to positive, which will be a local
        % minimum.
        if norm(pos_rel(j-1,1:3)) > norm(pos_rel(j,1:3)) && ...
               norm(pos_rel(j+1,1:3)) > norm(pos_rel(j,1:3))

            %disp(datestr(thf1(j,1),'mm-ddd-yyyy HH:MM:SS.FFF'))

            % Using the local minimum from above, we take the states at
            % that point and calculate the tca
            tca=tca_exact(pos_1(j,1:3),vel_1(j,1:3),pos_2(j,1:3),vel_2(j
                ,1:3));
            % To get the exact time in dd-mmm-yyyy HH:MM:SS.FFF format, we
            % take the time from the data file (which we be a date number
            % representing days).
            % The value from the tca_exact function will be in seconds.
            % We divide tca by 86400 (seconds in a day) and then add it to
            % our time from the data file.
            % When we convert this date number back to a date string, we
            % will have the exacat day and millisecond of the day for the
            % tca.
            time = thf1(j,1);
            tca=time + tca/86400;

            % We place the date string into an array.
            tca_s(counter)=tca;



            counter=counter+1;
        end
    end
end

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                          time_history_generator.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function new_data=time_history_generator(data,delta_v,delta_t,theta)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Creates a set of data that includes a delta_v at a specific time.
% Currently, it saves the new data in a matrix.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Creates a default 'theta', which is measured from the radial direction
% (i.e. straight out from the center of the earth
if ~exist('theta','var')
    theta = pi/2;
end

% Pull states from data at time of delta_v
position = data(delta_t,2:4);
velocity = data(delta_t, 5:7);

% To apply the delta_v, we need a transformation matrix.
% We apply the delta_v in the RIC frame, and then tranform back to
% Cartesian
[M_ric]=coordinateFrame_ric(position, velocity);
velocity=velocity + delta_v.*[cos(theta),sin(theta),0]*M_ric';

% Set the length of the for loop.
stop = length(data(delta_t:end,1));
time = data(delta_t,1);
```

```
% Each loop propagates out 300 seconds (5 min) to create a set of data at 5
% minute intervals.
for i=1: stop-1

    tspan=[0 300];

    X = propagate(position, velocity, tspan);


    position = X(end,1:3);
    velocity = X(end,4:6);

    time = data(delta_t+i,1);

    new_data(i,:)=[time, position, velocity];
end

new_data=vertcat(data(1:delta_t,:),new_data);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                   trad_space_steps.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [trade_space_dv, stepsize_dv, trade_space_t, stepsize_t] =
    trade_space_steps(range_dv, range_t, n)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function takes the trade space made up of max and min delta_v and
% max and min delta_t, and then divides the range into equally distanced
% discrete points.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% n is the number of points we use to divide our possible delta v's and
% the time of the delta v's

% First we get the range of possible delta_vs

min_dv = range_dv(1,1);
max_dv = range_dv(1,2);

% We need to divide the range of delta_vs into n discrete segments
trade_space_dv= linspace(min_dv, max_dv, n);
stepsize_dv = trade_space_dv(2)-trade_space_dv(1);

% Next we get the range of times in terms of days.
max_t = range_t(1,2);
min_t = range_t(1,1);

% We then divide this time span in the discrete points.
trade_space_t = linspace(min_t, max_t, n);
stepsize_t= trade_space_t(2)- trade_space_t(1);
end
```

C.2.2 Supplemental Code: Optimization

Opt3_F.m

```
function [Del_V,theta] = Opt3_F(N, r_1, v_1, r_2, v_2, cov_1, cov_2, t ,P_T,
     P_C)

if ~((N == 1)||(N == 2)); error('Opt3: N must be 1 or 2'); end

[cov,M]=Comb_cov(r_1, v_1, cov_1, r_2, v_2, cov_2);   h_rel = M(:,3);
sigma_x = sqrt(cov(1,1)); sigma_z = sqrt(cov(3,3)); rho_xz = cov(1,2)/(
     sigma_x*sigma_z);

S = [r_1, v_1; r_2, v_2];   Y = Kepler3(S);
[xi_b, eta_b, u_b, C_ubc, S_ubc, r_bc, n_b, C_ub, S_ub, P_b ]= getorbitdata(
     Y,r_1,t);

M = ECI_to_RTN(r_1, v_1);   r = M(1,:); r_T = M(2,:);  % r_N = M(3,:); % to
     be checked

[a,b,c,d,e]=getorbitfn(xi_b, eta_b, C_ubc, S_ubc, r_bc, n_b, C_ub, S_ub, P_b
      ,t, Y(:,3));
r_rel = r_2-r_1;

if N == 1;   [theta, Del_r] = gettheta1(a, b, c, d, r_rel, r_T);
else         [theta, Del_r] = gettheta2(a, b, c, d, r, r_T, r_rel, h_rel,
     sigma_x, sigma_z, rho_xz);
end

[A, B, C] = getABC(r_rel, h_rel, Del_r, sigma_x, sigma_z, rho_xz, P_T, P_C);

 f = @(x) A*x^2 + B*x + C; x0 = 0;  x1 = 100;  tol = eps;  n=1000; % Max #
     iterations
 Del_V = secant2(x0, x1, n, f, tol);
end
```

Test_Opt3_F.m

```
% Test program for the optimization program
% test noncircular orbits
 clear; clc; close all

 P_T = 1e-10; % probability of collision threshold

 N=2; % 1 for miss vector; 2 for gradient

cov_1 = [0.613 -0.271 -0.018; -0.271 0.613 -0.144; -0.018 -0.144 0.312];
cov_2 = [3.099E+03 -2.437E+04  1.010E+02; -2.437E+04  2.051E+05 -3.100E+02;
     1.010E+02 -3.100E+02  2.248E+02];

%%%%%%%%%%%%%%%%%%%%%% construct test case %%%%%%%%%%%%%%%%%%%%%%%%

%We use somewhat arbitrary positions and velocities, such that they meet
%the requirement of violating our probability of collision threshold at the
     tca.
 r1=[-2400 -2400 6900];
 r2=[-2400 -2400.01 6899.99];
 v1=[5.0 -5.0 0.0];
 v2=[5.0 -5.0 3.6];
 test_tca = 5000;    % seconds to be run 'backwards' to get our initial
      conditions
 [r_1, r_2, v_1, v_2]=gettestcase(r1,r2,v1,v2,test_tca); % gives us our
      initial state
%axis('square'); plot3(Y1(:,1),Y1(:,2),Y1(:,3),'-','LineWidth',1); plot3(Y2
     (:,1),Y2(:,2),Y2(:,3),'-','LineWidth',1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M=ECI_to_RTN(r_1,v_1);
%We convert our primary object's coordinates from the ECI frame to the RTN
     frame,
%to perform our maneuver.
y1 = cat(2,r_1,v_1)';  y2 = cat(2,r_2,v_2)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
kmax = 500 ; %this is the number of unique Delta-V's we want to test
t_del = 10 ; %this is the amount of time between the test Delta-V's
Del_V = zeros(kmax,1);    theta = zeros(kmax,1);    P_C = zeros(kmax,1);

tca=t_nominal(r_1, r_2, v_1, v_2); %t_nominal finds a ruff estimate of our
     tca,
%used below, to give close_approach2 a place to start looking for the
%precise tca (accurate to the millisecond).
tca0=tca;
[pos_p, vel_p, pos_s, vel_s, tca] = close_approach2(r_1, v_1, r_2, v_2, tca)
     ;
disp(tca);
P_C0 = prob_collision6(pos_p, vel_p, pos_s, vel_s, cov_1, cov_2);
if P_C0 <= P_T; fprintf('probability of collision is below threshold: %10.3e
     versus %10.3e \n',P_C0,P_T); break;
else  fprintf('probability of collision is %10.3e and the probability
     threshold is %10.3e \n',P_C0,P_T); end

[V,th]=Opt3_F(N, pos_p, vel_p, pos_s, vel_s, cov_1, cov_2, tca, P_T, P_C0);
     Del_V(1,1)=V;  theta(1,1)=th;
v_1_new = v_1 + V*[cos(th), sin(th), 0]*M';

tca=t_nominal(r_1, r_2, v_1_new, v_2);
[pos_p, vel_p, pos_s, vel_s, tca_1] = close_approach2(r_1, v_1_new, r_2, v_2
     , tca);
P_C(1,1) = prob_collision6(pos_p, vel_p, pos_s, vel_s, cov_1, cov_2) ;
t_trix = zeros(kmax,1);
t_trix(1,1) = tca_1;
for k = 2:kmax;    if mod(k,10)==0; disp(k); end
% update satellites position, velocity
t=tca_1-((k-1)*t_del);    tspan = [0 t];
[T1,Y1]=ode45(@orbit,tspan,y1,odeset('RelTol',1e-9));
[T2,Y2]=ode45(@orbit,tspan,y2,odeset('RelTol',1e-9));
r_1_t=Y1(end,1:3);  r_2_t=Y2(end,1:3);  v_1_t=Y1(end,4:6);   v_2_t=Y2(end
     ,4:6);
t_trix(k,1) = t;
% update satellites tca
tca=t_nominal(r_1_t, r_2_t, v_1_t, v_2_t);
[pos_p, vel_p, pos_s, vel_s, tca] = close_approach2(r_1_t, v_1_t, r_2_t,
     v_2_t, tca);
```

```
% current maneuver
[V,th]=Opt3_F(N, pos_p, vel_p, pos_s, vel_s, cov_1, cov_2, tca-t, P_T, P_C0)
    ;  Del_V(k,1)=V;   theta(k,1)=th;
M_t=ECI_to_RTN(r_1_t, v_1_t);
v_1_new = v_1_t + V*[cos(th), sin(th), 0]*M_t';

% update prob collision
tca=t_nominal(r_1_t, r_2_t, v_1_new, v_2_t);
[pos_p, vel_p, pos_s, vel_s, tca] = close_approach2(r_1, v_1_new, r_2, v_2,
    tca);
P_C(k,1) = prob_collision6(pos_p, vel_p, pos_s, vel_s, cov_1, cov_2) ;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

hold on
 figure(1);plot(tca0-(1:kmax)*t_del,Del_V,'r-');title('Delta_V');
 axis([0 5100 0 .5]); xlabel('time_in_seconds'); ylabel('change_in_velocity_
     in_km/s');
 figure(2);plot(tca0-(1:kmax)*t_del,theta,'b-.');title('theta');
 xlabel('time_in_seconds'); ylabel('radians');
 figure(3);plot(tca0-(1:kmax)*t_del,P_C,'g.');title('Probability_of_
     Collision');
axis([0 5100 0 1.0e-255]);

moves = horzcat(t_trix,Del_V,theta,P_C);
disp(moves);

%subplot(3,1,1), plot(tca-(1:kmax)*t_del,Del_V,'r')
%subplot(3,1,2), plot(tca-(1:kmax)*t_del,theta,'b')
%subplot(3,1,3), plot(tca-(1:kmax)*t_del,P_C,'g')

%fprintf('time          Del_V        theta \n');
%fprintf('%6d %10.4f %10.4f \n',k,Del_V(k,1),theta(k,1));
```

close_approach2.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                       close_approach2.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This function calculates the time of closest approach (TCA).  It outputs
% the position vector at this time.

function [pos_prim_ca,vel_prim_ca,pos_sec_ca,vel_sec_ca,tca] = ...
    close_approach2(pos_primary,vel_primary,pos_secondary, vel_secondary,
        tca)

% t_diff is the difference between the nominal tca and the actual tca.  At
% the start we set out t_diff to be equal to the nomial tca. t_diff becomes
% the time span over which we search.  In the end, we want t_diff to be
% zero.
t_diff = tca;
```

```
% These options are for the ode solver.
options = odeset('RelTol',1e-12,'AbsTol',1e-12);

% This loop will calculate t_diff, and then make the new tca the former tca
% plus t_diff.  This loop runs until t_diff is less than the specificed
% epsilon.
while abs(t_diff) > 1e-09

    % We set our inittial state for the ode solver.  The first time the
    % loop runs, we will use the states supplied as imputs.  For each time
    % afterward, we use the states at the new tca.
    x0=[pos_primary,vel_primary];
    y0=[pos_secondary, vel_secondary];

    % We want to make sure that the time step for the ode solver is small
    % enough to calculate the exact tca.  We use a time step of t_diff*.01
    % so that as t_diff gets, smaller, we have a descreasing stepsize.
    % Also, since t_diff is not always positive, we will have a vector with
    % the properly signed values.
    tspan = [0:t_diff*.01:t_diff];

    % We use the ode solver to propogate the states to the necessary times
    [tp1,x] = ode45(@orbit,tspan,x0,options);
    [tp2,y] = ode45(@orbit,tspan,y0,options);

    % The end time, which will be the initial time plus t_diff, is the new
    % states for the two objects.  Notice that at first t_diff is equal to
    % the nominal tca.
    pos_primary = x(end,1:3);
    vel_primary = x(end,4:6);

    pos_secondary = y(end,1:3);
    vel_secondary = y(end,4:6);

    % We calculate the relative states, and then use these to calculate
    % t_diff, the difference in time between the nomical tca and the actual
    % tca.
    pr = pos_primary-pos_secondary;
    vr = vel_primary-vel_secondary;

    t_diff = -dot(pr,vr)/norm(vr)^2;

    % Our new tca is the old one plus the calcuated difference in time.  As
    % we iterate this t_diff becomes smaller and eventually we will have
    % our new tca as well as the states at that time.
    tca = tca + t_diff;

end

pos_prim_ca = pos_primary;
vel_prim_ca = vel_primary;
pos_sec_ca = pos_secondary;
vel_sec_ca = vel_secondary;

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function f = orbit(t,y)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%This function turns our second order two body problem equation
%%%into a system of first order differential equations.
%%%Inputs: t = time
%%%         y = some position
%%%Output: used in ode45.

mu = 398600.5;
f = zeros(6,1);
f(1) = y(4);
f(2) = y(5);
f(3) = y(6);
f(4) = -(mu*y(1))/((y(1))^2+(y(2))^2+(y(3))^2)^(3/2);
f(5) = -(mu*y(2))/((y(1))^2+(y(2))^2+(y(3))^2)^(3/2);
f(6) = -(mu*y(3))/((y(1))^2+(y(2))^2+(y(3))^2)^(3/2);
end
```

Comb_cov.m

```
function [combined_cov,M] = Comb_cov(pos_1,vel_1,cov_1,pos_2,vel_2,cov_2 )
% RIC frame for object 1
R_1=pos_1/norm(pos_1); V_1=vel_1/norm(vel_1);
h_1=cross(R_1,V_1);     C_1=h_1/norm(h_1);     I_1=cross(C_1,R_1);
% converts from inertial cartesian frame to RIC #1 frame
M_1=zeros(3,3); M_1(1,:)=R_1; M_1(2,:)=I_1; M_1(3,:)=C_1;
cov_1n = M_1'*cov_1*M_1;

% RIC frame for object 2
R_2=pos_2/norm(pos_2); V_2=vel_2/norm(vel_2);
h_2=cross(R_2,V_2);     C_2=h_2/norm(h_2);     I_2=cross(C_2,R_2);
% converts from inertial cartesian frame to RIC #2 frame
M_2=zeros(3,3); M_2(1,:)=R_2; M_2(2,:)=I_2; M_2(3,:)=C_2;
cov_2n = M_2'*cov_2*M_2;

% relative velocity and two arbitray orthogonal vectors frame
vel_r=vel_2-vel_1; pos_r=pos_2-pos_1;
R_r=pos_r/norm(pos_r); V_r=vel_r/norm(vel_r);
h_r=cross(R_r,V_r);     C_r=h_r/norm(h_r);     I_r=cross(C_r,V_r);
% converts from inertial cartesian frame to relative velocity frame
M=zeros(3,3); M(:,1)=C_r; M(:,2)=V_r; M(:,3)=I_r;

combined_cov=M*(cov_1n+cov_2n)*M';
```

ECI_to_RTN.m

```
function M = ECI_to_RTN(r,v)
% converts from inertial Earth-centered cartesian frame to Radial,
% Transverse, Normal
R=r/norm(r );    V=v/norm(v);    h=cross(R,V);    T=h/norm(h);    N=cross(T,R);
M=zeros(3,3); M(1,:)=R; M(2,:)=T; M(3,:)=N;
%cov_n = M'*cov*M;
```

getABC.m

```
function [A, B, C] = getABC(r_rel , h_rel , Del_r , sigma_x , sigma_z , rho_xz ,
    P_T, P_C)
% Determine the delta-V magnitude
A = (dot(Del_r , r_rel)/sigma_x)^2 + (dot(Del_r , h_rel)/sigma_z)^2 - ...
    (2*rho_xz*dot(Del_r , r_rel)*dot(Del_r , h_rel))/(sigma_x*sigma_z);

B = (2*norm(r_rel)/sigma_x)*((dot(Del_r , r_rel)/sigma_x) -...
    rho_xz*(dot(Del_r , h_rel)/sigma_z));

C = 2*(1-rho_xz^2)*log(P_T/P_C);
```

getorbitdata.m

```
function [xi_b , eta_b , u_b ,C_ubc , S_ubc , r_bc , n_b , C_ub , S_ub , P_b ]=
    getorbitdata(Y,r_1 ,t)
a= Y(:,1); e = Y(:,2); h= Y(:,3); w = Y(:,4); nu= Y(:,5);

% The following eqautions are described in the Appendix of the paper.
xi_b = e(1)* cos(w(1));      % eccentricity * cos(arg. of perigee)
eta_b = e(1)*sin(w(1));      % eccentricity * sin(arg. of perigee)
u_b = nu(1) + w(1);          % Argument of Latitude before burn
C_ubc = cos(u_b);            % cos(arg. of latitude) at time of burn
S_ubc = sin(u_b);            % sin(arg. of latitude) at time of burn
r_bc = norm(r_1);            % magnitude of position vector
mu = 398600.4418;            % G * M_Earth (km^3/s^2).
n_b = sqrt(mu/a(1)^3);       % mean motion before burn (check validity of
    equn.)

C_ub = C_ubc*cos(n_b*t) + S_ubc*sin(n_b*t);
S_ub = S_ubc*cos(n_b*t) - C_ubc*sin(n_b*t);
P_b = norm(h(1))^2/mu;
```

getorbitfn.m

```
function [a,b,c,d,e]=getorbitfn(xi_b , eta_b , C_ubc, S_ubc , r_bc , n_b, C_ub,
    S_ub, P_b ,t, h)
a = (r_bc^2/ h(1))*sin(n_b * t);
b = -3*(xi_b*S_ub - eta_b*C_ub)*t;
c = (r_bc^2/ h(1))*(2 - xi_b*C_ub - eta_b*S_ub - ...
    (2 - xi_b*C_ub - eta_b*S_ub)* cos(n_b * t)) - ...
    3*(xi_b*S_ubc - eta_b*C_ubc)*t;
d = -3*t *((1-xi_b*S_ubc - eta_b*C_ubc)/(1-xi_b*C_ub - eta_b*S_ub));
e = r_bc* (P_b/h(1))*(1-xi_b*C_ub - eta_b*S_ub)*sin(n_b*t);
```

gettestcase.m

```
function [r_1 , r_2 , v_1 , v_2]=gettestcase(r1 ,r2 ,v1 ,v2 ,test_tca)
 y1=cat(2,r1,v1)';  y2=cat(2,r2,v2)';
 tspan=[0 -test_tca];
```

```
[T1,Y1]=ode45(@orbit,tspan,y1,odeset('RelTol',1e-9));
[T2,Y2]=ode45(@orbit,tspan,y2,odeset('RelTol',1e-9));
r_1=Y1(end,1:3);   r_2=Y2(end,1:3);   v_1=Y1(end,4:6);    v_2=Y2(end,4:6);
%axis('square'); plot3(Y1(:,1),Y1(:,2),Y1(:,3),'-','LineWidth',1); plot3(Y2
    (:,1),Y2(:,2),Y2(:,3),'-','LineWidth',1);
```

gettheta1.m

```
function [theta,Del_r] = gettheta1(a, b, c, d,r, r_rel, r_T)
 % Find the burn angles for increasing the miss vector. Equns (5-7)
    K_mv =(a^2 + c^2)*(dot(r,r_rel)^2) + (b^2 + d^2)*(dot(r_T,r_rel)^2) +...
        2*(a*b + c*d) *(dot(r_T,r_rel))*(dot(r,r_rel));
    K_mv=sqrt(K_mv);

    % In-plane angle of the burn in RTN frame.
    sin_theta = (c*(dot(r,r_rel)) + d*dot(r_T,r_rel))/K_mv;
    cos_theta = (a*(dot(r,r_rel)) + b*dot(r_T,r_rel))/K_mv;
    theta = asin(sin_theta);  theta2 = acos(cos_theta);

    Del_r = [a*cos(theta)+ c*sin(theta); b*cos(theta)+d*sin(theta); 0];
    J=dot(Del_r,r_rel);
    if J<0; theta=-theta;end
end
```

gettheta2.m

```
function [theta,Del_r] = gettheta2(a, b, c, d, r, r_T, r_rel, h_rel, sigma_x
    , sigma_z, rho_xz)
 % Find the burn angles for the steepest gradient Burn. Equns (10-12)
    D_1 = dot((a*r + b*r_T),(r_rel' - rho_xz*(sigma_x/sigma_z)*h_rel));
    D_2 = dot((c*r + d*r_T),(r_rel' - rho_xz*(sigma_x/sigma_z)*h_rel));
 %   D_3 = dot(e*r_N,(r_rel' - rho_xz*(sigma_x/sigma_x)*h_rel));

    % In plane angle of the burn in RTN frame.
    sin_theta = D_2/(sqrt(D_1^2 + D_2^2));
%     cos_theta = D_1/(sqrt(D_1^2 + D_2^2));
    theta = asin(sin_theta); % theta2 = acos(cos_theta);

    Del_r = [a*cos(theta)+ c*sin(theta); b*cos(theta)+d*sin(theta); 0];
    J=dot(Del_r , r_rel' - rho_xz*(sigma_x/sigma_z)*h_rel);
    if J<0; theta=-theta;end
```

orbit.m

```
function f = orbit(t,y)

%%%This function turns our second order two body problem equation
%%%into a system of first order differential equations.
%%%Inputs: t = time
%%%        y = some position
%%%Output: used in ode45.
```

```
mu = 398600.5;
f = zeros(6,1);
f(1) = y(4);
f(2) = y(5);
f(3) = y(6);
f(4) = -(mu*y(1))/((y(1))^2+(y(2))^2+(y(3))^2)^(3/2);
f(5) = -(mu*y(2))/((y(1))^2+(y(2))^2+(y(3))^2)^(3/2);
f(6) = -(mu*y(3))/((y(1))^2+(y(2))^2+(y(3))^2)^(3/2);
end
```

prob_collision6.m

```
function[p_int] = prob_collision6(pos_1, vel_1, pos_2, vel_2, cov_1, cov_2)
% This computes the probability of collision using a two dimensional
    integral.

% hard body radius in kilometers
dist = .015;

% RIC frame for object 1
R_1=pos_1/norm(pos_1); V_1=vel_1/norm(vel_1);
h_1=cross(R_1,V_1);     C_1=h_1/norm(h_1);     I_1=cross(C_1,R_1);
% converts from inertial cartesian frame to RIC #1 frame
M_1=zeros(3,3); M_1(1,:)=R_1; M_1(2,:)=I_1; M_1(3,:)=C_1;
cov_1n = M_1'*cov_1*M_1;

% RIC frame for object 2
R_2=pos_2/norm(pos_2); V_2=vel_2/norm(vel_2);
h_2=cross(R_2,V_2);     C_2=h_2/norm(h_2);     I_2=cross(C_2,R_2);
% converts from inertial cartesian frame to RIC #2 frame
M_2=zeros(3,3); M_2(1,:)=R_2; M_2(2,:)=I_2; M_2(3,:)=C_2;
cov_2n = M_2'*cov_2*M_2;

% relative velocity and two arbitray orthogonal vectors frame
vel_r=vel_2-vel_1; pos_r=pos_2-pos_1;
R_r=pos_r/norm(pos_r); V_r=vel_r/norm(vel_r);
h_r=cross(R_r,V_r);     C_r=h_r/norm(h_r);     I_r=cross(C_r,V_r);
% converts from inertial cartesian frame to relative velocity frame
M=zeros(3,3); M(:,1)=C_r; M(:,2)=V_r; M(:,3)=I_r;

G=M*(cov_1n+cov_2n)*M';   pos_r2=M'*pos_r';   mu=pos_r2(1:2:3);

g=G(1:2:3,1:2:3); ginv=inv(g);
a = ginv(1,1); b = ginv(1,2); c = ginv(2,1); d = ginv(2,2);

% limits of integration are [m,n]
n=dist; m=-n; tol = 1.0e-9;
int1 = @(x,y)exp(-.5*(a*(x-mu(1)).^2 + (c+b)*(x-mu(1)).*(y-mu(2))+ d*(y-mu
    (2))^2)).*(x.^2+y.^2<n^2);
out2=dblquad(int1, m, n, m, n, tol);
p_int = 1/(2*pi*sqrt(det(g)))*out2;
```

secant2.m

```
function [x0 ] = secant2(x0, x1, n, f ,tol)
%UNTITLED2 Summary of this function goes here
% x0 = first initial value
% x1 = second initial value
% n = the number of iterations the method executes before it gives up
% the function which we are finding the root, f(x).
%Secant method
format long
err = abs(x1 - x0);      % error

fx0 = f(x0);
fx1 = f(x1);

if abs(fx0) > abs(fx1)
    temp = x0;    % If the value of the function at second point
    x0 = x1;      % x1 were smaller than the value of the function
    x1 = temp;    % at the first point x0, then we interchange
    temp = fx0;   % the values so that the point with the lower
    fx0 = fx1;    % index has a lower functional value.
    fx1 = temp;
end

k = 0;
while err > tol
    x_new = (x1 - x0)/(fx1 - fx0);
    x1 = x0;
    fx1 = fx0;
    x_new = x_new * fx0;
    if abs(x_new) < tol
        break;
    end
    x0 = x0 - x_new;
    fx0 =  f(x0);
    err = abs(x1 - x0);
    k = k + 1;
    if k > n
        disp('Method failed to converge')
    end
end
% fprintf('root is %4.16f.\n', x0);
end
```

t_nominal.m

```
function [t] = t_nominal(r_1,r_2, v_1, v_2)
options = odeset('RelTol',1e-9,'AbsTol',1e-9);
y0 = cat(2,r_1,v_1)';
x0 = cat(2,r_2, v_2)';
tspan = 0:1:6000;
[tp1, pv1] = ode45(@orbit, tspan, y0, options);
[tp2, pv2] = ode45(@orbit, tspan, x0, options);
rp = pv1(:,1:3)-pv2(:,1:3);
[mins,t] = min(diag(rp*rp'));
end
```

C.2.3 Supplemental Code: Analytic Form

analytic_form.m

```
function Analytic_form()  %(data, period, start_pnt, crcl_num)

% The function Analytic_form transforms given periodic data to its analytic
    form.
% Input:
%           data     = vector of periodic data needed to be transformed to
    its analytic form
%           period   = period of the data
%           start_pnt = # of the vector element, that is
%                       starting point of the first circle
%           crcl_num  = number of total circles
% Output:   Analytic_form.txt = text file containing the analytic
%                               expression of the give data
%
% Volodymyr Kondratenko        Math Clinic      Spring 2011
%------------------------------------------
%
%https://ccrma.stanford.edu/~jos/mdft/mdft.html
%
% Comments: According to the real situation the data needed to be
% described is periodic and decaying. We assume, that it is decaying
% linearly and so the resulting function F(x)=Tr(x)+P(x), where
% Tr(x)- describes the periodicity of the function;
% P(x) - linear polynom, that describes the decay of the function
% We first interpolate function on one cicle and find Tr(x),
% then interpolate P(x)=F(x)-Tr(x), by finding the linear function, which
% is line with the start of the first cicle, as the first point and start of
     the
% last given cicle, as the second point.

data = [1 1 2 2 3 3 4 4];
period = 2;
start_pnt = 2;
crcl_num = 3;

% Cut the data, throwing away the elements, that are out of the cicle

data = data(start_pnt:(start_pnt+period*crcl_num-1));

% Finding the coefficients for fft
tri_coef=fft(data(1:period));
display('Coefficients_for_trigonometric_interpolation');
tri_coef
tri_size=size(tri_coef);

% To find the coefficients for P, we evaluate Tr(x1), Tr(x2), where x1,x2
% starting points of first and last cicle
compl=complex(0,1);
x1=data(1);
x2=data((crcl_num-1)*period+1);
Tr1=0;
Tr2=0;
```

```
if (mod(tri_size(2),2)==1)
a=(tri_size -1)/2;
    for i=1:a
        Tr1=Tr1+tri_coef(i+1)*exp(compl*i*x1)+tri_coef(i+1+a)*exp(-(tri_size
            (2)-i-a)*compl*x1);
        Tr2=Tr2+tri_coef(i+1)*exp(compl*i*x2)+tri_coef(i+1+a)*exp(-(tri_size
            (2)-i-a)*compl*x2);
    end
    Tr1=(Tr1+tri_coef(1))/tri_size;
    Tr2=(Tr2+tri_coef(1))/tri_size;
else
    for i=1:tri_size/2
        Tr1=Tr1+tri_coef(i)*exp(compl*(i-1)*x1)+tri_coef(i+tri_size(2))*exp
            (-(tri_size(2)-i-a)*compl*x1);
        Tr2=Tr2+tri_coef(i)*exp(compl*i*x2)+tri_coef(i+1+a)*exp(-(tri_size
            (2)/2+i-1)*compl*x2);
    end
    Tr1=tri_coef(1)/tri_size(2);
    Tr2=tri_coef(1)/tri_size(2);

end

y1=x1-Tr1;
y2=x2-Tr2;
z1=1;
z2=(crcl_num -1)*period+1;

P1=y1-z1*(y2-y1)/(z2-z1);
P2=(y2-y1)/(z2-z1);

% P(x)=P1+P2*x;
display('P(x)=P1+P2*x');
P1
P2


%Tr1+P1+P2
%data(1)


%Tr2+P1+P2*x2
%data(x2)

%
x=5;
res=form_calc(x,tri_coef,P1,P2)
```

Kepler.m

```
function Y = Kepler(N,p,t,u, S)
% Generates a Cartesian or a Keplerian element set.
% Usage:  Y = Kepler(N,p,t,u, S)
% Purpose1: Given a vector of Cartesian state vector, time history as posi-
%           tion and velocity, of an orbiting, object this function genera-
%           tes a vector of the corresponding Keplerian element set and
%           plot the elements as a functoin of time.
% Purpose 2: Given a vector of Keplerian  element set, (a,e,i,Omega, w, M),
%           this function calculates and generates a vector of the corresp-
%           onding Cartesian state vector.
% N = 1  - Change Cartesian coordinates into Keplerian elements.
% N = 2  - Change Keplerian elements into Cartesian sets.
% Note: If N = 2 you can set p = t = u = [];
% p = 1  - Plot the kepletian element sets against time.
% p = 2  - Hold the plotting.
% t = time column in units of seconds, minutes, or days.
% u = 1 - The units of time is seconds.
% u = 2 - The units of time is minutes.
% u = 3 - The units of time is days.
% S = input martix with 6 columns. If converting from Cartesian to Kepler
%      it should be the position and veolcity of satellite. If converting
%      from Keplerian to Cartesian the order of the elements should be
%      (a, e, i, Omega, w, M)
% Keplerian elements: semi-major axis (a), eccentricity (e),
%           inclination (I),  right ascention of ascending node (Omega),
%           argument of perigee (w), and mean anomaly (M)
%
% Math Clinic      February 2011     YDG
Y = [];
if N == 1
    %format long
    [m, n] = size(S);
    for j = 1:m
        r = S(j,1:3);              v = S(j,4:6);
        mu = 398600.4418;              % G * M_Earth (km^3/s^2).
        R = [1 0 0; 0 1 0; 0 0 1];       % Unit vectors in Cartesian
            coordinate

        h = cross(r, v);           % Angular momentum vector
        n = cross(R(3,:), h);      % The node vector.
        % Eccentricity vector.
        e_vec = (1/mu)*((norm(v)^2 - (mu/norm(r))) * r - dot(r,v)* v );
        e = norm(e_vec);                            % Eccentricity (unitless).
        a = (dot(h, h))/ (mu*(1 - e^2));            % Semi-major axis  (km).
        i = acos(dot(R(3,:), h)/ norm(h)) * (180/ pi); % Inclination (deg)
        %------------- Right ascension of ascending node ----------
        if (dot(n, R(2,:))) < 0
            Omega  = 360 - acos((dot(R(1,:), n))/(norm(n))) * (180/ pi);
        else
            Omega  = acos((dot(R(1,:), n))/(norm(n))) * (180/ pi);
        end
        %-------------  Argument of perigee -----------
        if (dot(e_vec, R(3,:))) < 0
            w = 360 -acos((dot(n, e_vec))/(norm(n)*e)) * (180/pi);
        else
            w = acos((dot(n, e_vec))/(norm(n)*e)) * (180/pi);
        end
        %-------------  True anomaly (theta) ----------
        if (dot(r, v)) < 0
            theta = 360 - acos((dot(e_vec, r))/(norm(r)*e)) * (180/pi);
        else
            theta = acos((dot(e_vec, r))/(norm(r)*e)) * (180/pi);
        end
        %--------------- Mean anomaly (M) ------------
        if (theta > 90) && (theta < 360)
            E = 360 - (acos((dot(e_vec, r))/(norm(r)*e)) * (180/pi));
        else
            E = acos((e + cos(theta))/(1+ e*cos(theta)))* (180/pi);
        end
        M = E - e*sin(E);

        Y = [Y; a,e,i,Omega,w, M];  % Append to output
    end     % end foo for loop.

    % Plot the keplerian elements as a function of time
    if p == 1
        if u == 1
            unit = 'Time_(Sec)';
        elseif u == 2
            unit = 'Time_(Min)';
        elseif u == 3
            unit = 'Time_(Days)';
        else
            disp('Not_valid_tim_unit')
        end

        if length(t) == m
            figure;
            subplot(3,2,1);
                plot(t, Y(:,1), 'k');
                xlabel(unit);       ylabel('Semi-major_axis_(km)')
            subplot(3,2,2);
                plot(t, Y(:,2), 'k');
                xlabel(unit);       ylabel('Eccentricity_(Unitless)')
            subplot(3,2,3);
                plot(t, Y(:,3), 'k');
                xlabel(unit);       ylabel('Inclination_(deg)')
            subplot(3,2,4);
                plot(t, Y(:,4), 'k');
                xlabel(unit);
                ylabel('Right_ascension_of_ascending_node_(deg)')
            subplot(3,2,5);
                plot(t, Y(:,5), 'k');
                xlabel(unit);       ylabel('Argument_of_perigee_(deg)')
            subplot(3,2,6);
                plot(t, Y(:,6), 'k');
                xlabel(unit);       ylabel('Mean_anomaly_(deg)');
        else
            error('Dimension_mismatch')
        end
    elseif p == 2
        disp('No_plot_displayed.')
        t = [];
    else
        error('Not_a_valid_plot_choice')
    end      % end of plot choice
%------------------------------------------------------------
elseif N == 2          % input form {a, e, i, Omega, w, M}
    format short
    clearvars [m, n, a, e, i, Omega, w, M]
    [m, n] = size(S);
    if n == 6
        for j = 1:m
            mu = 398600.4418;                   % G * M_Earth (km^3/s^2).
            a = S(j,1);            e = S(j,2);            i = S(j,3);
            Omega = S(j,4);        w = S(j,5);            M = S(j,6);
            % Compute eccentric anomaly.
            err = 10e-15;          % Allwed uncertainty in E.
            k = 1;                 % Index for loop.
            E(k) = M;              % Intinialize E.
            tol = 1;               % Initialize tolerance.
            while  tol > err
                E(k+1) = E(k)-((E(k)-e*sin(E(k))-M)/(1-e*cos(E(k))));
                tol = abs(E(k+1) - E(k));
                if tol < err
                    break;
                end
                k = k+1;
            end     % end of while
            E = E(k+1);
            % Compute the unit vectors P and Q
            P = [cos(w)*cos(Omega) - sin(w)*cos(i)*sin(Omega), ...
                 cos(w)*sin(Omega) + sin(w)*cos(i)*cos(Omega), ...
                 sin(w)*sin(i)];
            Q = [-sin(w)*cos(Omega) - cos(w)*cos(i)*sin(Omega), ...
                 -sin(w)*sin(Omega) + cos(w)*cos(i)*cos(Omega), ...
                  sin(i)*cos(w)];

            r = a*(cos(E) - e)*P + a*sqrt(1 - e^2) * sin(E)*Q;
            E_dot = sqrt(mu/a^3) * (1/(1 - e*cos(E)));
            v = -a*sin(E)*E_dot*P + a*sqrt(1 - e^2)*cos(E)*E_dot*Q;

            Y = [Y; r(1), r(2), r(3), v(1), v(2), v(3)];
        end
    elseif n > 6
        disp('The first six columns of S are used.')
    else
        error('Not enough columns.')
    end
else
    error('Not a valid choice of problems.')
end    % end of problem choice.
```

Kepler_test.m

```
function Kepler_test
clc;      clear;

tspan = [0 20000];
options = odeset('Refine',6,'RelTol',1e-13,'AbsTol',1e-13);

IC =  [-2436.45 -2436.45 6891.037 5.08611 -5.088611 0.0];

[t,S_out] = ode45(@odefun, tspan, IC,options);
```

```
y_1 = Kepler_IV(1,1, 1, t, S_out);

%S = [7567.18, 0.06, 55.77, 134.99, 351.83, 124.65];

%y_2 = Kepler_IV(2, 1, 1,1, y_1);

function dS = odefun(t, S_in)
mu = 398658.3;
dS(1:3, 1) = S_in(4:6, 1);
dS(4:6, 1) = - mu * S_in(1:3, 1) / (norm(S_in(1:3,1)))^3;
```

form_calc.m

```
function res=form_calc(x,tri_coef,P1,P2)
% Calculates the F(x), by given Trigonometric coeficients and Polynomial
% coefficients
tri_size=size(tri_coef);
Tr1=0;
if (mod(tri_size(2),2)==1)
a=(tri_size -1)/2;
    for i=1:a
        Tr1=Tr1+tri_coef(i+1)*exp(compl*i*x)+tri_coef(i+1+a)*exp(-(tri_size
            (2)-i-a)*compl*x);
    end
    Tr1=(Tr1+tri_coef(1))/tri_size;
else
    for i=1:tri_size/2
        Tr1=Tr1+tri_coef(i)*exp(compl*(i-1)*x)+tri_coef(i+tri_size(2))*exp
            (-(tri_size(2)-i-a)*compl*x);
    end
    Tr1=tri_coef(1)/tri_size(2);

end

res=Tr1+P1+P2*x;
```

run_analytic_form.m

```
function data_out=run_analytic_form(data, period,sample_num,dt)
% data - time - 1st column, 2-4 Cartezian coordinates position; 5-7 -
    velocity. Size N by 7
% period - array of six periods for 6 keplerian elements:  (a, e, i, Omega,
    w, M)
% Keplerian elements: semi-major axis (a), eccentricity (e),
%           inclination (I),  right ascention of ascending node (Omega),
%           argument of perigee (w), and mean anomaly (M)
% sample_num - size of the sample for the output data
% dt - timestep for output data in seconds
% start_pnt assumed to be 1
% crcl_num= kepl(:,i) mod period(i)
datasize=size(data);
t = data(:,1);
```

```
t = t*(3600*24);
time=t(1);

% 1st Step: Convert data to Keplerian elements

Y = Kepler(1,1,2,t,3,data);

% Keplerian elements: semi-major axis (a), eccentricity (e),
%            inclination (I),  right ascention of ascending node (Omega),
%            argument of perigee (w), and mean anomaly (M)
%
%   Y = [a,e,i,Omega,w, M];



% 2nd Step: Build Analytic function for elements and create a bigger sample
% for each of them separately
start_pnt=1;

for j=1:6


    crcl_num=floor(datasize(1)/period(j));
    data1 = Y(:,j);
    period1=period(j);
    [tri_coef,P1,P2] = Analytic_form(data1, period1, start_pnt, crcl_num);

    for i=1:sample_num
        time=time+dt;
        kepl(i,j)=form_calc(time,tri_coef,P1,P2);
    end
 %    kepl(10:20,j)
 %    time=t(1);
end


% 3rd step: Convert Keplerian elements back to Cartesian
time_end=time+(sample_num-1)*dt;
time1=time:dt:time_end;

%time1=time1';
%kepl=kepl' ;
size(time1)';
size(kepl);
%time1(10:20)
%kepl(6,10:20)

data_out = Kepler(2,1,2,time1,3,kepl);
```

C.2.4 Supplemental Code: Homework Solutions

We include the solution code to the homework problems given in Appendix B, that also gives some general suggestions for other programming assignments.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                                      %%%
%%%                              CRASH.m                                 %%%
%%%                                                                      %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%AE

% This file implements a series of functions that can be used to solve the
% different problems from homework assignments 1 and 2. I usually try to
% follow some set of coding guidelines, and encourage you to do the same:

% 1. Documentation: add comments and give sufficient explanation for others
% to understand your code; use meaningful variable names and minimize the
% number of implementation "tricks" that nobody but you will understand

% 2. Modularization: put as much code as possible into separate functions,
% that you can reuse later for other coding tasks without major adjustment;
% in particular, minimize the number of "hard" programming decisions, use
% actual numbers only for actual constants, and otherwise use variables

% 3. Error Handling (my code is still very weak on this, maybe later): try
% to detect possible errors in input and fix them or at least send warning
% to user rather than have the code crash (e.g., row versus column vectors,
% plotting time versus position/velocity vectors of different sizes, etc)

function crash % There are several reasons to write code using functions:
% 1. You can include subfunctions, which is not possible for procecures
% 2. You can call the function without accidently using older variables
% 3. You enable the distinction to already exisiting MATLAB functions

format short; format compact; warning off; % That's the way I like it.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                                      %%%
%%%          Some General Settings for Homework 1 and Homework 2         %%%
%%%                                                                      %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Specify settings for ode solver (reduce tolerance, discused in class)
options_ode = odeset('RelTol',10^-9);

%%% Specify settings for optimization solver (disable "annoying" output)
options_optim = optimset('display','off');

%%% Nominal Initial Position and Velocity of Object 1
pos1_0_nom = [-2436.45     -2436.45      6891.037];
vel1_0     = [5.088611     -5.088611     0.0];

%%% Nominal Initial Position and Velocity of Object 2
pos2_0_nom = [-5351.66     1596.76       5310.02];
vel2_0     = [-2.471876    5.370908      -4.099681];

%%% Initial State Error Covariance Matrix (both objects)
sigma = 0*[0.613   -0.271   -0.018;
          -0.271    0.613   -0.144;
          -0.018   -0.144    0.312];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                                %%%
%%%                          Homework 1                            %%%
%%%                                                                %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if true% true to run (the easiest way to disable parts of your code)

    %%% Specify time span for ode solver
    tspan = [0 1000]*60; % don't forget to convert from minutes to seconds

    %%% Solve two-body problem (call internal Matlab solver ode45)
    [ time_points , pos_vel_vec ] = ...
        ode45( @odefun, tspan, [ pos1_0_nom vel1_0 ], options_ode );

    % Plot trajectories (call internal subfunction) in new figure
    figure; plotTrajectories( time_points , pos_vel_vec );

end

return


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                                %%%
%%%                          Homework 2                            %%%
%%%                                                                %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% For homework 2, I changed essentially all of my initial code (that solved
% the problems but was very "sequential") into subfunctions that we should
% be able to use also for later parts of the project (modulo improvements,
% of course). Although I did not expect you to do the same, I hope you
% recognize the beauty and power of such an approach to write (new) code
% that is relatively short (of course, a lot of work is hidden in the
% subfunctions), and very easy to use and modify. Have some fun with it!

%%% NOTE: Positions, velocities, and covariances are specified above.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                             Part 1                             %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nbRuns = 10; % specify number of runs, everything else is now automatic

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Do Monte Carlo simulation for first object (compare function at bottom)
[time_points1, pos_vel_vec1] = mc45pert(nbRuns, @odefun, [0 60000], ...
    pos1_0_nom, vel1_0, sigma, [], options_ode); % returns cell structures

%%% Plot at most 10 perturbed trajectories (otherwise takes too much time)
figure; for k = 1 : min( nbRuns, 10 );
    plotTrajectories(time_points1{k}, pos_vel_vec1{k});
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                         Parts 2 and 3                            %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nbTrials = 1000;    % number of trials / Monte Carlo simulations
tspan = [0 50]*60;  % span between initial time and nominal crash
collDist = .1;      % minimum separation distance / collision radius
seed = [];          % optional random seed to repeat runs (can use [])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Compute collision probability at nominal tcpa
    [cp_nominal, seed] = mcCollProb('nominal', collDist, nbTrials, @odefun,
        tspan, ...
        pos1_0_nom, vel1_0, pos2_0_nom, vel2_0, sigma, seed, options_ode);

%%% Compute collision probability at actual tcpa (optimization)
    [cp_actual, seed] = mcCollProb('actual', collDist, nbTrials, @odefun,
        tspan, ...
        pos1_0_nom, vel1_0, pos2_0_nom, vel2_0, sigma, seed, options_ode,
            options_optim);

%%% Compute collision probability at approximate tcpa (linear approx)
    [cp_approx, seed] = mcCollProb('approx', collDist, nbTrials, @odefun,
        tspan, ...
        pos1_0_nom, vel1_0, pos2_0_nom, vel2_0, sigma, seed, options_ode);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                                  %%%
%%%                                                                  %%%
%%%     SUBFUNCTIONS (that's where all the hard work is happening)   %%%
%%%                                                                  %%%
%%%                                                                  %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function mat = rot_mat_x(theta)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% rotation matrix around x-axis (input requires angle in radians)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mat = [1 0 0; 0 cos(theta) -sin(theta); 0 sin(theta) cos(theta)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function mat = rot_mat_y(theta)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% rotation matrix around y-axis (input requires angle in radians)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mat = [cos(theta) 0 sin(theta); 0 1 0; -sin(theta) 0 cos(theta)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function mat = rot_mat_z(theta)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% rotation matrix around z-axis (input requires angle in radians)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mat = [cos(theta) -sin(theta) 0; sin(theta) cos(theta) 0; 0 0 1];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function seed = initRandomGenerator(seed)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% creates new random seed if seed = []; initializes random number generator
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if isempty(seed); seed = sum(fix(clock)); end; randn( 'state', seed );

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function dr = odefun(t,r)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% encodes first-order system of the two-body problem differential equation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
muE = 398600.5;
dr(1:3) = r(4:6);
dr(4:6) = -muE*r(1:3)/norm(r(1:3))^3;
dr = dr(:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function rnd = rndgauss(mu,sigma,varargin)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% returns matrix of Gaussian random vectors (in columns)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin > 2; N = varargin{1}; else N = 1; end
mu = mu(:);
n = length(mu);
[U,D,V] = svd(sigma);
rnd = (U*sqrt(D))*randn(n,N) + mu*ones(1,N);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function d = dist(t, odefun, pos1, vel1, pos2, vel2, t0, options)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% returns distance between two objects at time t with initial values at t0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if t == t0; d = norm(pos1-pos2); else
    [tp1,pvv1] = ode45( odefun, [t0,t], [pos1 vel1], options);
    [tp2,pvv2] = ode45( odefun, [t0,t], [pos2 vel2], options);
    d = norm( pvv1(end,1:3) - pvv2(end,1:3) );
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [tp, pvv, seed] = mc45pert( ...
    n, odefun, tspan, pos0, vel0, cov, seed, options )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Monte Carlo Simulation of perturbed trajectories with given initial data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% inputs:   n = Monte Carlo sample size, number of perturbations
%           odefun = function handle to DE that describes motion
%           tspan = 2D time span in the form [t@beginning t@end]
%           pos0, vel0, cov = position, velocity, and covariance
%           seed = random seed, can be number or empty matrix []
%           options = additional set of option settings for ODE
% outputs:  tp = cell structure of list of time point histories
%           pvv = cell structure of n position velocity vectors
%           seed = the random seed used, repeats value if given
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

seed = initRandomGenerator(seed); % initializes random number generator
```

```
%% Simulate n actual (perturbed) initial positions of given object
rnd0 = rndgauss( pos0, cov, n ); %mvnrnd( pos0, cov, n );

%% Compute the corresponding perturbed trajectories
for k = 1:n; [ tp{k}, pvv{k} ] = ...
    ode45( odefun, tspan, [rnd0(:,k)' vel0], options);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [collProb, seed] = mcCollProb(test, collDist, n, odefun, ...
    tspan, pos1, vel1, pos2, vel2, cov, seed, options_ode, varargin);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%function [collProb, seed] = mcCollProb(test, collDist, n, odefun, tspan,
    ...
%   pos1, vel1, pos2, vel2, cov, seed, options_ode, [options_optim]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% inputs:   odefun, pos1, vel1, pos2, vel2, cov, seed, options_ode (clear)
%           test = string indicating the collision test to use
%           - 'nomial' = nominal collision time (for 2.2. only)
%           - 'actual' = actual collision time (unconstrained optimization)
%           - 'approx' = approximate collision time (linear appromization)
%           n = number of trials
%           tspan = [t0 tX] = span between initial time and nominal tcpa
% outputs:  collProb = collision probability
%           seed = the random seed that was used
% Note that this function produces output (could be moved to main file.)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic; seed = initRandomGenerator(seed); % initialize timer and RN generator

% Collect and assign optional optimization settings
if nargin == 13; options_optim = varargin{1}; end;

fprintf('MC Crash Test (%s tcpa, %d trials, seed %g)\n', test, n, seed);

%% Run Monte Carlo simulations to perturb trajectories for the two objects
[tp1, pvv1] = mc45pert(n, odefun, tspan, pos1, vel1, cov, seed, options_ode)
    ;
[tp2, pvv2] = mc45pert(n, odefun, tspan, pos2, vel2, cov, seed, options_ode)
    ;

%% Initialize collision counter and start main loop
collisionCounter = 0; for k = 1:n

    %% Collect actual positions and velocities at nominal tcpa
    pos1_X_nom = pvv1{k}(end,1:3); vel1_X_nom = pvv1{k}(end,4:6);
    pos2_X_nom = pvv2{k}(end,1:3); vel2_X_nom = pvv2{k}(end,4:6);

    %% Compute relative positions and velocities at nominal tcpa
    posR_X_nom = pos1_X_nom - pos2_X_nom;
    velR_X_nom = vel1_X_nom - vel2_X_nom;

    switch test %% Compute separation distances using different methods

        case 'nominal' % nominal collision time (for 2.2. only)
            dcpa = norm( pos1_X_nom - pos2_X_nom );
            tcpa = tspan(2);

        case 'approx'   % linear approximation of trajectories
            delta_t = - (posR_X_nom * velR_X_nom')/ ...
                        (velR_X_nom * velR_X_nom');
            tcpa = tspan(2) + delta_t;
            dcpa = norm( posR_X_nom + delta_t * velR_X_nom );

        case 'actual'   % actual distance minimization over time
            [tcpa, dcpa] = fminunc( @(t) dist( t, odefun, ...
            pos1_X_nom, vel1_X_nom, pos2_X_nom, vel2_X_nom, ...
            tspan(2), options_ode), tspan(2), options_optim );
            %Function fminsearch works the same but was slower.
    end

    %%% Count and entertain yourself with collisions
    if dcpa <= collDist; collisionCounter = collisionCounter + 1;
        fprintf('Collision %d in trial %d at t = %g minutes\n', ...
            collisionCounter, k, tcpa/60); end

    end; tictoc = toc; % end main loop and timer

%%% Finally, compute and output collision probability
collProb = collisionCounter/n;
fprintf('Collision Probability %d/%d = %g (%g seconds)\n', ...
collisionCounter, n, collProb, tictoc);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [status, fid] = writeTimeHistoryFile( filename, tp, pvv )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% writes time history [tp, pvv] into tab-separated file called 'name.txt'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fid = fopen([filename, '.txt'], 'w');
for i = 1:length(tp);
    fprintf(fid, '%g\t', tp(i), pvv(i,:)); fprintf(fid, '\n');
end
status = fclose(fid);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function plotTrajectories( tp, pvv, varargin )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% creates 4x2 matrix plot of object positions and velocities
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% inputs:   tp = Times Points
%           pvv = Position Velocity Vector
% varargin (optional arguments):
%           color = string of plot color ('k', 'r', 'b', 'g', etc)
% outputs:  none
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if nargin > 3 color = varargin{3}; else color = 'k'; end % default color

    % 3D position trajectory (parametric curve)
    subplot(4,2,1); hold on;
        title('Position Trajectory');
        xlabel('x'); ylabel('y'); zlabel('z');
        plot3( pvv(:,1), pvv(:,2), pvv(:,3), [color,'-'] );
```

```
% 3D velocity trajectory (parametric curve)
subplot(4,2,2); hold on;
    title('Velocity Trajectory');
    xlabel('v_x'); ylabel('v_y'); zlabel('v_z');
    plot3( pvv(:,4), pvv(:,5), pvv(:,6), [color,'-'] );
% 2D time versus position curves (x, y, z in same plot)
subplot(4,2,3); hold on;
    title('Time versus Position');
    xlabel('time'); ylabel('position');
    plot(   tp, pvv(:,1), [color,'-'], ...
            tp, pvv(:,2), [color,'-.']', ...
            tp, pvv(:,3), [color,':'] );
    legend('x','y','z');
% 2D time versus velocity curves (x, y, z in same plot)
subplot(4,2,4); hold on;
    title('Time versus Velocity');
    xlabel('time'); ylabel('velocity');
    plot( tp, pvv(:,4), [color,'-'], ...
          tp, pvv(:,5), [color,'-.'], ...
          tp, pvv(:,6), [color,':'] )
    legend('v_x','v_y','v_z');
% 2D time versus total position (measured by L2 norm)
subplot(4,2,5); hold on;
    title('Time versus Total Position (L_2-norm)');
    xlabel('time'); ylabel('total position');
    plot( tp, sqrt( sum( pvv(:,1:3).^2, 2 ) ), [color,'-'] )
% 2D time versus total velocity (measured by L2 norm)
subplot(4,2,6); hold on;
    title('Time versus Total Velocity (L_2-norm)');
    xlabel('time'); ylabel('total velocity');
    plot( tp, sqrt( sum( pvv(:,4:6).^2, 2 ) ), [color,'-'] )
% 2D time versus total position (measured by L1 norm)
subplot(4,2,7); hold on;
    title('Time versus Total Position (L_1-norm)');
    xlabel('time'); ylabel('total position');
    plot( tp, sum( abs( pvv(:,1:3) ), 2 ), [color,'-'] )
% 2D time versus total velocity (measured by L1 norm)
subplot(4,2,8); hold on;
    title('Time versus Total Velocity (L_1-norm)');
    xlabel('time'); ylabel('total velocity');
    plot( tp, sum( abs( pvv(:,4:6) ), 2 ), [color,'-'] )
```

C.3 Ephemeris Data Files

The following data underlies the computational experiment in Chapter 5. For each satellite, the corresponding data entries are given in the form

```
time_stamp   pos_x   pos_y   pos_z   vel_x   vel_y   vel_z
```

and indicate the position and velocity at 577 specified time stamps. We only show the first, middle and last five observations each; the full data sets can be requested by e-mail from `alexander.engau@ucdenver.edu`.

sat1.txt (primary)

```
16 Jan 2005 02:14:37.212    -6981.418748067837    926.2870386137012     1273.515447075486
                            -1.20595757568536     1.371138170924282     -7.247806683812038
16 Jan 2005 02:19:37.212    -6999.440404462371    1286.14489912759      -927.3210262172742
                            1.083601106410356     1.008903344271513     -7.305002980506311
16 Jan 2005 02:24:37.212    -6343.604280937016    1522.230700366655     -3039.184114708578
                            3.250341862753424     0.5529438074274717    -6.66141674625386
16 Jan 2005 02:29:37.212    -5081.985720926802    1613.00244489037      -4861.136970012316
                            5.090810166076802     0.04800045674356464   -5.389360612351899
16 Jan 2005 02:34:37.212    -3338.623347237927    1550.925281912202     -6222.618514459669
                            6.43786915158877      -0.4580412295233666   -3.616872640678516
. . .
17 Jan 2005 02:04:37.212    -4225.862115525803    1599.701581620111     -5635.6822379328
                            5.861568253087271     -0.218022087738775    -4.520226628595808
17 Jan 2005 02:09:37.212    -2295.928656734673    1459.936681989716     -6704.810421403832
                            6.901822184410294     -0.7059429502608741   -2.552510185060476
17 Jan 2005 02:14:37.212    -150.1262498418984    1182.974974382047     -7143.967726781428
                            7.290454501452286     -1.125695816360741    -0.35316243588648
17 Jan 2005 02:19:37.212    2009.799003259059     795.1622674856794     -6913.752799779228
                            6.995624492355361     -1.439334034311279    1.875696142916376
17 Jan 2005 02:24:37.212    3981.451389655614     332.8784938149461     -6035.984298689072
                            6.044735500923345     -1.618320975783792    3.930692752136895
. . .
18 Jan 2005 01:54:37.212    3070.472505376031     561.5499845159807     -6534.52146628431
                            6.579186615327043     -1.550539350244867    2.978032157757702
18 Jan 2005 01:59:37.212    4869.241157064861     77.3491327954868      -5348.641841918648
                            5.317188365688231     -1.652033740502962    4.866480243201368
18 Jan 2005 02:04:37.212    6209.920711838697     -414.0914673953902    -3659.800839616828
                            3.548125820206969     -1.598197414253326    6.304487619308354
18 Jan 2005 02:09:37.212    6963.192613757555     -866.3207249905576    -1625.029643208934
                            1.431033414225879     -1.392344725437844    7.153375352726526
18 Jan 2005 02:14:37.212    7053.152496262693     -1235.979317957119    564.2629929186934
                            -0.8392454538967212   -1.051748220338898    7.324702968356784
```

sat2.txt (secondary-1)

```
16 Jan 2005 02:14:37.212    -4383.965242030842    -4020.483002583562    3921.074684039107
                            -3.928702489636912    -1.85833179423258     -6.103238356374871
16 Jan 2005 02:19:37.212    -5328.629535189863    -4371.985237086851    1928.001806161137
                            -2.320952991537846    -0.4689296946863907   -7.072422452556804
16 Jan 2005 02:24:37.212    -5756.266115881159    -4299.186367146359    -252.6650708287376
                            -0.5108370645853604   0.9471827302769824    -7.346809648726899
16 Jan 2005 02:29:37.212    -5632.504851993595    -3814.484909936124    -2409.568308179557
                            1.325761952278912     2.255749244817582     -6.918166575478212
16 Jan 2005 02:34:37.212    -4975.216970649736    -2968.398810037543    -4338.594093854739
                            3.018926831251242     3.338507349894456     -5.841599043383295
. . .
17 Jan 2005 02:04:37.212    -3453.414539270797    -1481.464972929916    -6203.032477188371
                            4.748605380703364     4.251824901325072     -3.679076777169923
```

```
17 Jan 2005 02:09:37.212    -1889.99042067621     -156.7166093256909    -7000.254847099077
                            5.592423862670997     4.510523973169075     -1.594049564436147
17 Jan 2005 02:14:37.212    -150.2183392874946    1182.616608866891     -7144.10122129902
                            5.915074157661759     4.348047398579944     0.6441146131042568
17 Jan 2005 02:19:37.212    1603.465544400431     2411.012297779376     -6618.529766446501
                            5.683378524788908     3.775743190263148     2.834810383480659
17 Jan 2005 02:24:37.212    3205.494712269438     3411.586479023301     -5468.142198432629
                            4.910155634719813     2.839811005412897     4.776704656467614
. . .
18 Jan 2005 01:54:37.212    4799.143121915431     4209.894389535818     -3237.654607198698
                            3.219362398794622     1.230376414543775     6.613254962000633
18 Jan 2005 01:59:37.212    5515.310502255186     4367.827176463786     -1128.312409175688
                            1.511921600289397     -0.1894020191499937   7.335075818988195
18 Jan 2005 02:04:37.212    5688.967585581911     4096.212865496261     1091.510248694548
                            -0.3678973025313385   -1.609326953053296    7.340449094913762
18 Jan 2005 02:09:37.212    5296.014336948612     3416.703401699992     3202.18043489389
                            -2.233040917709377    -2.884822574431419    6.610733755013042
18 Jan 2005 02:14:37.212    4370.075566413979     2393.40702898179      4990.218951710357
                            -3.889410899329996    -3.88033715873083     5.206346754677181
```

sat3.txt
(secondary-2)

```
16 Jan 2005 02:14:37.212    -6984.973660222872    904.3787220685445     1269.763403464822
                            -1.185725927819648    1.49582236179512      -7.226453125156158
16 Jan 2005 02:19:37.212    -6996.851865159574    1302.097624809837     -924.5889439712788
                            1.103992412967993     1.134571484395886     -7.283480909565655
16 Jan 2005 02:24:37.212    -6335.120651959349    1574.513857020191     -3030.230040859446
                            3.268936652015534     0.667540318247676     -6.641790815347538
16 Jan 2005 02:29:37.212    -5068.416262510674    1696.62869987422      -4846.815040913616
                            5.10585411673506      0.1407137556190312    -5.373482425620123
16 Jan 2005 02:34:37.212    -3321.253426642681    1657.973136307084     -6204.285375173392
                            6.447965349905754     -0.3958200834075802   -3.606216575276211
. . .
17 Jan 2005 02:04:37.212    -4210.130578590749    1696.652357281327     -5619.078336023874
                            5.874186088721133     -0.1402605214711639   -4.50690909287064
17 Jan 2005 02:09:37.212    -2277.212737643814    1575.279695235145     -6685.056643626363
                            6.908947303705229     -0.6620320568005509   -2.544989954689555
17 Jan 2005 02:14:37.212    -130.1844597600511    1305.872821203338     -7122.920099473226
                            7.291440324937332     -1.119620354571165    -0.3521219452787474
17 Jan 2005 02:19:37.212    2029.098167572776     914.099721438983      -6893.383433931886
                            6.990388643010892     -1.4716016796969      1.870169948667327
17 Jan 2005 02:24:37.212    3998.300336406961     436.7156753542577     -6018.201022950543
                            6.033763299936839     -1.685940776744206    3.919112106859614
. . .
18 Jan 2005 01:54:37.212    3088.713077053014     673.9635128135495     -6515.269395485338
                            6.570873686934285     -1.601770508328205    2.96925825040537
18 Jan 2005 01:59:37.212    4884.171444675872     169.3619456926604     -5332.883621220775
                            5.303603991966441     -1.735751915942781    4.852142571703667
18 Jan 2005 02:04:37.212    6220.136740645512     -351.1318264530675    -3649.018298731265
                            3.530527368828313     -1.706653664806989    6.285913276470953
18 Jan 2005 02:09:37.212    6967.728748474798     -838.3653013649706    -1620.241965043422
                            1.411065363538796     -1.515404411871882    7.132300008579447
18 Jan 2005 02:14:37.212    7051.577403178862     -1245.686347653161    562.6005557931413
                            -0.8596917513337044   -1.177755259821945    7.303122857119323
```