



**RADAR ORBIT ANALYSIS TOOL
USING LEAST SQUARES ESTIMATOR**

THESIS

Sandra M. Rashash, Second Lieutenant, USAF

AFIT/GSS/ENY/07-S01

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

AFIT/GSS/ENY/07-S01

**RADAR ORBIT ANALYSIS TOOL
USING LEAST SQUARES ESTIMATOR**

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the
Degree of Master of Science (Space Systems)

Sandra M. Rashash, BS

Second Lieutenant, USAF

September 2007

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

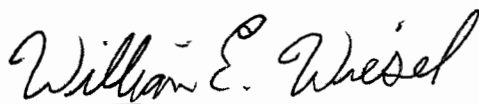
AFIT/GSS/ENY/07-S01

**RADAR ORBIT ANALYSIS TOOL
USING LEAST SQUARES ESTIMATOR**

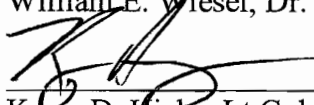
Sandra M. Rashash, BS

Second Lieutenant, USAF

Approved:



William E. Wiesel, Dr. (Chairman)

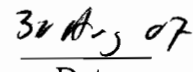


Kerry D. Hicks, Lt Col, USAF (Member)



Nathan A. Titus, Lt Col, USAF (Member)


Date


Date


Date

Abstract

Most objects tracked in space follow a regular Keplerian orbit; unfortunately, non-Keplerian objects such as maneuvering satellites, tethered systems, and thrusting ballistic missiles are becoming more common. It is important to be able to distinguish between Keplerian and non-Keplerian objects due to the potential risk of a tethered satellite being mistaken for an object on re-entry. This research focused on creating a computer model that can detect the non-gravitational acceleration present in non-Keplerian orbits. A 3rd order Taylor series expansion was used to model the dynamics and to produce simulated radar data. Linear least squares estimation was used to estimate the initial state of a space object with a state vector composed of position, velocity, acceleration, and its first derivative. Monte Carlo analysis was used to verify that the estimator was unbiased and representative of the uncertainty in the data. The Monte Carlo method detected non-gravitational acceleration as small as 1.12 cm/s^2 ; however, a subsequent approach that analyzed the data sets individually only detected acceleration as small as 10.63 cm/s^2 . At smaller magnitudes, the estimator was able to detect the presence of non-gravitational acceleration, but was ultimately unable to estimate the true value with statistical accuracy.

Table of Contents

	Page
Abstract	iv
Table of Contents	v
List of Figures	vii
List of Tables	viii
List of Symbols	x
I. Introduction	1
Background.....	1
Problem Statement.....	3
Research Objectives	3
Research Focus	3
Investigative Questions	4
Methodology.....	4
Assumptions and Limitations	5
Implications	6
Preview	6
II. Literature Review	7
Chapter Overview.....	7
Non-Keplerian Orbits	7
Galileo's Projectile Trajectory	10
Least Squares.....	13
Summary.....	20
III. Methodology	22
Chapter Overview.....	22

	Page
Raw Data	22
Truth Model.....	25
Estimator.....	32
Summary.....	36
IV. Analysis and Results.....	38
Chapter Overview.....	38
STK Simulation vs. MATLAB Model	38
MATLAB Simulations with Zero Non-Gravitational Acceleration.....	52
MATLAB Simulations with Non-Gravitational Acceleration Present	53
Investigative Questions Answered	67
Summary.....	67
V. Conclusions and Recommendations	69
Chapter Overview.....	69
Conclusions of Research	69
Significance of Research	69
Recommendations for Action.....	70
Recommendations for Future Research.....	71
Summary.....	72
Appendix A: Equations of Motion.....	73
Appendix B: MATLAB Truth Model.....	74
Appendix C: MATLAB Estimator.....	79
Appendix D: Estimation Process	83
Bibliography	84
Vita.....	86

List of Figures

	Page
Figure 1. Tether Force Components	8
Figure 2. Trajectories	11
Figure 3. Gaussian Distribution Function	13
Figure 4. Observation Geometry	23
Figure 5. Satellite Position	23
Figure 6. STK vs. MATLAB Position Error (2 nd Order)	39
Figure 7. STK vs. MATLAB Velocity Error (2 nd Order)	40
Figure 8. STK vs. MATLAB Position Error (3 rd Order)	41
Figure 9. STK vs. MATLAB Velocity Error (3 rd Order)	42
Figure 10. STK vs. MATLAB Position Error (4 th Order)	43
Figure 11. STK vs. MATLAB Velocity Error (4 th Order)	44

List of Tables

	Page
Table 1. Ascension Atlantic Radar Specifics.....	29
Table 2. STK vs. MATLAB RMS Error (3 rd Order EOM)	31
Table 3. STK vs. MATLAB RMS Error.....	44
Table 4. MATLAB Estimated State Vector with No Noise	45
Table 5. MATLAB Variance for Noiseless Scenario	46
Table 6. MATLAB Estimated State Vector with Noise	47
Table 7. State Vector Data Sets 1-5 (4 th Order).....	48
Table 8. Monte Carlo Estimated State Vector Values (4 th Order)	50
Table 9. State Vector Data Sets 1-5 (3 rd Order).....	50
Table 10. Monte Carlo Estimated State Vector Values (3 rd Order).....	51
Table 11. Estimated State Vectors	53
Table 12. Non-Gravitational Acceleration Present.....	55
Table 13. Case 1 Monte Carlo Estimated State	55
Table 14. Case 1 Estimated Non-gravitational Acceleration.....	56
Table 15. Case 1 Monte Carlo Results.....	56
Table 16. Monte Carlo Estimated State Vector	57
Table 17. Estimated Non-gravitational Acceleration for all Cases.....	57
Table 18. Case 2 Monte Carlo Results.....	57
Table 19. Case 3 Monte Carlo Results.....	58
Table 20. Case 4 Monte Carlo Results.....	58
Table 21. Case 5 Monte Carlo Results.....	58

	Page
Table 22. Case 6 Monte Carlo Results.....	59
Table 23. MATLAB Case 1 Results for Individual Data Sets.....	61
Table 24. MATLAB Case 2 Results for Individual Data Sets.....	62
Table 25. MATLAB Case 3 Results for Individual Data Sets.....	63
Table 26. MATLAB Case 4 Results for Individual Data Sets.....	64
Table 27. MATLAB Case 5 Results for Individual Data Sets.....	65

List of Symbols

Symbol		Units
r	magnitude of position.....	km
$\vec{r}$	position vector.....	km
$\dot{\vec{r}}$	velocity vector.....	km/s
$\vec{v}$	velocity vector.....	km/s
$\ddot{\vec{r}}$	acceleration vector.....	km/s ²
$\vec{a}$	acceleration vector.....	km/s ²
$\dddot{\vec{r}}$	first derivative of acceleration vector.....	km/s ³
$\vec{r}_0$	position vector at epoch.....	km
$\vec{v}_0$	velocity vector at epoch.....	km/s
$\vec{a}_0$	acceleration at epoch.....	km/s ²
x	component of position vector in the I-direction.....	km
y	component of position vector in the J-direction.....	km
z	component of position vector in the K-direction.....	km
$\dot{x}$	component of velocity vector in the I-direction.....	km/s
$\dot{y}$	component of velocity vector in the J-direction.....	km/s
$\dot{z}$	component of velocity vector in the K-direction.....	km/s
μ	standard gravitational parameter for Earth.....	km ³ /s ²
m	mass.....	kg
m_p	mass of parent satellite.....	kg

Symbol		Units
$\vec{r}_p$	position vector of parent satellite.....	km
$\vec{F}$	force.....	N
t_0	time at epoch.....	sec
g	gravity.....	km/s ²
$\bar{g}_{2-body}$	gravity due to two-body effects.....	km/s ²
$\bar{g}_{J_2}$	gravity due to J ² effects.....	km/s ²
W	weight.....	N
ρ	magnitude of range to space object.....	km
ρ_e	range component in the e-direction.....	km
ρ_s	range component in the s-direction.....	km
ρ_z	range component in the z-direction.....	km
α	azimuth angle.....	rad
β	elevation angle.....	rad
ρ_{error}	radar site range error.....	km
α_{error}	radar site azimuth error.....	rad
β_{error}	radar site elevation error.....	rad
$R_{\oplus}$	radius of the Earth.....	km
$\vec{R}_{site}$	position vector from center of Earth to radar site.....	km
$\vec{R}_{ijk}$	position vector from center of Earth to space object.....	km

Symbol		Units
$\bar{\rho}_{ijk}$	range vector from radar site to space object.....	km
σ	standard deviation.....	various units
$V_{geopotential}$	potential of Earth's gravity field.....	m ² /s ²
Φ	state transition matrix.....	various units
H	linearized observation relation matrix.....	km
$\bar{X}(t_0)$	estimated mean state vector at epoch.....	various units
$\bar{X}_0$	true state vector at epoch.....	various units
D	rotation matrix.....	rad
J	Jacobian matrix.....	various units
$\bar{Z}$	observation data for estimator.....	km
Q	instrumental covariance matrix.....	various units
P	state covariance matrix.....	various units
J_2	zonal harmonic.....	unitless
$e_{\oplus}$	eccentricity of Earth.....	unitless
H	radar site altitude.....	km
L	radar site geodetic latitude.....	rad
LST	radar site local sidereal time.....	rad

RADAR ORBIT ANALYSIS TOOL USING LEAST SQUARES ESTIMATOR

I. Introduction

Background

Radars have been used to track satellites ever since the Soviet Union successfully launched Sputnik in October 1957. These radars, along with optical sensors, eventually grew into what is commonly known today as the Space Surveillance Network (SSN). The SSN has 25 sensor sites located world-wide with the capability of tracking space objects from low earth orbit (LEO) all the way to geosynchronous earth orbit (GEO). There are currently over 10,000 objects being tracked today; however, the number of objects increases each year due to more launches and the increased capability/desire to track smaller and smaller objects. These objects include satellites, rocket debris, bolts, and sometimes even an astronaut's glove.

The various sensors used within the SSN offer their own capabilities and challenges. Optical sensors use a Charge Coupled Device (CCD) to take pictures of the satellite streaking across the sky. These images are cross-checked with star charts to determine the satellite's topocentric right ascension and declination (Vallado, 2001:243). Optical sensors, unfortunately, are not practical during the day nor do they work very well when it is cloudy or overcast.

Radars, however, can be used day or night and in cloudy conditions. Radars use both a transmitter to send electromagnetic radiation and a receiver to obtain the EM

energy reflected back from the detected object. This method obtains the topocentric azimuth and elevation of the space object. The SSN divides radars into three categories: tracking, detecting and phased array (interferometers). Interferometers use a collection of transmitters and receivers to detect objects. The transmitters create a fan of energy that is reflected back to the receivers when an object passes through. Varying the phase creates the ability to obtain direction cosines that can be manipulated to obtain elevation and azimuth data (Vallado, 2001:241).

Radars used for tracking are typically more accurate than those used for initial detection (Thomas, 1967:75-86); however, all systems contain a certain level of bias and noise. Biases in range, azimuth, and elevation are taken into account to correct the raw sensor data. Noise statistics are very useful in helping to accurately weigh the validity of the data. This is important when the data is to be run through estimators or filters.

In the 50 years since Sputnik, the ability to track space objects has become increasingly more complex due to technological advancements both foreign and domestic. Satellites can follow more than just a regular Keplerian orbit. For a given radar track, the space object may be maneuvering, part of a tethered system, or even a thrusting ballistic missile.

Tethered systems are of key interest these days. The force created by a tether changes the dynamics of the end masses and may result in the satellite being mistaken for an object on a re-entry trajectory (Asher and others, 1988:514). If the end mass is in a lower orbit than the center-of-mass, its velocity will be smaller than what should be expected from a single satellite in a Keplerian orbit; similarly, an end mass in a higher

orbit than the center-of-mass will have a higher velocity than it should (Cicci and others, 2002:340).

Problem Statement

In order to determine if an object is following a regular Keplerian orbit or has some other motion, it is necessary to determine how the dynamics between the two groups differ. One solution is to develop a model that can detect non-gravitational acceleration. This model must be easily implemented and have the ability to produce results relatively quickly. Such a model can be produced using Galileo's projectile motion dynamics and a linear least squares estimator. A *linear* least squares estimator is ideal because there is no need for an initial guess or iteration, so results are produced in a timely manner.

Research Objectives

The primary objective of this research was to create a statistically accurate computer model capable of determining if a space object has non-gravitational acceleration. A truth model and least squares estimator must be produced in order to create and test the computer model. This model is merely a filter to be used at radar sites to get an initial idea if a space object is following a non-Keplerian orbit.

Research Focus

This research focused on using the projectile equations of motion with a linear least squares estimator to solve the estimation problem instead of a sequential method such as either a Bayes filter or a Kalman filter. The state vector will include not only

position ($\vec{r}$) and velocity ($\dot{\vec{r}}$), but also acceleration ($\ddot{\vec{r}}$) and its first derivative ($\dddot{\vec{r}}$). The inclusion of acceleration in the state vector was used to verify if non-gravitational acceleration was present. The covariance must also statistically validate that the acceleration exists.

Investigative Questions

Several questions were addressed during this research. The most important question was, “What magnitude of acceleration can the model detect?” Tethered systems and maneuvering space objects can have non-gravitational accelerations that are quite small (i.e. cm/s^2 , $\mu\text{m/s}^2$). The computer model will be more versatile if it can detect these smaller accelerations. Another question of interest was, “Are the dynamics accurate enough?” Galileo’s equations for projectile motion are *very* general. A Taylor series expansion of Galileo’s equations of motion may be utilized for better accuracy. The question of whether or not the geopotential is accurate enough with just J_2 and two-body terms was also addressed.

Methodology

Solving the estimation problem required dividing the process into four stages. First, the raw radar data was converted to a more usable format. Second, a truth model that can simulate various aspects of real-world data was developed. Third, an estimator that outputs $\vec{r}$, $\dot{\vec{r}}$, $\ddot{\vec{r}}$, and $\dddot{\vec{r}}$ was created. The estimator must also produce a covariance matrix to verify that the state estimate was accurate. Last, test case scenarios were run through the truth model and estimator to ensure proper function and accuracy.

Assumptions and Limitations

A number of assumptions and limitations were taken into account and addressed during this research. The equations of motion used for the truth model were produced using Galileo's projectile trajectory with a Taylor series expansion. Two-body equations of motion are more accurate in modeling space object motion; however, those equations are nonlinear and would make the estimation problem more complex. For these equations of motion, the value for gravity includes more than the standard 9.81 m/s^2 . Gravity was instead modeled as the gradient of the geopotential taking into account both two-body and J_2 effects. Air drag was not modeled. The omission of air drag works well for a short radar track, but would limit the accuracy of the model if used for a much longer simulation. For this research, the local gravity vector was calculated only once, using the initial position vector. For such a short track of data, it was assumed that the gravity vector would not change dramatically.

This model is limited by radar capabilities and how fast the space object is accelerating. Assuming the radar has a range error of 100 meters, given a five-minute radar track, the object must be accelerating faster than 0.667 cm/s^2 in order for the model to statistically say there is an extra acceleration component present. This model is, therefore, unable to detect extremely small accelerations. It was also assumed that the sensor's total instrument error was composed of only little error sources, thereby allowing the concept of Gaussian Distribution to be taken into account during estimation.

Implications

This computer model is a simple tool that radar sites run their data through to quickly get an idea if a space object is following a non-Keplerian orbit. It is merely a filter. It can determine if non-gravitational acceleration is present; however, it cannot distinguish between a tethered system and a maneuvering satellite. A more complex model would need to be utilized to provide specific object identification and orbit propagation.

Preview

Chapter II touches on some of the key research areas studied while solving the problem at hand, the most important of which being least squares and how it all works. The importance of Galileo's projectile trajectory is also examined. A summary of previous attempts to model space objects with additional acceleration components is included. Chapter III discusses the specific processes used to create the conversion from raw radar data, the truth model, and the estimator; as well as how the routines are executed and validated. Chapter IV goes over the different test case scenarios, as well as an analysis of the data. Chapter V concludes with the end result and recommendations for future research.

II. Literature Review

Chapter Overview

The purpose of this chapter is to explain the theory behind the key concepts used in this research. Section 2.2 covers some of the previous research efforts related to space objects with non-Keplerian orbits. Section 2.3 explores Galileo's insight in projectile motion and how it can be adapted to modeling satellite motion. Section 2.4 explains the fundamental assumptions required to make least squares work, as well as how to obtain the estimated state and covariance for a set of data.

Non-Keplerian Orbits

Orbit determination of space objects has been a topic of interest ever since astronomers first tried tracking the planets and moons in the solar system. Many scientists and astronomers, including Galileo, Brahe, Kepler, and Newton provided insight into how to track these objects. It was Newton's work combined with Kepler's Planetary Laws that created the two-body equations of motion:

$$\ddot{\vec{r}} = -\frac{\mu\vec{r}}{r^3} \quad (1)$$

Equation (1) is fundamental in understanding the dynamics of orbiting objects. Many different orbit determination methods use the two-body equations of motion as the foundation.

Also noteworthy was the development of Kepler's Problem. Given an initial position and velocity of a space object, plus a time span $(t - t_0)$, the position and velocity at the future time could be determined. This method revolutionized the orbit

determination process. The propagation of objects following a Keplerian orbit was now obtainable. A thorough discussion on Kepler's Problem can be found in *Fundamentals of Astrodynamics* (Bate and others, 1971:177-203) or *Fundamentals of Astrodynamics and Applications* (Vallado, 2001:87-103).

Kepler's work has been exceedingly useful in tracking satellites orbiting Earth. The development of tethered systems, however, has presented a problem: the end masses of tethered systems do not follow a regular Keplerian orbit. The tether creates an additional acceleration in the radial and tangential directions. The observed 'daughter' satellite (m) and the unobserved 'parent' satellite (m_p) with their force components are presented below in Figure 1.

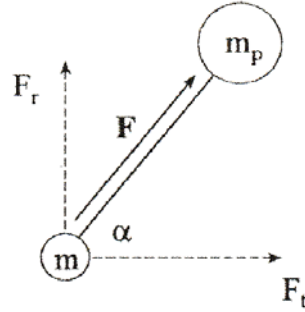


Figure 1. Tether Force Components (Cicci and others, 2001a:314)

As a result of this new development, a plethora of orbit determination methods for tethered systems have been researched over the past decade. Most techniques used to model tethered orbits assume the following equations of motion

$$\ddot{\vec{r}} = \frac{-\mu\vec{r}}{r^3} + \frac{\vec{F}}{m} \quad (2)$$

$$\ddot{\vec{r}}_p = \frac{-\mu\vec{r}_p}{r_p^3} + \frac{\vec{F}}{m_p} \quad (3)$$

Many attempts to model tethered systems have focused on estimating the additional acceleration components seen in Figure 1. Thus, the state vector is commonly written as

$$\bar{X} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ a_r \\ a_t \end{bmatrix} \quad (4)$$

Engineers have used least squares (Cicci and others, 2001a:309-326) and ridge-type estimators (Cicci and others, 2001b:297-316) to estimate the initial state of tethered systems. More in-depth techniques have been used to find libration angle, libration rate, and trajectory prediction (Cicci and others, 2002:340). These techniques use more complex dynamics and require longer arcs of observation data. Since the equations of motion listed above are non-linear, the estimation process requires an initial guess for the state.

It is not just tethered systems that contain additional acceleration components. Maneuvering space objects produce additional acceleration that creates a non-Keplerian orbit. Satellites use thrust for various reasons, such as: station keeping, rendezvous, and orbital transfers. Thrust is a non-instantaneous force that can be written as $F=ma$, using Newton's Law; therefore, when thrust is present, it is an additional acceleration that

ultimately causes a non-Keplerian orbit. Since this research primarily focused on the orbit determination of any non-Keplerian object, not just tethered systems, the equations of motion and estimation techniques mentioned above will be replaced with a more general model.

Galileo's Projectile Trajectory

In 1638, Galileo Galilei helped to redefine man's understanding of motion. Galileo's book, *Dialogues Concerning Two New Sciences* (Galilei, 1914), covered four days of experimentation and contemplation that ultimately created the foundation for projectile equations of motion. Galileo conveyed the following insights (Hahn, 2002:341):

1. All bodies falling in a vacuum do so with the same constant acceleration. For a body falling from rest, the speed is proportional to the elapsed time. This is so both in the situation of free fall and for balls rolling on an inclined plane.
2. The law of fall, namely, that the distance covered by a body moving from rest is proportional to the square of the time of the motion.
3. The trajectory of a projectile has a parabolic shape.

It is not until later, with the development of calculus, that Galileo's insights were put into the familiar form

$$v_y = v_{y_0} - gt \quad (5)$$

$$y = v_{y_0} t - \frac{1}{2} gt^2 \quad (6)$$

$$v_x = v_{x_0} \quad (7)$$

$$x = v_x t \quad (8)$$

It was Galileo's work concerning the motion of projectiles that produced the basic equations of motion used for this research. Galileo studied the properties of both uniform motion and naturally accelerated motion and proposed that a projectile is a combination of the two. The horizontal motion is produced by the launching mechanism, whereas the vertical motion is due to gravity. A fired cannonball is a prime example of projectile motion.

Satellites and other space objects can also be modeled using projectile motion (although a more accurate model would use two-body equations of motion). Figure 2 helps illustrate the basics of orbital motion.

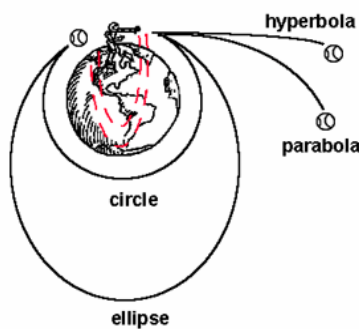


Figure 2. Trajectories (Sellers, undated:31)

Imagine a person standing on one side of the Earth throwing a baseball. The faster the ball is thrown, the further it travels. Earth, however, is not flat: it is round. Therefore, as the ball is flying through the air, the Earth is actually curving away from the ball at a rate of 5 meters for every 8 kilometers traveled. At the right velocity, the object is falling slower than the rate at which the Earth is curving away. At this point, the object has reached 'freefall,' also known as orbit. An object must be moving at 7.9 km/s (ignoring air drag) to be considered in orbit (Sellers, 2000:106). With this example in

mind, it is easy to see how Galileo's first insight listed above is applicable to space objects.

In order to find the "constant acceleration" exerted on the space object, Newton's 2nd Law is required. Rearranging Newton's 2nd Law yields $\bar{a} = \frac{\bar{F}}{m}$. In space, where weight (W) = force (F), Newton's law looks like

$$\bar{a} = \frac{\bar{F}}{m} = \frac{W}{m} \quad (9)$$

Substituting mg for W and simplifying yields

$$\bar{a} = \bar{g} \quad (10)$$

The above discussion has just shown that space objects are subject to constant acceleration in the form of gravity as long as gravity is assumed to be constant. It seems only reasonable, therefore, to be able to model objects in LEO using projectile motion, with the understanding that the model is very crude and only 'good enough' for a short period of time. Below are Galileo's equations in vector form.

$$\bar{r} = \bar{r}_0 + \bar{v}_0\Delta t + \frac{1}{2}\bar{g}\Delta t^2 \quad (11)$$

$$\bar{v} = \bar{v}_0 + \bar{g}\Delta t \quad (12)$$

$$\bar{a} = \bar{g} \quad (13)$$

A quick glance at the above equations verifies that they are actually a 2nd order Taylor series approximation. If the 2nd order does not yield accurate results, then an expansion to the 3rd order or even 4th order may prove better. The equations of motion for a 3rd order and 4th order Taylor series approximation are listed in Appendix A.

Least Squares

In 1801, German mathematician Carl Friedrich Gauss discovered a technique that would revolutionize the methods of orbit determination. Gauss' discovery, known as Least Squares, relies on several principle assumptions: the dynamics contain no error, the instruments do contain error, Gaussian distribution, and Principle of Maximum Likelihood (Hall, 1994:7). In order for the dynamics to contain no error, it is important to model the object of interest with a relevant dynamics system. The Central Limit Theorem addresses the instrument error and Gaussian distribution. The Principle of Maximum Likelihood is the final assumption required in order to produce the estimate of a state and its covariance. These assumptions are addressed in the following sub-sections.

Central Limit Theorem.

The Central Limit Theorem states that if an instrument has many little errors, then no matter how the little errors are distributed, the overall error can be described using a Gaussian function (Figure 3).

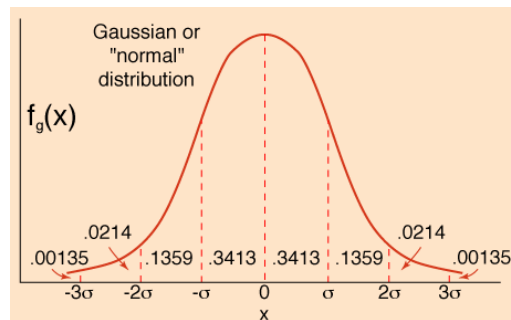


Figure 3. Gaussian Distribution Function (Zaninetti, 2002)

The Gaussian function is centered about the true value. The width of the curve is described by the standard deviation (σ). A smaller σ means a narrower curve. The

probability that the answer is within $\pm 1 \sigma$ is 68%. Within $\pm 2 \sigma$ yields a probability of ~95 % while $\pm 3 \sigma$ is 99.7%. The Gaussian Distribution can be written as

$$G_{x_0, \sigma}(X) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(X-X_0)^2}{2\sigma^2}} \quad (14)$$

The Gaussian Distribution is an important concept utilized by statisticians, scientists, and engineers.

Expectation Operator.

The expectation operator is a linear operator that facilitates the estimation process.

The expectation operator can be written as

$$E(-) \equiv \int_{-\infty}^{\infty} (-) f(X) dX \quad (15)$$

The term $f(X)$ is the probability density function and $(-)$ is the variable of interest.

After taking Equation (14) and inputting it into Equation (15), the expectation operator looks like

$$E(-) = \int_{-\infty}^{\infty} (-) \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(X-X_0)^2}{2\sigma^2}} dX \quad (16)$$

Equation (16) yields several cases of interest:

$$E(X) = X_0 \quad (17)$$

$$E(X - X_0) = 0 \quad (18)$$

$$E\left((X - X_0)^2\right) = E(e^2) = \sigma^2 \quad (19)$$

These results imply that: “the expected value of a measurement is the true value”

(Equation (17)), “the average error in the measurement is zero” (Equation (18)), and “the

average squared error is σ^2 ”(Equation (19)) (Wiesel, 2003b:19). The result of Equation (19) plays a key role in least squares estimation.

Principle of Maximum Likelihood.

Suppose that N measurements are taken of an object. As long as each measurement is independent, the probability of obtaining the data set is the product of the individual probability:

$$P(X_1, X_2 \dots X_N) = \frac{1}{(2\pi)^{N/2}} \left[\prod_{i=1}^N \frac{1}{\sigma_i} \right] e^{-\sum_{i=1}^N \frac{(X_i - X_0)^2}{2\sigma_i^2}} \quad (20)$$

Unfortunately, no matter how well the object of interest is measured, x_0 cannot be obtained. Therefore, a different approach must be attempted.

The Principle of Maximum Likelihood is where the estimate $\bar{X}$ is defined as, “the value of X_0 which maximizes the probability of having obtained the actual data set” (Wiesel, 2003b:20). Equation (20) now looks like

$$P(X_1, X_2 \dots X_N) = \frac{1}{(2\pi)^{N/2}} \left[\prod_{i=1}^N \frac{1}{\sigma_i} \right] e^{-\sum_{i=1}^N \frac{(X_i - \bar{X})^2}{2\sigma_i^2}} \quad (21)$$

Subsequently, the true error ($e = X - X_0$) has been exchanged for a residual ($r = X - \bar{X}$).

Equation (21) is maximized when the term within the exponential is minimized:

$$\frac{d}{d\bar{X}} \sum_{i=1}^N \frac{(X_i - \bar{X})^2}{2\sigma_i^2} = 0 \quad (22)$$

which yields

$$\sum_{i=1}^N \frac{(X_i - \bar{X})}{\sigma_i^2} = 0 \quad (23)$$

and simplifies to

$$\bar{X} = \sum_{i=1}^N w_i X_i \quad (24)$$

where

$$w_i = \frac{\frac{1}{\sigma_i^2}}{\sum_{j=1}^N \frac{1}{\sigma_j^2}} \quad (25)$$

The above process of minimizing the exponential is how the Method of Least Squares acquired its name. As stated by Weld (1916:59), “the most probable value of a measured quantity that can be deduced from a series of direct observations, made with equal care and skill, is that for which the sum of the squares of the residuals is a minimum.”

As with any answer that is determined, it is important to know how accurate the answer really is; therefore, the variance of $\bar{X}$ must be found:

$$\sigma_{\bar{X}}^2 = E((\bar{X} - X_0)^2) \quad (26)$$

After inserting Equation (24) into the above equation, rearranging, and simplifying, the result is

$$\sigma_{\bar{X}}^2 = \frac{1}{\sum_{i=1}^N \frac{1}{\sigma_i^2}} \quad (27)$$

Using the result from Equation (27), the estimate from Equation (24) can then be simplified to

$$\bar{X} = \sigma_{\bar{X}}^2 \sum_{i=1}^N \frac{X_i}{\sigma_i^2} \quad (28)$$

Equations (27) and (28) are the two most important equations of the estimation process.

Given data X_i and its standard deviation σ_i , the estimate of the true value and its standard deviation is now obtainable.

Multi-Dimensional Probability.

Thus far, the Gaussian function has been written as the one variable case. In orbit determination, the Gaussian function will be multi-dimensional:

$$f(\bar{X}) = \frac{1}{(2\pi)^{N/2}} |P|^{-1/2} e^{\left(\frac{-1}{2}(\bar{X}-\bar{X}_0)^T P^{-1}(\bar{X}-\bar{X}_0)\right)} \quad (29)$$

where

$$P = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1N} \\ P_{12} & P_{22} & \cdots & P_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ P_{1N} & P_{2N} & \cdots & P_{NN} \end{bmatrix} \quad (30)$$

The covariance matrix (P) is a positive semi-definite matrix. The diagonal terms are the σ^2 quantities and are called the variances. It is the square root of the variance that relates the accuracy of the estimate of the state. The covariance matrix is normally defined as

$$P = E(\bar{e}\bar{e}^T) \quad (31)$$

The above equation looks remarkably similar to Equation (19) from the one variable problem. This is why the diagonal terms of the covariance matrix are defined as σ^2 .

Linearized Dynamics.

Unlike the above process where just one component of an object is estimated, orbit determination is usually composed of several different components. These components; such as position, velocity, and acceleration, are placed into a state vector

($\bar{X}$). Engineers and scientists are interested in how the *state* of a space object changes with time. This can be written as

$$\frac{d\bar{X}}{dt} = g(\bar{X}, t) \quad (32)$$

or

$$\bar{X}(t) = h(\bar{X}(t_0), t) \quad (33)$$

Equation (33) shows the actual solution written in terms of the initial state and time. The state transition matrix (Φ) propagates the actual state as a function of time. If the dynamics can be written as Equation (32), then Φ can be written as

$$\Phi(t, t_0) = \nabla_{x(t_0)} h(\bar{X}(t_0), t) \quad (34)$$

or

$$\Phi(t, t_0) = \frac{\partial \bar{X}(t)}{\partial \bar{X}(t_0)} \quad (35)$$

This approach greatly simplifies the estimation process.

Linear Least Squares.

Linear least squares is an estimation process used on systems where the dynamics can be written as linear differential equations (*i.e.* the function and its derivatives appear only in their first power and are also not multiplied with each other). The observations must also be linear in order to utilize linear least squares. Orbit determination problems are generally nonlinear because they use two-body equations of motion to model the dynamics; and the observations are in the form of range, azimuth, and elevation, which are also nonlinear.

The state of a linear system can be found at any time using the state transition matrix:

$$\bar{X}(t) = \Phi(t, t_0) \bar{X}(t_0) \quad (36)$$

As long as the observations are linear, they can be written using the *observation relation*:

$$\bar{Z}_i(t_i) = H_i \bar{X}(t_i) + \bar{e}_i \quad (37)$$

Substituting Equation (36) into the above equation, and solving for the error yields

$$\bar{e}_i = \bar{Z}_i(t_i) - H_i \Phi(t_i, t_0) \bar{X}(t_0) \quad (38)$$

The term $H_i \Phi(t_i, t_0)$ can be replaced with T_i to yield

$$\bar{e}_i = \bar{Z}_i(t_i) - T_i \bar{X}(t_0) \quad (39)$$

Given that true error can never be determined, the above equation is replaced with the residual:

$$\bar{r} = \bar{Z} - T \bar{X}(t_0) \quad (40)$$

Before proceeding, the following shorthand notation is used to simplify the least squares equations:

$$T \equiv \begin{bmatrix} T_1 \\ T_2 \\ \dots \\ T_N \end{bmatrix} = \begin{bmatrix} H_1 \Phi(t_1, 0) \\ H_2 \Phi(t_2, 0) \\ \dots \\ H_N \Phi(t_N, 0) \end{bmatrix} \quad (41)$$

$$Z \equiv \begin{bmatrix} Z_1 \\ Z_2 \\ \dots \\ Z_N \end{bmatrix} \quad (42)$$

$$Q = \begin{bmatrix} Q_1 & 0 & \dots & 0 \\ 0 & Q_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & Q_N \end{bmatrix} \quad (43)$$

where Q is the instrumental covariance matrix and its inverse is

$$Q^{-1} = \begin{bmatrix} Q_1^{-1} & 0 & \dots & 0 \\ 0 & Q_2^{-1} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & Q_N^{-1} \end{bmatrix} \quad (44)$$

The multi-dimensional probability function resembles

$$f(\bar{X}) = \frac{1}{(2\pi)^{N/2}} |Q|^{-1/2} e^{\left(\frac{-1}{2} (\bar{Z} - T \bar{X}(t_0))^T Q^{-1} (\bar{Z} - T \bar{X}(t_0)) \right)} \quad (45)$$

After minimizing the exponential, the estimated state vector at t_0 can be written as

$$\bar{X}(0) = (T^T Q^{-1} T)^{-1} T^T Q^{-1} \bar{Z} \quad (46)$$

The covariance of the estimate, $P_{\bar{X}} = E(\bar{e}_{\bar{X}} \bar{e}_{\bar{X}}^T)$, goes through several lines of substitution and simplification to arrive in the form

$$P_{\bar{X}} = (T^T Q^{-1} T)^{-1} \quad (47)$$

The state of a system and its covariance can now be estimated given a batch of data and the instrumental covariance.

Summary

The ability to identify and track space-based objects has played a vital role in the United States' space superiority. Scientists and engineers have utilized the concept of Keplerian motion to aid in tracking and propagating space objects; however, as

technology improves and space objects become more complex (*i.e.* increased maneuverability, tethered systems, etc.), the space objects will not always follow Keplerian motion. Many new methods have been designed to track non-Keplerian orbits. Most techniques use non-linear equations of motion for the dynamics and focus on finding the additional acceleration components created by either the tether of a tethered satellite system or the thrust from a maneuvering space object.

Galileo's work on projectile motion lends itself to linear equations of motion that can be adapted to modeling objects in LEO. These equations are very general and do not take into account atmospheric drag; also, gravity is assumed to be constant. When modeling orbits with these equations of motion, the results are only accurate for a short arc of observation data. Linear least squares can be combined with these equations of motion to estimate the state of the system. The linear least squares method has been used for hundreds of years and is a vital part of estimation theory.

III. Methodology

Chapter Overview

The purpose of this chapter is to explain the processes used to create the MATLAB computer model. Section 3.2 explores how to transform the raw radar data into position vectors for use in the estimator. Section 3.3 explains how the truth model was created using Galileo's projectile motion equations. Section 3.4 explains the creation of the linear least squares estimator, which produces the estimate of the state and its covariance.

Raw Data

Radar data is presented in various forms depending on the type of sensor that obtains it. For the purposes of this research, it was assumed that the radar data will consist of range (ρ), azimuth (α), and elevation (β) values. Range is measured in kilometers while azimuth and elevation are measured in radians. These radar values represent the space object's position in the radar site's topocentric frame, where the axes are labeled South, East, and Zenith (SEZ). Figure 4 represents the geometry of the problem. Azimuth is measured clockwise from the north. Elevation is measured upwards from the site's horizon, and range is measured from the radar site to the satellite or space object being observed.

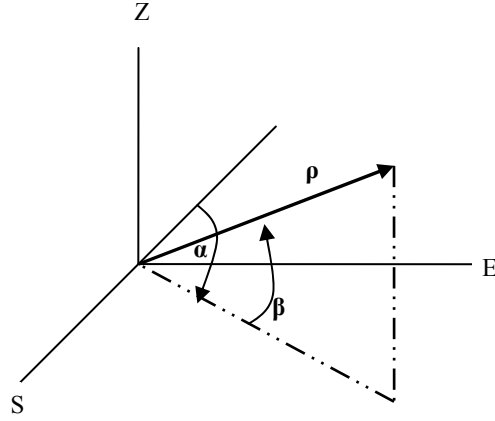


Figure 4. Observation Geometry

The SEZ frame is not inertial; it rotates with the Earth. The raw data, therefore, must be converted from the SEZ coordinate frame to position vectors in the Earth Centered Inertial (IJK) frame, which requires several steps. Figure 5 helps to visualize the problem.

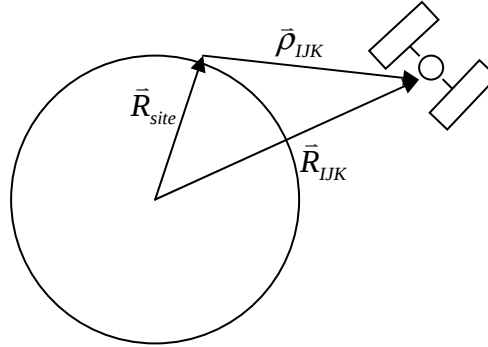


Figure 5. Satellite Position

$\vec{R}_{site}$ represents the position from the center of the Earth to the radar site in IJK coordinates, while $\vec{\rho}_{IJK}$ is the position vector from the radar site to the satellite. The components of $\vec{\rho}$ can be found in the SEZ frame quite easily using Figure 4 and basic trigonometry:

$$\bar{\rho}_{SEZ} = \begin{bmatrix} -\rho \cos(\alpha) \cos(\beta) \\ \rho \sin(\alpha) \cos(\beta) \\ \rho \sin(\beta) \end{bmatrix} \quad (48)$$

$\bar{\rho}$ can then be converted to the IJK frame by multiplying it by the inverse of the rotation matrix D:

$$\bar{\rho}_{IJK} = D^{-1} \bar{\rho}_{SEZ} \quad (49)$$

where

$$D = \begin{bmatrix} \sin(L) \cos(LST) & \sin(L) \sin(LST) & -\cos(L) \\ -\sin(LST) & \cos(LST) & 0 \\ \cos(L) \cos(LST) & \cos(L) \sin(LST) & \sin(L) \end{bmatrix} \quad (50)$$

L represents the site's geodetic latitude and LST represents the local sidereal time. LST is measured from the vernal equinox to the radar site in a counter-clockwise direction.

LST is also the sum of the Greenwich Sidereal Time (GST) and the site's longitude.

Equation (51) is used next to calculate the position from the center of the Earth to the radar site:

$$\bar{R}_{site} = \begin{bmatrix} x \cos(LST) \\ x \sin(LST) \\ z \end{bmatrix} \quad (51)$$

The quantities x and z take into account the fact that Earth is not a perfect sphere, but rather an ellipsoid:

$$x = \left[\frac{R_{\oplus}}{\sqrt{1 - e_{\oplus}^2 \sin^2 L}} + H \right] \cos L \quad (52)$$

$$z = \left[\frac{R_{\oplus}(1-e_{\oplus}^2)}{\sqrt{1-e_{\oplus}^2 \sin^2 L}} + H \right] \sin L \quad (53)$$

With $\bar{R}_{site}$ and the values of $\bar{\rho}_{IJK}$ for the entire track in hand, the position of the satellite in IJK coordinates is obtained for each observation. These values are directly inserted into the estimator as the observation data $(\bar{Z}_i)$ to create an initial state vector $\bar{X}(t_0)$.

Truth Model

The truth model is basically a tool used to create simulated data and an aid for verifying the estimator. This model creates data in a format that a typical radar site would expect: range, azimuth, and elevation values. Due to the specifics of this research, the truth model must be able to simulate satellite motion with or without an extra acceleration component. The truth model must also simulate real-world factors such as noise. This section explains how all of these requirements were addressed, as well as how the truth model is executed and validated. This section is divided into three sub-sections: Program Execution, Equations, and Program Validation.

Program Execution.

The truth model takes an initial input for $\bar{r}$, $\dot{\bar{r}}$, $\ddot{\bar{r}}$, and $\dddot{\bar{r}}$ in units of km, km/s, km/s², km/s³ respectively. These inputs are used in the equations of motion to create a matrix of position and velocity vectors for a five-minute radar track for the radar site located at Ascension, Atlantic (another site can be specified at the beginning of the program). Matrices for range, azimuth, and elevation are created from the position data for use in the estimator. The model places the epoch time at the middle of the trajectory

as opposed to the beginning. Output from the truth model consists of range (kilometers), azimuth (radians), elevation (radians), time (in Julian days) for the entire five-minute track, plus the site's latitude and noise statistics in range, azimuth, and elevation.

The truth model simulation used a start time of 13 Sep 2007, 12:00:00 UTC (Coordinated Universal Time) with an epoch of 12:02:30 UTC. Noise was added to the range, azimuth, and elevation data. Data from the truth model was output to a file for later use in the estimator.

Equations.

The first step to simulating data and creating a truth model was to define the dynamics. Satellite motion in this case was not modeled using two-body motion, but was instead estimated using Galileo's expertise on modeling projectile trajectories with a Taylor series expansion. The equations of motion below represent a 3rd order Taylor series approximation:

$$\vec{r} = \vec{r}_0 + \vec{v}_0\Delta t + \frac{1}{2}(\vec{A}_0 + \vec{g}_0)\Delta t^2 + \frac{1}{6}(\dot{\vec{A}}_0 + \dot{\vec{g}}_0)\Delta t^3 \quad (54)$$

$$\vec{v} = \vec{v}_0 + (\vec{A}_0 + \vec{g}_0)\Delta t + \frac{1}{2}(\dot{\vec{A}}_0 + \dot{\vec{g}}_0)\Delta t^2 \quad (55)$$

$$\vec{a} = (\vec{A}_0 + \vec{g}_0) + (\dot{\vec{A}}_0 + \dot{\vec{g}}_0)\Delta t \quad (56)$$

$$\dot{\vec{a}} = (\dot{\vec{A}}_0 + \dot{\vec{g}}_0) \quad (57)$$

A discussion on why the 3rd order Taylor series approximation was used for the truth model, and not some other order, is saved for the Analysis and Results chapter. In brief, the 3rd order equations were more accurate than the 2nd order equations, and had results sufficient enough to not need the 4th order equations.

$\bar{A}_0$ is the non-gravitational acceleration vector that is to be detected. This component is assumed to be constant. The g vector in this case is not 9.81 m/s^2 ; it is the negative gradient of the geopotential:

$$-\nabla V_{\text{geopotential}} = \begin{bmatrix} \frac{\partial V}{\partial x} \\ \frac{\partial V}{\partial y} \\ \frac{\partial V}{\partial z} \end{bmatrix} = \bar{g} \quad (58)$$

where

$$V_{\text{geopotential}} = \frac{-\mu}{\sqrt{x^2 + y^2 + z^2}} + \frac{\mu R_{\oplus}^2 J_2}{2(x^2 + y^2 + z^2)^{3/2}} \left(\frac{3z^2}{(x^2 + y^2 + z^2)} - 1 \right) \quad (59)$$

The first term of the geopotential represents the two-body problem and the second term represents the J_2 effects (Wiesel, 2003a:144). The variables x , y , and z are the Cartesian coordinates of the initial value of $\bar{r}$, which can also be written as $\bar{R}_{IJK}$. An expanded view of the partial derivatives can be seen in Equations (60)-(62).

$$\frac{\partial V}{\partial x} = -\frac{\mu x}{(x^2 + y^2 + z^2)^{3/2}} - \frac{3x\mu J_2 R_{\oplus}^2}{2(x^2 + y^2 + z^2)^{5/2}} + \frac{15xz^2\mu J_2 R_{\oplus}^2}{2(x^2 + y^2 + z^2)^{7/2}} \quad (60)$$

$$\frac{\partial V}{\partial y} = -\frac{\mu y}{(x^2 + y^2 + z^2)^{3/2}} - \frac{3y\mu J_2 R_{\oplus}^2}{2(x^2 + y^2 + z^2)^{5/2}} + \frac{15yz^2\mu J_2 R_{\oplus}^2}{2(x^2 + y^2 + z^2)^{7/2}} \quad (61)$$

$$\frac{\partial V}{\partial z} = -\frac{\mu z}{(x^2 + y^2 + z^2)^{3/2}} - \frac{9z\mu J_2 R_{\oplus}^2}{2(x^2 + y^2 + z^2)^{5/2}} + \frac{15z^3\mu J_2 R_{\oplus}^2}{2(x^2 + y^2 + z^2)^{7/2}} \quad (62)$$

The above equations can be divided into the $\bar{g}_{2\text{-body}}$ (Equation (63)) and the $\bar{g}_{J_2}$ (Equation (64)) components.

$$\bar{g}_{2-body} = \begin{bmatrix} \frac{-\mu x}{r^3} \\ \frac{-\mu y}{r^3} \\ \frac{-\mu z}{r^3} \end{bmatrix} \quad (63)$$

$$\bar{g}_{J_2} = \frac{-\mu J_2 R_\oplus^2}{2} \begin{bmatrix} \frac{3x}{r^5} - \frac{15xz^2}{r^7} \\ \frac{3y}{r^5} - \frac{15yz^2}{r^7} \\ \frac{9z}{r^5} - \frac{15z^3}{r^7} \end{bmatrix} \quad (64)$$

Since the equations of motion require the derivative of $\bar{g}$, it was necessary to differentiate. Keeping in mind that x , y , z and r are all functions of time, the following derivatives for $\bar{g}_{2-body}$ and the $\bar{g}_{J_2}$ were obtained:

$$\dot{\bar{g}}_{J_2} = \frac{-\mu J_2 R_\oplus^2}{2} \begin{bmatrix} \frac{-15x(\bar{r} \bullet \bar{v})}{r^7} + \frac{3\dot{x}}{r^5} + \frac{105xz^2(\bar{r} \bullet \bar{v})}{r^9} - \frac{30xzz\dot{z}}{r^7} - \frac{15\dot{x}z^2}{r^7} \\ \frac{-15y(\bar{r} \bullet \bar{v})}{r^7} + \frac{3\dot{y}}{r^5} + \frac{105yz^2(\bar{r} \bullet \bar{v})}{r^9} - \frac{30yzz\dot{z}}{r^7} - \frac{15\dot{y}z^2}{r^7} \\ \frac{-45z(\bar{r} \bullet \bar{v})}{r^7} + \frac{9\dot{z}}{r^5} + \frac{105z^3(\bar{r} \bullet \bar{v})}{r^9} - \frac{45\dot{z}z^2}{r^7} \end{bmatrix} \quad (65)$$

$$\dot{\bar{g}}_{2-body} = \begin{bmatrix} \frac{3\mu x(\bar{r} \bullet \bar{v})}{r^5} - \frac{\mu \dot{x}}{r^3} \\ \frac{3\mu y(\bar{r} \bullet \bar{v})}{r^5} - \frac{\mu \dot{y}}{r^3} \\ \frac{3\mu z(\bar{r} \bullet \bar{v})}{r^5} - \frac{\mu \dot{z}}{r^3} \end{bmatrix} \quad (66)$$

The variables $\dot{x}$, $\dot{y}$, and $\dot{z}$ are the individual components of the initial value for velocity and are written as u , v , and w in the MATLAB code. The term $\bar{r} \bullet \bar{v}$ can be written as $xu+yv+zw$. The value of $\bar{g}$ is the local gravity vector for the given initial

position and velocity. Since the values for $x, y, z, \dot{x}, \dot{y},$ and $\dot{z}$ are taken from only the initial position and velocity, the value of both $\bar{g}$ and $\dot{\bar{g}}$ is constant for the five-minute radar track.

After using the equations of motion to find the position vectors (in IJK), the data was transformed to range, azimuth and elevation values (in SEZ) because this is the format of the data that radar sites would receive. The values for range, azimuth, and elevation change depending on the location of the radar site. The radar site represented in the truth model is Ascension, Atlantic. Table 1 lists the specifics for this site.

Table 1. Ascension Atlantic Radar Specifics (Vallado, 2001:242)

Location			Noise		
Latitude (°)	Longitude (°)	Altitude (m)	Range (m)	Azimuth (°)	Elevation (°)
-7.91	-14.40	56.1	101.7	0.0248	0.0283

The site location was used to determine $\bar{R}_{site}$ (Equations (51)-(53)). The noise values in Table 1 were saved for use in the estimator. Figure 5 shows how $\bar{\rho}_{IJK}$ was found given $\bar{R}_{site}$ and $\bar{R}_{IJK}$. The conversion from $\bar{\rho}_{IJK}$ to $\bar{\rho}_{SEZ}$ was obtained by using Equation (49). The values of range, azimuth, and elevation for each observation were found using

$$\rho = \sqrt{\rho_s^2 + \rho_e^2 + \rho_z^2} \quad (67)$$

$$\alpha = \tan^{-1} \frac{\rho_e}{-\rho_s} \quad (68)$$

$$\beta = \sin^{-1} \frac{\rho_z}{\rho} \quad (69)$$

These radar values for the entire time span are saved in matrices.

Radar sites will never receive perfect data; therefore, the truth model adds noise to the range, azimuth, and elevation values to simulate real world results. MATLAB's random number generator (*randn*) was used to simulate noise. *Randn* is a pseudo-random function that creates noise with a normal (Gaussian) distribution. The mean and standard deviation of the noise can be specified when using *randn*. Range, azimuth, and elevation each have a noise matrix associated with it. The mean for these noise matrices is zero.

Program Validation.

The position and velocity matrices obtained from the equations of motion for a given initial position, velocity, and for both zero acceleration and its derivative were compared to the same initial inputs in a Satellite Tool Kit (STK) simulation. STK allows the user to estimate an orbit using various propagation methods such as two-body, J_2 , J_4 etc. Since the intent was to verify how close the MATLAB model was to reality, the High Precision Orbit Propagator (HPOP) was chosen to propagate the trajectory. This propagator takes into account many elements such as: central body gravity, solar radiation pressure, third body gravity (sun, moon) and drag, plus uses an RKF 7(8) integrator. The initial conditions for the simulated orbit were written in Cartesian coordinates with position in kilometers and velocity in km/s:

$$\vec{r}_0 = \begin{bmatrix} -6079.6 \\ 1837.9 \\ -1596.6 \end{bmatrix} \quad (70)$$

$$\bar{\mathbf{v}}_0 = \begin{bmatrix} -2.96 \\ -5.65 \\ 4.82 \end{bmatrix} \quad (71)$$

The magnitude of the position is roughly 6549 km. Taking into account that the radius of the Earth is approximately 6738 km, the simulated orbit has an altitude of only 171 km. This altitude is a very low LEO and most objects do not last long at such a low altitude due to atmospheric drag. However, this altitude could be quite realistic for a tethered satellite or ballistic missile.

The results for position and velocity between STK and MATLAB were very close, but changed in accuracy depending on which order of the Taylor series approximation was used. A visual comparison of the accuracy of the 2nd, 3rd, and 4th order approximations can be seen in Chapter IV. Table 2 shows the root mean square (RMS) error for the x , y , z , $\dot{x}$, $\dot{y}$, and $\dot{z}$ components of position and velocity.

Table 2. STK vs. MATLAB RMS Error (3rd Order EOM)

Position (km)			Velocity (km/s)		
x	y	z	$\dot{x}$	$\dot{y}$	$\dot{z}$
0.096835	0.034235	0.029878	0.002951	0.00099	0.000866

The values in Table 2 were calculated using Equation (72):

$$RMS = \sqrt{\frac{1}{N} \sum_{i=1}^N e_i^2} \quad (72)$$

The component e_i is the individual magnitudes of error between the MATLAB and STK values and N is the number of data points. In this case, $N=300$, one for each second during the five-minute track.

STK was also used to validate the local gravity vector by finding the satellite's acceleration component at epoch in the STK simulation. Given the above initial conditions, the local gravity vector calculated by the truth model is

$$\bar{g} = \begin{bmatrix} 0.00863715 \\ -0.0026111 \\ 0.00227524 \end{bmatrix} \text{ km/s}^2 \quad (73)$$

This is extremely close to STK's acceleration value at epoch:

$$\bar{g} = \begin{bmatrix} 0.008638 \\ -0.002610 \\ 0.002275 \end{bmatrix} \text{ km/s}^2 \quad (74)$$

The range, azimuth, and elevation portion of the code was validated with the STK simulation as well as with code from the previous section.

Estimator

Execution.

The estimator reads in an input file that contains: elevation, range, azimuth, time (in Julian days), site latitude, and noise statistics for range, azimuth, and elevation. The truth model used units of kilometers, radians and Julian days for the various components of output data; however, radar sites typically get their data in units of meters, degrees, and time (in year, day number, hour, minutes, and seconds). Therefore, the estimator has an option where the radar data can be converted into the kilometers, radians, and Julian

day format. The data is then converted to position vectors using the process outlined in Section 3.2. The estimator uses linear least squares to determine the estimate of the state vector at epoch to include: $\bar{r}_0$, $\bar{v}_0$, $\bar{a}_0$, and $\dot{\bar{a}}_0$. The estimator also produces the covariance of the state vector. Since this is a linear system, there is no need to compute residuals and iterate. There is also no need for an initial guess of the state vector.

Equations.

The state vector for this problem was defined as

$$\bar{X} = \begin{bmatrix} \bar{r} \\ \bar{v} \\ \bar{a} \\ \dot{\bar{a}} \end{bmatrix} \quad (75)$$

The initial state vector looks like

$$\bar{X}_0 = \begin{bmatrix} \bar{r}_0 \\ \bar{v}_0 \\ (\bar{A}_0 + \bar{g}_0) \\ (\dot{\bar{A}}_0 + \dot{\bar{g}}_0) \end{bmatrix} = \begin{bmatrix} \bar{r}_0 \\ \bar{v}_0 \\ \bar{a}_0 \\ \dot{\bar{a}}_0 \end{bmatrix} \quad (76)$$

With this definition of the state vector and the dynamics of the system already known, a closed form solution of the state transition matrix was obtained using Equation (35) from Chapter II:

$$\Phi(t, t_0) = \begin{bmatrix} \frac{\partial \bar{r}}{\partial \bar{r}_0} & \frac{\partial \bar{r}}{\partial \bar{v}_0} & \frac{\partial \bar{r}}{\partial \bar{a}_0} & \frac{\partial \bar{r}}{\partial \dot{\bar{a}}_0} \\ \frac{\partial \bar{v}}{\partial \bar{r}_0} & \frac{\partial \bar{v}}{\partial \bar{v}_0} & \frac{\partial \bar{v}}{\partial \bar{a}_0} & \frac{\partial \bar{v}}{\partial \dot{\bar{a}}_0} \\ \frac{\partial \bar{a}}{\partial \bar{r}_0} & \frac{\partial \bar{a}}{\partial \bar{v}_0} & \frac{\partial \bar{a}}{\partial \bar{a}_0} & \frac{\partial \bar{a}}{\partial \dot{\bar{a}}_0} \\ \frac{\partial \dot{\bar{a}}}{\partial \bar{r}_0} & \frac{\partial \dot{\bar{a}}}{\partial \bar{v}_0} & \frac{\partial \dot{\bar{a}}}{\partial \bar{a}_0} & \frac{\partial \dot{\bar{a}}}{\partial \dot{\bar{a}}_0} \end{bmatrix} \quad (77)$$

Each component is a 3x3 matrix, which makes the state transition matrix a square 12x12 matrix. The diagonal terms are the identity matrix. All terms to the left of the diagonal are the null matrix. The terms $\frac{\partial \bar{r}}{\partial \bar{v}_0}$, $\frac{\partial \bar{v}}{\partial \bar{a}_0}$ and $\frac{\partial \bar{a}}{\partial \dot{\bar{a}}_0}$ are the identity matrix multiplied by time. The terms $\frac{\partial \bar{r}}{\partial \bar{a}_0}$ and $\frac{\partial \bar{v}}{\partial \dot{\bar{a}}_0}$ are the identity times $\frac{t^2}{2}$, while $\frac{\partial \bar{r}}{\partial \dot{\bar{a}}_0}$ is the identity times $\frac{t^3}{6}$. A multi-dimensional array for the state transition matrix, with a one-second time step, within the five-minute track is created.

In the quest to simplify the observation relation G and to linearize the data, the observed data vector consisting of range, azimuth, and elevation components is converted to pseudo-data (Wiesel, 2003b:94-95):

$$\begin{aligned} \bar{Z}' &= \bar{r} \\ \bar{Z}' &= G(\bar{X}, t) = (I, \varphi, \varphi, \varphi) \bar{X} \end{aligned} \quad (78)$$

In this equation, I is the identity matrix and φ is the null matrix. This creates a simplified G function with its linearization H also simplified and not a function of time:

$$H = \frac{\partial G}{\partial \bar{X}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (79)$$

After H is obtained, the observation matrix (T) can be obtained (see Equation (39)).

The use of pseudo-data instead of range, azimuth, and elevation creates a Q matrix that is no longer constant (Wiesel, 2003b:95). Therefore, the values of Q must be calculated for every position. Wiesel shows that these values are easily obtained through a simple rotation:

$$Q' = JQJ^T \quad (80)$$

with the original covariance written as

$$Q = \begin{bmatrix} \rho_{error}^2 & 0 & 0 \\ 0 & \alpha_{error}^2 & 0 \\ 0 & 0 & \beta_{error}^2 \end{bmatrix} \quad (81)$$

The values of ρ_{error} , α_{error} , and β_{error} are obtained from Table 1.

The Jacobian is obtained by

$$J = D^{-1}K \quad (82)$$

where

$$K = \begin{bmatrix} -\cos \beta \cos \alpha & \rho \cos \beta \sin \alpha & \rho \sin \beta \cos \alpha \\ \cos \beta \sin \alpha & \rho \cos \beta \cos \alpha & -\rho \sin \beta \sin \alpha \\ \sin \beta & 0 & \rho \cos \beta \end{bmatrix} \quad (83)$$

and D^{-1} is the inverse of Equation (50).

All of the necessary matrices needed to find the state vector at epoch and its covariance have been found. Since least squares is a batch process, the least squares

equations obtained in Chapter II, Equations (27) and (28), can be reformatted and written as

$$P_{\bar{X}} = \left(\sum_{i=1}^N T_i^T Q_i^{-1} T_i \right)^{-1} \quad (84)$$

$$\bar{X}(t_0) = P_{\bar{X}} \sum_{i=1}^N T_i^T Q_i^{-1} \bar{Z}_i \quad (85)$$

in order to save computer space and ensure a quicker processing time. In this case, the product of the covariance and state vector are summed for a five-minute radar track from $t = -150$ to $t = 149$ seconds. After the state vector at epoch is obtained, any state vector thereafter is obtained simply by using Equation (36).

Validation.

The truth model was used to validate the estimator. Various initial positions and velocities without non-gravitational acceleration were run through the truth model and the estimator to verify that the estimated state was within 1σ of the true state. The estimated state and its covariance were also validated using the Monte Carlo method. The Monte Carlo method was used to ensure that the covariance was representative of the uncertainty in the data. An in-depth look at the Monte Carlo method and its results is included in Chapter IV.

Summary

Numerous MATLAB routines were created in order to develop the truth model (Appendix B) and least squares estimator (Appendix C). Given an initial position and velocity, the truth model develops a five-minute track of data and converts it to the familiar range, azimuth, and elevation format. The least squares estimator takes both the

radar data and instrument covariance in order to estimate the epoch value of the state vector (to include $\bar{r}$, $\dot{\bar{r}}$, $\ddot{\bar{r}}$, and $\ddot{\bar{r}}$) and the covariance of the state. A flowchart that depicts this process, from the initial input data to the final estimate of the state and its covariance, is located in Appendix D. The magnitude of the additional acceleration and its covariance were the components of interest in this research. A more thorough analysis of the acceleration and its covariance is discussed in the next chapter.

IV. Analysis and Results

Chapter Overview

The purpose of this chapter is to analyze the various sets of equations of motion used in the truth model to model a space object in LEO; and to determine if one of these sets of equations of motion combined with a linear least squares estimator can satisfactorily detect non-gravitational acceleration with statistical accuracy. Section 4.2 explores the accuracy of the various sets of equations of motion compared to an STK simulation of the same orbit. Sections 4.3 and 4.4 present several test cases with various initial conditions used to analyze how well the truth model and estimator function. These test cases help to determine the degree of non-gravitational acceleration that can be adequately detected. Section 4.5 addresses the investigative questions that were posed in Chapter I. Section 4.6 summarizes the main discoveries of the research.

STK Simulation vs. MATLAB Model

The initial conditions stated in Equations (70) and (71) were used to create a STK simulation for use as a baseline model. The 2nd, 3rd, and 4th order Taylor series approximations were compared to the STK simulation to determine which equations modeled a space object accurately enough. The same initial conditions were input into the truth model. For this analysis, both the non-gravitational acceleration and its derivative were assumed to be zero. After obtaining the position and velocity values from the truth model for the five-minute radar track, these values were compared to the STK values. The amount of error between the STK values and MATLAB values was calculated using

$$\text{error} = \text{observed} - \text{expected} \Rightarrow \text{MATLAB} - \text{STK} \quad (86)$$

The values of error for both position and velocity were graphed using Excel in order to get a visual idea of the accuracy of the equations. The following sub-sections go into detail about the accuracy of the various orders of the equations of motion.

2nd Order.

The 2nd order equations of motion were the initial equations tested for the truth model. Figure 6 shows the relative error in position compared to the STK simulation. As would be expected, the error is minimal around epoch but grows as time moves on. The amount of error at these other times is not particularly ideal.

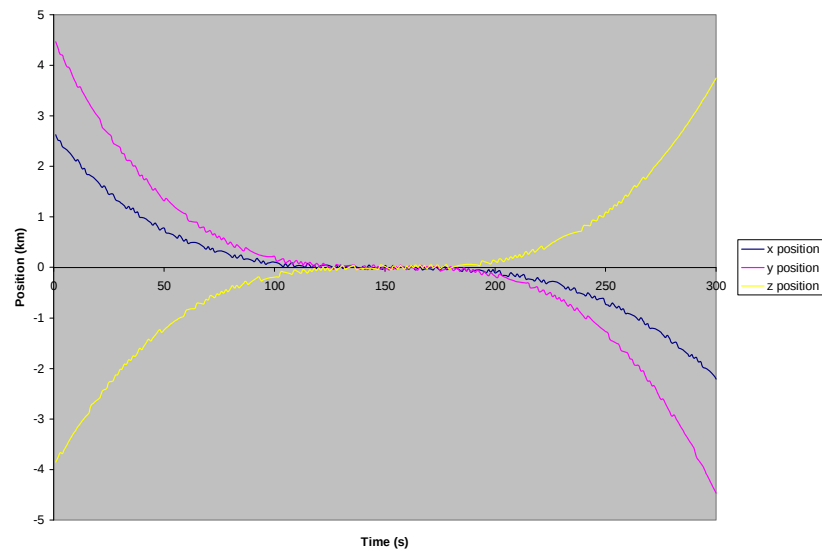


Figure 6. STK vs. MATLAB Position Error (2nd Order)

Figure 7 shows the error in velocity compared to the STK simulation. As seen in Figure 6, the error is zero at epoch but grows significantly as time moves away. Looking at the z velocity line, the greatest magnitude in error is at t=0 seconds where velocity is

roughly 0.08 km/s or 80 m/s. The error magnitude, 0.53 m/s^2 , was obtained by dividing the velocity by half the observation time

$$\frac{80 \text{ m/s}}{150 \text{ s}} = 0.53 \text{ m/s}^2 \quad (87)$$

This value is potentially the amount of acceleration that could go unnoticed due to the level of error in the equations of motion. The magnitude of error in this model is too high for the 2nd order equations of motion to be of any use. The model must be able to detect accelerations in the cm/s^2 or possibly even $\mu\text{m/s}^2$ range; therefore, the equations of motion must be expanded out to obtain better accuracy.

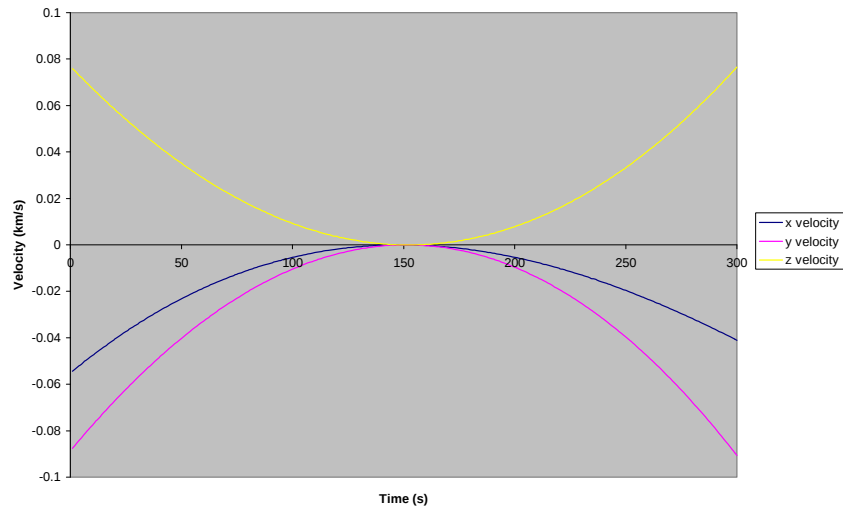


Figure 7. STK vs. MATLAB Velocity Error (2nd Order)

3rd Order.

The equations of motion were expanded out to the 3rd order to obtain a higher degree of accuracy. Figure 8 shows the amount of position error for the 3rd order equations of motion. As was seen in the previous graphs, the amount of error at epoch is minimal and then grows as time increases. Unlike the previous graphs, however, the

magnitude of error is much less. The greatest magnitude of position error in Figure 6 is roughly 4.5 km; whereas, the greatest error in Figure 8 is only 0.3 km. This is a 15 times improvement.

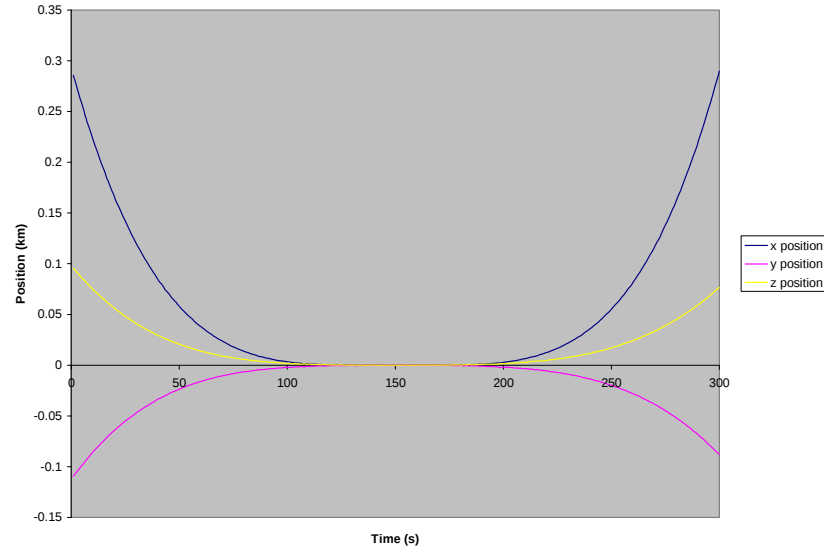


Figure 8. STK vs. MATLAB Position Error (3rd Order)

So far, the use of the 3rd order equations of motion seems to be yielding better results.

Figure 9 shows the velocity error for the 3rd order equations of motion. These results are also much better than that of the 2nd order. The greatest magnitude of error at t=0 is 0.008 km/s. Dividing this value by 150 s yields

$$\frac{8 \text{ m/s}}{150 \text{ s}} = 0.053 \text{ m/s}^2 \quad (88)$$

Therefore, the amount of undetected acceleration that could be present in the 3rd order equations of motion is 0.053 m/s². This result is 10 times better than the 2nd order equations of motion. If the 3rd order yielded much better results, it seems only reasonable that expanding to the 4th order would obtain an even higher level of detection.

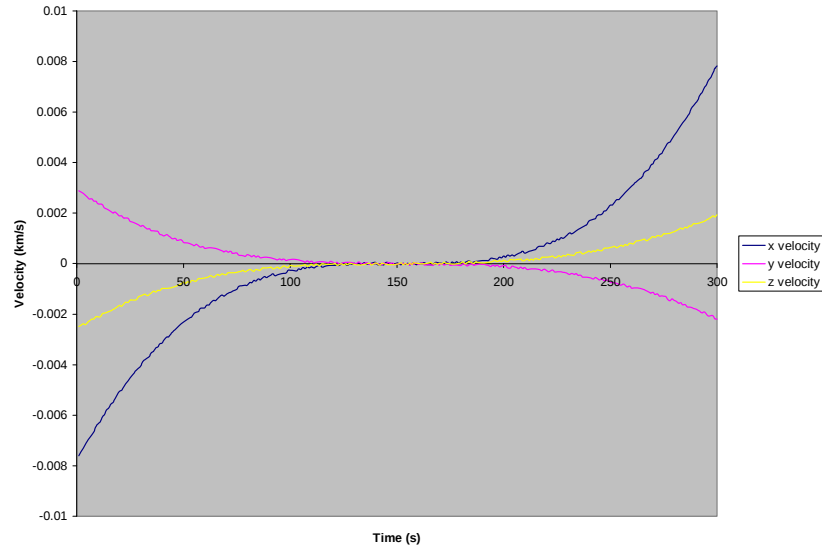


Figure 9. STK vs. MATLAB Velocity Error (3rd Order)

4th Order.

The 4th order equations of motion yielded graphs that were a bit different than those seen in the previous figures. Figure 10 represents the position error for the 4th order expansion. Just like the previous graphs, there is zero error at epoch; but instead of growing exponentially thereafter, the graph curves again at t=250 seconds. The amount of error present is also significantly less than that of the previous graphs.

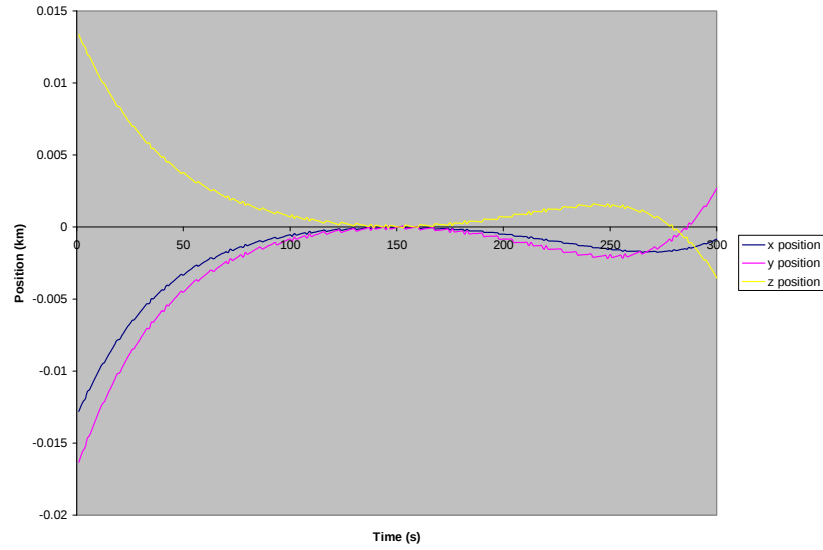


Figure 10. STK vs. MATLAB Position Error (4th Order)

The velocity graph seen in Figure 11 also yielded much better results. The maximum error at $t=0$ seconds is roughly 0.0004 km/s^2 . Following the same process seen in Equations (87) or (88), the amount of undetected acceleration is 0.00267 m/s^2 . The level of detection is roughly 20 times better than what was seen using the 3rd order equations.

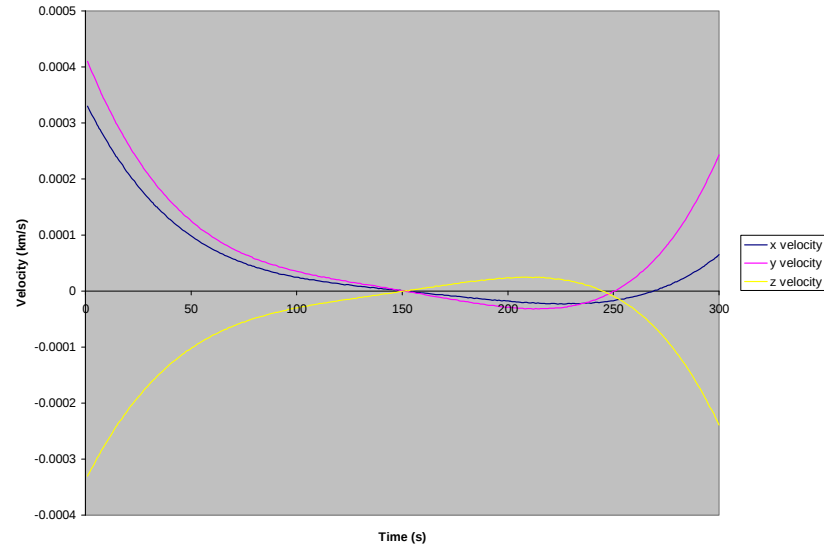


Figure 11. STK vs. MATLAB Velocity Error (4th Order)

Besides the visual comparison of the accuracy of the different Taylor series expansions, there is the numerical approach. The RMS error for all components in position and velocity was calculated using Equation (72) for the 2nd, 3rd, and 4th order equations of motion. Table 3 lists the results. As is expected, the RMS error decreases as the order used increases. The 4th order may yield the best results; however, the 3rd order results are also quite viable compared to the fairly inadequate results seen from the 2nd order.

Table 3. STK vs. MATLAB RMS Error

	Position (km)			Velocity (km/s)		
	x	y	z	$\dot{x}$	$\dot{y}$	$\dot{z}$
2 nd Order	0.91264	1.69929	1.45603	0.02163	0.04016	0.03436
3 rd Order	0.096835	0.034235	0.029878	0.002951	0.00099	0.000866
4 th Order	0.003297	0.004272	0.003524	8.88E-05	0.000121	0.000102

The question is, “Which order of equations of motion is accurate enough for the purposes of this research?” The position and velocity data obtained from the truth model for the 3rd order equations of motion was processed through the least squares estimator to obtain the state vector and its covariance. The 4th order data was also processed through the estimator. The 4th order equations of motion produced a 15x1 state vector while the 3rd order equations of motion produced a 12x1 state vector, as seen in Table 4.

Table 4. MATLAB Estimated State Vector with No Noise

Variable	4th Order	3rd Order
x (km)	-6079.6	-6079.6
y (km)	1837.9	1837.9
z (km)	-1596.6	-1596.6
$\dot{x}$ (km/s)	-2.96	-2.96
$\dot{y}$ (km/s)	-5.65	-5.65
$\dot{z}$ (km/s)	4.82	4.82
a_x (km/s ²)	0.008637157	0.008637157
a_y (km/s ²)	-0.002611065	-0.002611065
a_z (km/s ²)	0.002275235	0.002275235
$\dot{a}_x$ (km/s ³)	4.2799E-06	4.2799E-06
$\dot{a}_y$ (km/s ³)	8.00425E-06	8.00425E-06
$\dot{a}_z$ (km/s ³)	-6.84906E-06	-6.84906E-06
$\ddot{a}_x$ (km/s ⁴)	-1.41524E-08	
$\ddot{a}_y$ (km/s ⁴)	4.41528E-09	
$\ddot{a}_z$ (km/s ⁴)	-3.90375E-09	

The estimates of the state vector for both the 3rd and 4th order equations of motion in a noiseless scenario were nearly identical. Differences were seen, however, in their covariance. For a given covariance matrix, the top left diagonal term represents the variance of the x position term. The second term is the variance of the y position and so forth all the way down to the bottom right diagonal term, which in the 3rd order case, is the last component of $\ddot{a}$. Table 5 lists the variances obtained for the noiseless state.

Table 5. MATLAB Variance for Noiseless Scenario

Variable	4th Order	3rd Order
x (km)	0.002266	0.001569
y (km)	0.001178	0.000944
z (km)	0.000549	0.000482
$\dot{x}$ (km/s)	7.1E-07	6.7E-07
$\dot{y}$ (km/s)	4.02E-07	2.55E-07
$\dot{z}$ (km/s)	2.32E-07	1.27E-07
a_x (km/s ²)	8.98E-10	8.99E-11
a_y (km/s ²)	3.27E-10	4.89E-11
a_z (km/s ²)	1.64E-10	3.18E-11
$\dot{a}_x$ (km/s ³)	1.38E-13	1.3E-13
$\dot{a}_y$ (km/s ³)	7.51E-14	4.63E-14
$\dot{a}_z$ (km/s ³)	4.9E-14	2.77E-14
$\ddot{a}_x$ (km/s ⁴)	3.63E-16	
$\ddot{a}_y$ (km/s ⁴)	1.29E-16	
$\ddot{a}_z$ (km/s ⁴)	7.7E-17	

The results from Tables 4 and 5 use data from a noiseless environment, which, unfortunately, does not accurately portray reality. Gaussian noise was added to the position and velocity data for both the 3rd and 4th order simulations to obtain new estimates of the state and covariance. Table 6 lists the estimates of the states. As expected, these new values are slightly different than the values with no noise.

Table 6. MATLAB Estimated State Vector with Noise

Variable	4 th Order	3 rd Order
x (km)	-6079.585554	-6079.5851
y (km)	1837.895358	1837.8787
z (km)	-1596.596015	-1596.5725
$\dot{x}$ (km/s)	-2.959402644	-2.9605417
$\dot{y}$ (km/s)	-5.650467507	-5.6498852
$\dot{z}$ (km/s)	4.819993529	4.8202066
a_x (km/s ²)	0.008611162	0.0086256
a_y (km/s ²)	-0.002594883	-0.0026044
a_z (km/s ²)	0.00226863	0.0022762
$\dot{a}_x$ (km/s ³)	4.07297E-06	4.513E-06
$\dot{a}_y$ (km/s ³)	8.18258E-06	7.943E-06
$\dot{a}_z$ (km/s ³)	-6.85554E-06	-6.763E-06
$\ddot{a}_x$ (km/s ⁴)	2.76991E-09	
$\ddot{a}_y$ (km/s ⁴)	-6.81815E-09	
$\ddot{a}_z$ (km/s ⁴)	1.0762E-09	

As with any estimator, it is necessary to ensure that the results are unbiased and that the covariance matrix accurately reflects the amount of uncertainty in the data. Thus far, the results of the covariance matrix have not been validated. A technique called Monte Carlo analysis is often used to validate the function of the estimator.

Monte Carlo Analysis.

There are several steps required in order to use the Monte Carlo method. For a given trajectory, the truth model and estimator must produce N number of data sets. Each data set has different noise, but with the same mean and standard deviation. Ultimately, this produces slightly different estimates of the state vector. These N estimates are used to confirm that the estimator is, “ i) on the average unbiased, ii) that the average estimate is the true value, and iii) that the output covariance is actually

representative of the uncertainty in the estimate” (Wiesel, 2003b:138). As long as the estimator is unbiased, the true state should be obtainable using Equation (89):

$$\bar{X}_0 \approx \frac{1}{N} \sum_{i=1}^N \bar{X}_i \quad (89)$$

Unlike in reality, the true state ($\bar{X}_0$) is known because it was chosen in the truth model.

The variable $\bar{X}_i$ is the different estimates of the state. Using this same method, the covariance of the state should be

$$P \approx \frac{1}{N} \sum_{i=1}^N (\bar{X}_i - \bar{X}_0)(\bar{X}_i - \bar{X}_0)^T \quad (90)$$

Up to a certain point, increasing the number of data sets yields more accurate results.

Ten data sets were obtained for the test case. Table 7 displays data sets 1-5 where the

results are listed in the order $\bar{X}^T = [\bar{r} \quad \bar{v} \quad \bar{a} \quad \dot{\bar{a}} \quad \ddot{\bar{a}}]$.

Table 7. State Vector Data Sets 1-5 (4th Order)

	Set 1	Set 2	Set 3	Set 4	Set 5
x (km)	-6079.607723	-6079.669544	-6079.627863	-6079.585554	-6079.588185
y (km)	1837.866979	1837.957821	1837.940816	1837.895358	1837.894576
z (km)	-1596.559154	-1596.628007	-1596.627217	-1596.596015	-1596.598784
$\dot{x}$ (km/s)	-2.960495055	-2.957693797	-2.960761762	-2.959402644	-2.959472693
$\dot{y}$ (km/s)	-5.650189915	-5.650678485	-5.648872485	-5.650467507	-5.650354919
$\dot{z}$ (km/s)	4.820082613	4.819354213	4.81885531	4.819993529	4.820127364
a_x (km/s ²)	0.008648758	0.008666269	0.008630577	0.008611162	0.008616777
a_y (km/s ²)	-0.00259395	-0.002621308	-0.00263258	-0.002594883	-0.002595558
a_z (km/s ²)	0.002266441	0.002273629	0.002259587	0.00226863	0.002270821
$\dot{a}_x$ (km/s ³)	4.47891E-06	3.49766E-06	4.72945E-06	4.07297E-06	3.92502E-06
$\dot{a}_y$ (km/s ³)	8.08021E-06	8.33767E-06	7.45924E-06	8.18258E-06	7.99297E-06
$\dot{a}_z$ (km/s ³)	-6.70901E-06	-6.74418E-06	-6.40824E-06	-6.85554E-06	-6.90807E-06
$\ddot{a}_x$ (km/s ⁴)	-2.94E-08	-3.25441E-08	-1.77783E-09	2.76991E-09	4.34767E-09
$\ddot{a}_y$ (km/s ⁴)	-2.08876E-09	5.79647E-09	1.59187E-08	-6.81815E-09	-1.81219E-09
$\ddot{a}_z$ (km/s ⁴)	1.59348E-09	-4.27323E-09	1.14212E-08	1.0762E-09	-4.33308E-10

Plugging the ten data sets into Equations (89) and (90) yields an estimate of the state vector and its covariance. The diagonal terms of the Monte Carlo estimate of the covariance matrix for the 4th order are listed below in Table 8. It is important to remember from Chapter II that the square root of the variance determines the standard deviation. Of notable interest are the results in columns 4 and 5. Column 4 displays the Monte Carlo estimation of both the true state and its standard deviation. Careful analysis shows that the true value of the state is within the bounds of the estimated value except in the $\ddot{a}$ components. The estimated values and the true values are quite off. Also, the magnitude of the estimated $\ddot{a}_y$ component is on order of 10^{-11} km/s³ with a much larger standard deviation on order of 10^{-8} km/s³.

Essentially, the estimator is incapable of properly estimating these small magnitudes. Evidence of this can be verified above in Table 7. All five data sets have vastly different values for the $\ddot{a}$ components. Since the estimator is unable to accurately estimate the $\ddot{a}$ components, it is inefficient to use the higher order equations. It seems quite reasonable to go down a level of accuracy, and simply use the 3rd order equations.

Table 8. Monte Carlo Estimated State Vector Values (4th Order)

Variable	Variance	Sigma	Estimated Value	True Value
x (km)	0.0023495	0.048471641	-6079.606304 ± 0.048471641	-6079.6
y (km)	0.002409	0.049081565	1837.916153 ± 0.049081565	1837.9
z (km)	0.00078343	0.02798982	-1596.608641 ± 0.02798982	-1596.6
$\dot{x}$ (km/s)	1.2272E-06	0.001107791	-2.959794471 ± 0.001107791	-2.96
$\dot{y}$ (km/s)	5.4565E-07	0.000738681	-5.650153622 ± 0.000738681	-5.65
$\dot{z}$ (km/s)	2.1741E-07	0.000466272	4.819836215 ± 0.000466272	4.82
a_x (km/s ²)	1.2268E-09	3.50257E-05	0.008631874 ± 3.50257E-05	0.008637157
a_y (km/s ²)	5.6799E-10	2.38325E-05	-0.002609078 ± 2.38325E-05	-0.002611065
a_z (km/s ²)	2.1518E-10	1.4669E-05	0.0022729 ± 1.4669E-05	0.002275235
$\dot{a}_x$ (km/s ³)	2.0364E-13	4.51265E-07	4.21247E-06 ± 4.51265E-07	4.2799E-06
$\dot{a}_y$ (km/s ³)	1.0374E-13	3.22087E-07	8.0802E-06 ± 3.22087E-07	8.00425E-06
$\dot{a}_z$ (km/s ³)	4.3888E-14	2.09495E-07	-6.8051E-06 ± 2.09495E-07	-6.84906E-06
$\ddot{a}_x$ (km/s ⁴)	6.9845E-16	2.64282E-08	-7.10425E-09 ± 2.64282E-08	-1.41524E-08
$\ddot{a}_y$ (km/s ⁴)	1.5985E-16	1.26432E-08	2.88701E-11 ± 1.26432E-08	4.41528E-09
$\ddot{a}_z$ (km/s ⁴)	8.2957E-17	9.10807E-09	-1.37777E-09 ± 9.10807E-09	-3.90375E-09

The process used for obtaining the 4th order data sets was also used to obtain the 3rd order data sets. Table 9 displays data sets 1-5. By inspection, it is apparent that the estimates in each row all have values that are quite close.

Table 9. State Vector Data Sets 1-5 (3rd Order)

	Set 1	Set 2	Set 3	Set 4	Set 5
x (km)	-6079.585097	-6079.644248	-6079.637398	-6079.621801	-6079.619295
y (km)	1837.878714	1837.955713	1837.93543	1837.901333	1837.910241
z (km)	-1596.572505	-1596.631851	-1596.62999	-1596.583491	-1596.598289
$\dot{x}$ (km/s)	-2.960541698	-2.957667187	-2.96116703	-2.957978792	-2.95971252
$\dot{y}$ (km/s)	-5.649885182	-5.650481368	-5.649712807	-5.650918108	-5.650543327
$\dot{z}$ (km/s)	4.820206618	4.819342736	4.819486692	4.820305664	4.820406233
a_x (km/s ²)	0.008625584	0.008638831	0.008647061	0.008642268	0.008646995
a_y (km/s ²)	-0.002604419	-0.002619199	-0.002618182	-0.002611856	-0.002608052
a_z (km/s ²)	0.00227618	0.002274249	0.00227742	0.002271089	0.002280282
$\dot{a}_x$ (km/s ³)	4.51282E-06	3.49377E-06	4.90508E-06	3.60324E-06	4.07412E-06
$\dot{a}_y$ (km/s ³)	7.94287E-06	8.2494E-06	7.83093E-06	8.33482E-06	8.0473E-06
$\dot{a}_z$ (km/s ³)	-6.76297E-06	-6.73871E-06	-6.69302E-06	-7.042E-06	-6.88487E-06

Table 10 displays the results for the estimated state and its covariance using the 3rd order equations of motion data sets. For every component of the state vector, the true state is within the bounds of the estimated value and standard deviation.

Table 10. Monte Carlo Estimated State Vector Values (3rd Order)

Variable	Variance	Sigma	Estimated Value	True Value
x (km)	0.00078788	0.0280692	-6079.617422 ± 0.0280692	-6079.6
y (km)	0.0016537	0.04066571	1837.92279 ± 0.04066571	1837.9
z (km)	0.0006166	0.024831432	-1596.609597 ± 0.024831432	-1596.6
$\dot{x}$ (km/s)	1.69990000E-06	0.001303802	-2.959698749 ± 0.001303802	-2.96
$\dot{y}$ (km/s)	3.12090000E-07	0.00055865	-5.650207329 ± 0.00055865	-5.65
$\dot{z}$ (km/s)	1.25700000E-07	0.000354542	4.819932336 ± 0.000354542	4.82
a_x (km/s ²)	1.02870000E-10	1.01425E-05	0.008643123 ± 1.01425E-05	0.008637157
a_y (km/s ²)	9.94220000E-11	9.97105811E-06	-0.002616 ± 9.97105811E-06	-0.002611065
a_z (km/s ²)	4.20220000E-11	6.48243781E-06	0.00227658 ± 6.48243781E-06	0.002275235
$\dot{a}_x$ (km/s ³)	2.58550000E-13	5.08478121E-07	4.19095E-06 ± 5.08478121E-07	4.2799E-06
$\dot{a}_y$ (km/s ³)	5.42150000E-14	2.32841147E-07	8.09606E-06 ± 2.32841147E-07	8.00425E-06
$\dot{a}_z$ (km/s ³)	2.11380000E-14	1.45389133E-07	-6.83767E-06 ± 1.45389133E-07	-6.84906E-06

The variances obtained in Tables 8 and 10 have different magnitudes than their respective variances in Table 5. Fortunately, there is a plausible explanation for this.

The Monte Carlo analysis is a weighted average, not an exact answer. The uncertainty in a weighted average drops off proportional to $\frac{1}{\sqrt{N}}$. In this case, since there are only ten

data sets, the uncertainty in the Monte Carlo covariance matrix is roughly 32%.

Producing more data sets would decrease the uncertainty but there will come a point where an enormous amount of data sets is required to improve results by only a fraction of a percentage.

Based on the above results, the 3rd order equations of motion appear quite efficient at modeling an object in orbit over very short arcs. The above results also

support the conclusion that the estimator is sufficient at obtaining the state and its covariance for the 3rd order equations of motion. The following sections will examine how well the 3rd order equations of motion and the computer model are able to estimate orbits with different initial conditions for position and velocity, as well as estimate orbits with various magnitudes of non-gravitational acceleration present.

MATLAB Simulations with Zero Non-Gravitational Acceleration

Equations (70) and (71) contain the initial conditions of the orbit that was used to validate the accuracy of the truth model to that of STK. The estimator also proved capable of estimating these initial conditions given a five-minute radar track of data. It is important, however, to ensure that the estimator is capable of estimating the state given different values for the initial position and velocity. A few cases with different initial conditions were tested. The first case was comprised of the following components

$$\vec{r}_0 = \begin{bmatrix} 1611 \\ -1756 \\ 6100 \end{bmatrix} \text{ km} \quad (91)$$

$$\vec{v}_0 = \begin{bmatrix} 7 \\ 0.4 \\ 3.84 \end{bmatrix} \text{ km/s} \quad (92)$$

The second case consisted of

$$\vec{r}_0 = \begin{bmatrix} 6250 \\ 500 \\ -1891 \end{bmatrix} \text{ km} \quad (93)$$

$$\bar{\mathbf{v}}_0 = \begin{bmatrix} 1.7 \\ -5 \\ 6 \end{bmatrix} \text{ km/s} \quad (94)$$

Both of these cases are in LEO with roughly a 170 km altitude and a velocity magnitude of about 7.99 km/s. Gaussian noise was added to the data and the following estimates in Table 11 were obtained for a single run.

Table 11. Estimated State Vectors

	Case A	Case B
x (km)	1611.22	6250.073
y (km)	-1755.86	500.7342
z (km)	6099.83	-1891.45
$\dot{x}$ (km/s)	6.99748	1.700548
$\dot{y}$ (km/s)	0.40367	-5.00128
$\dot{z}$ (km/s)	3.844742	6.003245
a_x (km/s ²)	-0.00234	-0.00889
a_y (km/s ²)	0.00249	-0.00079
a_z (km/s ²)	-0.00855	0.002731
$\dot{a}_x$ (km/s ³)	-3.2E-06	-4.6E-06
$\dot{a}_y$ (km/s ³)	-7.2E-06	6.42E-06
$\dot{a}_z$ (km/s ³)	1.33E-05	-7.3E-06

Based on the above table, it appears that the estimator is quite capable of determining the state with different initial position and velocity values.

MATLAB Simulations with Non-Gravitational Acceleration Present

Given the equations of motion listed in Equations (54)-(57), the truth model and estimator are limited in the magnitude of non-gravitational acceleration that can be detected. In order to determine this magnitude, it is necessary to solve the linear first-order differential equation:

$$\bar{r} = \frac{1}{2} \bar{A}_0 t^2 + \frac{1}{6} \dot{\bar{A}}_0 t^3 \quad (95)$$

Assuming that the radar sensor has a range error of 100 m, the non-gravitational acceleration acting on the space object must move the object more than 100 m between its first and last point in order to statistically validate that acceleration is present; therefore, the vector $\bar{r}$ is 100 m. Solving for $\bar{A}_0$ yields the general solution

$$\bar{A}_0 = \frac{600}{t^2} + \frac{C}{t^3} \quad (96)$$

Time is the length of the observation, which in this case is five minutes. A particular solution for $\bar{A}_0$ could be obtained if some initial conditions were known. In reality, however, the exact value of the constant (C) is unobtainable. If C=0, the magnitude of non-gravitational acceleration required for detection given the equations of motion is 0.667 cm/s^2 . This value is the smallest allowable magnitude given the aforementioned conditions. Given this requirement, all test cases used in this research have a magnitude greater than or equal to 0.667 cm/s^2 .

Monte Carlo Approach.

Six test cases were run through the truth model and estimator. Each test case produced ten data sets with different values of noise. Table 12 lists the magnitudes and individual components of non-gravitational acceleration used for each test case. Large accelerations such as $\frac{1}{2} g$ or greater usually produce noticeable results; therefore, it is the smaller accelerations of magnitude cm/s^2 which are of particular interest in this research.

Table 12. Non-Gravitational Acceleration Present

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
A_{0x} (km/s ²)	1.00E-04	1.00E-03	2.00E-04	9.00E-05	4.00E-06	2.89E-06
A_{0y} (km/s ²)	-2.00E-05	-1.00E-04	8.00E-05	2.00E-04	3.00E-06	-5.56E-06
A_{0z} (km/s ²)	-3.00E-05	-3.00E-04	3.00E-04	7.00E-05	1.00E-05	2.285E-06
Magnitude (km/s ²)	1.063E-04	1.0489E-03	3.69E-04	2.30E-04	1.12E-05	6.67E-06

The values listed in Table 12 were input into the truth model as the $\bar{A}_0$ component.

Table 13 displays the estimated state for Case 1 after the Monte Carlo analysis. Of special interest is the highlighted section. These values represent the estimated amount of total acceleration present for the initial state vector. The estimated value due to non-gravitational acceleration is found by subtracting the known gravitational acceleration value found in the previous chapter, (Equation (73)). Table 14 lists the estimated values for the non-gravitational acceleration components.

Table 13. Case 1 Monte Carlo Estimated State

Component	Estimated State
x (km)	-6079.601784
y (km)	1837.901274
z (km)	-1596.595273
$\dot{x}$ (km/s)	-2.959936964
$\dot{y}$ (km/s)	-5.6501156
$\dot{z}$ (km/s)	4.820181948
a_x (km/s ²)	0.008737164
a_y (km/s ²)	-0.002630849
a_z (km/s ²)	0.002243172
$\dot{a}_x$ (km/s ³)	4.28601E-06
$\dot{a}_y$ (km/s ³)	8.03442E-06
$\dot{a}_z$ (km/s ³)	-6.94113E-06

Table 14. Case 1 Estimated Non-gravitational Acceleration

	Case 1
A_{0_x} (km/s ²)	0.000100007
A_{0_y} (km/s ²)	-1.97844E-05
A_{0_z} (km/s ²)	-3.20633E-05

Listed in Table 15 are the values for the variance from the Monte Carlo covariance matrix. The standard deviation (σ) obtained from these variances is also listed in Table 15, as well as the estimated minimum and maximum values for non-gravitational acceleration, which were determined by taking the estimated value from Table 14 and both subtracting and adding the standard deviation, respectively. By inspection, it is apparent that the true value is located between the minimum and maximum estimates. So far, the truth model and estimator have yielded the desired results.

Table 15. Case 1 Monte Carlo Results

Variable	Variance	Sigma (σ)	Estimate (min)	Estimate (max)	True
A_{0_x} (km/s ²)	1.115E-10	1.05594E-05	8.94475E-05	1.10566E-04	1.00E-04
A_{0_y} (km/s ²)	1.0278E-11	3.20593E-06	-1.65785E-05	-2.29904E-05	-2.00E-05
A_{0_z} (km/s ²)	1.7407E-11	4.17217E-06	-2.78911E-05	-3.62355E-05	-3.00E-05

The process used to obtain the results in Tables 13-15 for Case 1 was used for all subsequent test cases. Table 16 displays the estimated state vectors for all test cases using the Monte Carlo analysis. The highlighted rows are the various estimated components of $\bar{a}$ with gravitational and non-gravitational acceleration combined.

Table 16. Monte Carlo Estimated State Vector

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
$\vec{r}$	-6079.601784	-6079.592763	-6079.617408	-6079.595845	-6079.605393	-6079.617422
	1837.901274	1837.899984	1837.922794	1837.897105	1837.910345	1837.92279
	-1596.595273	-1596.602284	-1596.609607	-1596.597832	-1596.610173	-1596.609597
$\vec{v}$	-2.959936964	-2.959752536	-2.959699776	-2.960101323	-2.959403419	-2.959698775
	-5.6501156	-5.649968385	-5.650207202	-5.650044755	-5.650208869	-5.650207324
	4.820181948	4.819868758	4.819932365	4.820190012	4.819921725	4.819932347
$\vec{a}$	0.008737164	0.009630106	0.008843116	0.008726374	0.008641327	0.008646013
	-0.002630849	-0.002711875	-0.00253619	-0.002409998	-0.002608829	-0.002621751
	0.002243172	0.001973669	0.002576583	0.002343208	0.002287571	0.002278865
$\dot{\vec{a}}$	4.28601E-06	4.16182E-06	4.19147E-06	4.36836E-06	4.03754E-06	4.19097E-06
	8.03442E-06	8.08126E-06	8.09605E-06	7.99999E-06	8.07227E-06	8.09606E-06
	-6.94113E-06	-6.82963E-06	-6.83764E-06	-6.95477E-06	-6.81189E-06	-6.83767E-06

Just like Case 1, the known gravitational acceleration value is subtracted from $\vec{a}$ to obtain the estimated non-gravitational acceleration values (Table 17). Tables 18-21 list the results for Cases 2-5.

Table 17. Estimated Non-gravitational Acceleration for all Cases

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
A_{0x}	0.000100007	0.000992949	0.00020596	8.92169E-05	4.17071E-06	8.85662E-06
A_{0y}	-1.97844E-05	-0.00010081	7.48753E-05	0.000201067	2.23559E-06	-1.06865E-05
A_{0z}	-3.20633E-05	-0.000301566	0.000301348	6.79723E-05	1.23354E-05	3.62937E-06

Table 18. Case 2 Monte Carlo Results

Variable	Variance	Sigma (σ)	Estimate	Estimate	True Value
A_{0x} (km/s ²)	6.3165E-11	7.94764E-06	9.85001E-04	1.000897E-03	1.00E-03
A_{0y} (km/s ²)	3.7116E-11	6.09229E-06	-1.06902E-04	-9.47179E-05	-1.00E-04
A_{0z} (km/s ²)	2.1172E-11	4.6013E-06	-3.06167E-04	-2.96965 E-04	-3.00E-04

Table 19. Case 3 Monte Carlo Results

Variable	Variance	Sigma (σ)	Estimate	Estimate	True Value
A_{0_x} (km/s ²)	1.027E-10	1.01341E-05	1.95826E-04	2.16094E-04	2.00E-04
A_{0_y} (km/s ²)	9.9128E-11	9.9563E-06	6.4919E-05	8.48316E-05	8.00E-05
A_{0_z} (km/s ²)	4.188E-11	6.47148E-06	2.94876E-04	3.07819E-04	3.00E-04

Table 20. Case 4 Monte Carlo Results

Variable	Variance	Sigma (σ)	Estimate	Estimate	True Value
A_{0_x} (km/s ²)	9.908E-11	9.95389E-06	7.9263E-05	9.91708E-05	9.00E-05
A_{0_y} (km/s ²)	1.1389E-11	3.37476E-06	1.97692 E-04	2.04442 E-04	2.00E-04
A_{0_z} (km/s ²)	1.7184E-11	4.14536E-06	6.38269E-05	7.21176E-05	7.00E-05

Table 21. Case 5 Monte Carlo Results

Variable	Variance	Sigma (σ)	Estimate	Estimate	True Value
A_{0_x} (km/s ²)	4.3922E-11	6.62737E-06	-2.45665E-06	1.07981E-05	4.00E-06
A_{0_y} (km/s ²)	4.4168E-11	6.6459E-06	-4.41031E-06	8.88149E-06	3.00E-06
A_{0_z} (km/s ²)	1.7455E-11	4.17792E-06	8.1575E-06	1.65133E-05	1.00E-05

There are several noteworthy outcomes obtained from Cases 1-5. First of all, the values obtained in Table 17 for Cases 1-5 are extremely close to the true non-gravitational acceleration values input into the truth model. It is also important to note that the minimum and maximum values found by taking into account the standard deviation are also quite close to the true value. The results from these cases support the validity of the estimator.

The results for Case 6 are quite different than those seen in the previous five cases. The non-gravitational acceleration values in Table 17 do not reflect the true values whatsoever. The standard deviation is also the highest it has been for any case. In fact, for the A_{0_x} component, the standard deviation has a greater magnitude than the estimated value. When the standard deviation is combined with the values in Table 17, there is no

clue as to what the true values are (Table 22). The estimator is incapable of estimating the state correctly when the magnitude of non-gravitational acceleration is that small.

Table 22. Case 6 Monte Carlo Results

Variable	Variance	Sigma (σ)	Estimate	Estimate	True Value
A_{0_x} (km/s ²)	1.0287E-10	1.01425E-05	-1.28586E-06	1.89991E-05	2.89E-06
A_{0_y} (km/s ²)	9.942E-11	9.97096E-06	-2.06574E-05	-7.15501E-07	-5.56E-06
A_{0_z} (km/s ²)	4.2024E-11	6.48259E-06	-2.85322E-06	1.0112E-05	2.285E-06

Interestingly enough, the results from Case 6 do make sense. The magnitude of non-gravitational acceleration used in Case 6 is the smallest magnitude of acceleration that the 3rd order equations of motion can detect given the conditions outlined in the beginning of this section. This is true if the constant (C) is in fact zero. The constant, however, is more than likely not zero but some other value. This would make the magnitude greater than 0.667 cm/s² if it is assumed that C is positive. That being the case, the estimator is unable to accurately estimate the state at such small magnitudes due to the limitations of the equations of motion themselves.

Real World Approach.

The above analysis would make it seem like the estimator is capable of detecting non-gravitational acceleration with magnitude as small as 1.12 cm/s²; unfortunately, the method used to obtain the above results does not reflect reality. In the real world, radar sites do not have N number of data sets for one particular satellite; they will most likely only have one data set to work with. This being the case, the results produced by the estimator must also be analyzed using a real world approach. Cases 1-5 will be analyzed

using the real world method. Case 6 has been left out because it has already been shown with the Monte Carlo analysis that the results are invalid.

Instead of averaging the ten estimates within each case to arrive at a final estimate, each estimate is analyzed as an individual ($N=1$) data set. For each of these data sets, the known value for gravitational acceleration is subtracted from $\bar{a}$ to obtain the estimated non-gravitational acceleration values. Table 23 lists these estimated non-gravitational acceleration values for five of the data sets in Case 1. Again, in reality a radar site will more than likely only have one data set to work with, but for the purposes of this research, it is beneficial to see if the correct results are obtained every time or only once in a while by chance. After comparing the estimated values to their true values in the last column of Table 23, the results for Case 1 appear to be relatively accurate; however, various components in some of the data sets are a little off. For example, the estimated value of the A_{0y} component for set 1 is closer to $-1E-05 \text{ km/s}^2$ and in set 2 and 3 it is closer to $-3E-05 \text{ km/s}^2$ while the true value is really $-2E-05 \text{ km/s}^2$.

As with any analysis, it is important to take into account the standard deviation in order to get a true understanding of the accuracy of the results. The standard deviation values used in this section were obtained from the MATLAB filter not the Monte Carlo estimates. Table 23 lists the $\pm 1\sigma$ and $\pm 2\sigma$ estimates for the various non-gravitational acceleration components for five of the data sets in Case 1.

Table 23. MATLAB Case 1 Results for Individual Data Sets

		-2σ	-1σ	Estimate	1σ	2σ	True
Set 1	A_{0x}	6.94671E-05	7.8947E-05	8.8428E-05	9.79094E-05	1.0739E-04	1.00E-04
	A_{0y}	-2.7336E-05	-2.0346E-05	-1.3355E-05	-6.36526E-06	6.25089E-07	-2.00E-05
	A_{0z}	-4.0340E-05	-3.4700E-05	-2.9060E-05	-2.34206E-05	-1.7780E-05	-3.00E-05
Set 2	A_{0x}	8.27126E-05	9.2193E-05	1.0167E-04	1.11154E-04	1.20635E-04	1.00E-04
	A_{0y}	-4.2118E-05	-3.5127E-05	-2.8136E-05	-2.11452E-05	-1.4154E-05	-2.00E-05
	A_{0z}	-4.2266E-05	-3.6626E-05	-3.0986E-05	-2.5347E-05	-1.9707E-05	-3.00E-05
Set 3	A_{0x}	9.09363E-05	1.0041E-04	1.0989E-04	1.19379E-04	1.28859E-04	1.00E-04
	A_{0y}	-4.1090E-05	-3.4100E-05	-2.7109E-05	-2.01198E-05	-1.3129E-05	-2.00E-05
	A_{0z}	-3.9096E-05	-3.3456E-05	-2.7816E-05	-2.21768E-05	-1.6536E-05	-3.00E-05
Set 4	A_{0x}	8.61503E-05	9.5630E-05	1.0511E-04	1.14592E-04	1.24072E-04	1.00E-04
	A_{0y}	-3.4771E-05	-2.7780E-05	-2.0790E-05	-1.37995E-05	-6.8089E-06	-2.00E-05
	A_{0z}	-4.5425E-05	-3.9785E-05	-3.4146E-05	-2.85064E-05	-2.2866E-05	-3.00E-05
Set 5	A_{0x}	9.08761E-05	1.0035E-04	1.0983E-04	1.19318E-04	1.28799E-04	1.00E-04
	A_{0y}	-3.0967E-05	-2.3977E-05	-1.6986E-05	-9.99657E-06	-3.0062E-06	-2.00E-05
	A_{0z}	-3.6229E-05	-3.0589E-05	-2.495E-05	-1.93102E-05	-1.3670E-05	-3.00E-05

By inspection, there are several components where the estimate does not contain the true value within $\pm 1\sigma$. These components have been highlighted for easier identification. These highlighted components happen to be off by a magnitude of only 10^{-6} km/s^2 or 10^{-7} km/s^2 . This error is minimal considering the true values are of magnitude 10^{-4} km/s^2 and 10^{-5} km/s^2 .

Of the 15 components listed in Table 23, five of them do not contain their true value within $\pm 1\sigma$. In other words, roughly 33% of the answers are not within $\pm 1\sigma$. In Gaussian statistics the probability that the answer is not within $\pm 1\sigma$ is 32%, therefore, the distribution of the results in Case 1 is expected. All data sets in Case 1 are accurate within $\pm 2\sigma$.

The results for Cases 2 and 3 are quite similar to Case 1. The estimated values in Table 24 for Case 2 and Table 25 for Case 3 are pretty consistent across the data sets. Compared to their true values in the final column of Tables 24 and 25, the estimates appear to be very close. In fact, Cases 2 and 3 appear to be better at estimating the true accelerations than Case 1. This may be because Cases 2 and 3 have larger non-gravitational acceleration components, which makes it harder for their values to get lost in the noise.

Table 24. MATLAB Case 2 Results for Individual Data Sets

		-2σ	-1σ	Estimate	1σ	2σ	True
Set 1	A_{0x}	9.81200E-04	9.90686E-04	1.00017E-03	1.00966E-03	1.01914E-03	1.00E-03
	A_{0y}	-1.0891E-04	-1.0192E-04	-9.4929E-05	-8.7936E-05	-8.0943E-05	-1.00E-04
	A_{0z}	-3.1595E-04	-3.1031E-04	-3.0467E-04	-2.9903E-04	-2.9340E-04	-3.00E-04
Set 2	A_{0x}	9.85444E-04	9.94930E-04	1.00442E-03	1.01390E-03	1.02339E-03	1.00E-03
	A_{0y}	-1.3704E-04	-1.3005E-04	-1.2305E-04	-1.1606E-04	-1.0907E-04	-1.00E-04
	A_{0z}	-3.0151E-04	-2.9587E-04	-2.9024E-04	-2.8460E-04	-2.7896E-04	-3.00E-04
Set 3	A_{0x}	9.87489E-04	9.96975E-04	1.00646E-03	1.01595E-03	1.02543E-03	1.00E-03
	A_{0y}	-1.2624E-04	-1.1925E-04	-1.1225E-04	-1.0526E-04	-9.8271E-05	-1.00E-04
	A_{0z}	-3.0115E-04	-2.9551E-04	-2.8987E-04	-2.8424E-04	-2.7860E-04	-3.00E-04
Set 4	A_{0x}	9.93237E-04	1.00272E-03	1.01221E-03	1.02169E-03	1.03118E-03	1.00E-03
	A_{0y}	-1.1915E-04	-1.1216E-04	-1.0517E-04	-9.8176E-05	-9.1183E-05	-1.00E-04
	A_{0z}	-3.0529E-04	-2.9965E-04	-2.9401E-04	-2.8837E-04	-2.8273E-04	-3.00E-04
Set 5	A_{0x}	1.00255E-03	1.01204E-03	1.02152E-03	1.03101E-03	1.04049E-03	1.00E-03
	A_{0y}	-1.2343E-04	-1.1643E-04	-1.0944E-04	-1.0245E-04	-9.5458E-05	-1.00E-04
	A_{0z}	-3.2194E-04	-3.1630E-04	-3.1066E-04	-3.0502E-04	-2.9938E-04	-3.00E-04

Tables 24 and 25 also take into account the standard deviation for Cases 2 and 3. As seen with Case 1, most of these data sets have a component where the estimated value is not within $\pm 1\sigma$ of the true value. Again, these components have been highlighted. These components are off by a magnitude of 10^{-6} km/s² or 10^{-7} km/s² error. Since the

true values are of magnitude 10^{-3} km/s² through 10^{-5} km/s², the magnitude of error does not necessarily destroy the entire validity of the results. It is still easy to ascertain the magnitude of the true values.

Table 25. MATLAB Case 3 Results for Individual Data Sets

		-2σ	-1σ	Estimate	1σ	2σ	True
Set 1	A_{0x}	1.84874E-04	1.94345E-04	2.03815E-04	2.13286E-04	2.22757E-04	2.00E-04
	A_{0y}	7.09803E-05	7.79598E-05	8.49393E-05	9.19188E-05	9.88982E-05	8.00E-05
	A_{0z}	2.86076E-04	2.91711E-04	2.97346E-04	3.02982E-04	3.08617E-04	3.00E-04
Set 2	A_{0x}	1.62638E-04	1.72109E-04	1.81579E-04	1.9105E-04	2.0052E-04	2.00E-04
	A_{0y}	6.83558E-05	7.53357E-05	8.23156E-05	8.92955E-05	9.62754E-05	8.00E-05
	A_{0z}	2.86151E-04	2.91786E-04	2.97421E-04	3.03056E-04	3.08691E-04	3.00E-04
Set 3	A_{0x}	1.75432E-04	1.84902E-04	1.94373E-04	2.03843E-04	2.13314E-04	2.00E-04
	A_{0y}	6.40875E-05	7.10672E-05	7.80468E-05	8.50265E-05	9.20062E-05	8.00E-05
	A_{0z}	2.9456E-04	3.00195E-04	3.05829E-04	3.11464E-04	3.17099E-04	3.00E-04
Set 4	A_{0x}	1.97613E-04	2.07083E-04	2.16553E-04	2.26023E-04	2.35493E-04	2.00E-04
	A_{0y}	6.43551E-05	7.13347E-05	7.83143E-05	8.52939E-05	9.22735E-05	8.00E-05
	A_{0z}	2.88561E-04	2.99831E-04	2.94196E-04	2.82926E-04	3.05466E-04	3.00E-04
Set 5	A_{0x}	1.7555E-04	1.94491E-04	1.8502E-04	1.66079E-04	2.03961E-04	2.00E-04
	A_{0y}	7.36103E-05	8.75702E-05	8.059E-05	6.66303E-05	9.45502E-05	8.00E-05
	A_{0z}	2.91338E-04	3.02608E-04	2.9697E-04	2.85703E-04	3.08243E-04	3.00E-04

As seen with Cases 1-3, the estimates in Table 26 for Case 4 are consistent across the data sets and are close to portraying the true values. In fact, the results in Table 26 seem to be the best. Data set 5 is the only set that has a component not within $\pm 1\sigma$ of the true value.

Table 26. MATLAB Case 4 Results for Individual Data Sets

		-2σ	-1σ	Estimate	1σ	2σ	True
Set 1	A_{0x}	8.01006E-05	8.95745E-05	9.90484E-05	1.08522E-04	1.17996E-04	9.00E-05
	A_{0y}	1.81295E-04	1.88282E-04	1.95268E-04	2.02255E-04	2.09241E-04	2.00E-04
	A_{0z}	5.89685E-05	6.46025E-05	7.02365E-05	7.58705E-05	8.15045E-05	7.00E-05
Set 2	A_{0x}	7.24923E-05	8.19667E-05	9.1441E-05	1.00915E-04	1.1039E-04	9.00E-05
	A_{0y}	1.89723E-04	1.96709E-04	2.03695E-04	2.10681E-04	2.1766E-04	2.00E-04
	A_{0z}	5.33809E-05	5.90152E-05	6.46495E-05	7.02837E-05	7.5918E-05	7.00E-05
Set 3	A_{0x}	7.45168E-05	8.39912E-05	9.34655E-05	1.0294E-04	1.12414E-04	9.00E-05
	A_{0y}	1.85354E-04	1.9234E-04	1.99326E-04	2.06313E-04	2.13299E-04	2.00E-04
	A_{0z}	6.01894E-05	6.58237E-05	7.1458E-05	7.70922E-05	8.27265E-05	7.00E-05
Set 4	A_{0x}	6.78759E-05	7.73498E-05	8.68237E-05	9.62976E-05	1.05772E-04	9.00E-05
	A_{0y}	1.8821E-04	1.95196E-04	2.02183E-04	2.09169E-04	2.16156E-04	2.00E-04
	A_{0z}	5.72881E-05	6.29223E-05	6.85566E-05	7.41909E-05	7.98251E-05	7.00E-05
Set 5	A_{0x}	7.11074E-05	8.05815E-05	9.00557E-05	9.95298E-05	1.09004E-04	9.00E-05
	A_{0y}	1.9201E-04	1.98996E-04	2.05982E-04	2.12968E-04	2.19954E-04	2.00E-04
	A_{0z}	5.17785E-05	5.74129E-05	6.30472E-05	6.86816E-05	7.43159E-05	7.00E-05

In the previous section, Case 5 contained the smallest magnitude of non-gravitational acceleration that the estimator could detect using the Monte Carlo method. The results in Table 27, however, show that the estimator cannot find the estimate reliably. Not a single set has estimated values that are close to the true non-gravitational acceleration values. The true value may have been obtainable after taking an average using Monte Carlo analysis; but as separate individual estimates the true values are unclear.

Table 27. MATLAB Case 5 Results for Individual Data Sets

		-2σ	-1σ	Estimate	1σ	2σ	True
Set 1	A_{0x}	-3.2597E-06	6.2199E-06	1.5699E-05	2.5179E-05	3.4658E-05	4.00E-06
	A_{0y}	-1.4599E-05	-7.6089E-06	-6.189E-07	6.3711E-06	1.3361E-05	3.00E-06
	A_{0z}	-4.2422E-06	1.3977E-06	7.0376E-06	1.2677E-05	1.8317E-05	1.00E-05
Set 2	A_{0x}	-2.4232E-05	-1.4752E-05	-5.2729E-06	4.2068E-06	1.3686E-05	4.00E-06
	A_{0y}	9.6891E-07	7.9588E-06	1.4948E-05	2.1938E-05	2.8928E-05	3.00E-06
	A_{0z}	-4.1779E-06	1.4619E-06	7.1017E-06	1.2741E-05	1.8381E-05	1.00E-05
Set 3	A_{0x}	-2.381E-05	-1.4330E-05	-4.8512E-06	4.6280E-06	1.4107E-05	4.00E-06
	A_{0y}	-9.0425E-06	-2.0522E-06	4.9380E-06	1.1928E-05	1.8918E-05	3.00E-06
	A_{0z}	-6.0754E-06	-4.3569E-07	5.2040E-06	1.0843E-05	1.6483E-05	1.00E-05
Set 4	A_{0x}	-2.0674E-05	-1.1195E-05	-1.7156E-06	7.7639E-06	1.7243E-05	4.00E-06
	A_{0y}	-2.4414E-05	-1.7424E-05	-1.0434E-05	-3.4442E-06	3.5457E-06	3.00E-06
	A_{0z}	-1.7366E-06	3.9030E-06	9.5427E-06	1.5182E-05	2.0822E-05	1.00E-05
Set 5	A_{0x}	-2.0173E-05	-1.0694E-05	-1.2148E-06	8.2646E-06	1.7744E-05	4.00E-06
	A_{0y}	-1.3768E-05	-6.7779E-06	2.1217E-07	7.2023E-06	1.4192E-05	3.00E-06
	A_{0z}	-3.4714E-06	2.1676E-06	7.8068E-06	1.3446E-05	1.9085E-05	1.00E-05

Table 27 also lists the values for $\pm 1\sigma$ and $\pm 2\sigma$. At first glance, it may seem that since there are only three highlighted components the results must be fairly good, but this is not the case. The amount of error between these highlighted components to their true values is of magnitude 10^{-6} km/s². This also happens to be the same magnitude of the true values themselves. Ultimately, what this means is that the estimator has no idea what the true values really are and does a poor job estimating at such a small magnitude.

Depending on the actual use of the estimator, results such as these may still be useful. The estimator is still indicating that there is non-gravitational acceleration present even if the estimate is not statistically accurate. Sometimes it is more important to assume an object has non-gravitational acceleration present when it does not than to assume an object does not have non-gravitational acceleration when it does. Further

analysis can always be conducted on the objects that are assumed to contain non-gravitational acceleration.

After careful analysis of the previous five cases, the following conclusions can be drawn. The uncertainty in the results for Case 5 is too high for the estimator to be of much use at such a small magnitude of non-gravitational acceleration. The results seen in Cases 1-4 are much more accurate. Of these four cases, Case 1 had the smallest magnitude at 10.63 cm/s^2 . Based on the above analysis and results, it is fairly safe to say that the estimator can detect non-gravitational acceleration down to a magnitude of 10.63 cm/s^2 . Non-gravitational acceleration that only has components of magnitude 10^{-6} km/s^2 and 10^{-5} km/s^2 seem to get lost in both the noise and the estimation capability of the estimator. In these cases, the estimator is able to detect the non-gravitational acceleration but is unable to give a truly decent estimate as to its true value.

The inability to measure these small magnitudes may be caused by a combination of factors. First and foremost, the general equations used to model the motion of the space object already limits the detection level of the results. Also, it is important to realize that there are going to be inaccuracies caused by MATLAB itself. MATLAB, as with many other computer programs, will ultimately truncate numbers as they are being run through the computer code. This truncation may happen in the 10^{th} decimal place or even higher and may seem insignificant, but can produce very noticeable results especially when working with such small acceleration values.

Investigative Questions Answered

Several questions were posed in Chapter I concerning the development of the truth model and the detection capability of the estimator. The first question, concerning how much non-gravitational acceleration the estimator would be able to detect is answered after analyzing numerous test cases. It was shown that the estimator should be able to detect as little as 10.63 cm/s^2 in magnitude. The second question dealt with Galileo's projectile equations and whether or not they would be accurate enough to model an object in space. It was shown that these equations were in fact too general to obtain sufficient results; however, a Taylor series expansion to the 3rd order achieved success. The third question was concerning the geopotential. Multiple terms such as J_2 , J_4 or two-body effects can go into determining the geopotential. For this research, only two-body and J_2 effects comprised the terms of the geopotential. These two terms proved quite effective at modeling the geopotential and its gradient. Proof can be seen in Equations (73) and (74) where the MATLAB and STK acceleration values at epoch are nearly identical.

Summary

Various methods were used to analyze the simulated data to obtain valid results. An STK simulation was used as a baseline model to verify the accuracy of the equations of motion. The 2nd, 3rd, and 4th order Taylor series approximations were all tested to see which would yield better results at modeling an orbiting space object. The 3rd order equations of motion were shown to have the required accuracy necessary for detecting non-gravitational acceleration. Numerous test cases (both with and without non-

gravitational acceleration present) were tested with the estimator. The Monte Carlo method was used to verify the statistics of the results. A real world approach was used to analyze the various scenarios to determine the smallest magnitude that could be determined with certainty. Using the least squares estimator, accelerations with a magnitude as small as 10.63 cm/s^2 were detectable with statistical accuracy. At magnitudes of acceleration smaller than 10.63 cm/s^2 , confidence and validity of the results declines. Nearly all estimates were within 2σ ; however, this is not beneficial in the test cases where the magnitude of standard deviation is the same as or greater than the magnitude of the estimated values.

V. Conclusions and Recommendations

Chapter Overview

The purpose of this chapter is to conclude the findings and discuss the significance of this research. Suggestions as to what actions should be taken as a result of the findings will be explored. Various aspects of further research will also be addressed.

Conclusions of Research

It has been shown that a 3rd order Taylor series expansion can adequately model objects in LEO for a short period of time. These equations of motion, with the use of a truth model and linear least squares estimator, can detect constant non-gravitational acceleration down to roughly 10.63 cm/s^2 in magnitude with statistical accuracy. If the amount of non-gravitational acceleration is smaller than the 10.63 cm/s^2 limit, the estimator produces poor results. At the very least, the estimator is capable of detecting non-gravitational acceleration but may not necessarily be able to determine the exact magnitude of its components within 1σ .

Significance of Research

There are several significant aspects of this research. First of all, the above research has shown that the motion of space objects can be modeled using linear dynamics for a short time span. Adding non-gravitational acceleration to the dynamics, (as seen with objects such as tethered systems, maneuvering satellites, and thrusting ballistic missiles) the effects of non-Keplerian motion can now be tracked using linear equations. Even the data in range, azimuth, and elevation format can be exchanged for

linear data in the form of position and velocity. The use of a linear model enables estimates of the initial state to be calculated at the click of a button. This is much more favorable than that of iterative methods seen in nonlinear systems.

There are currently a plethora of elaborate methods available for identifying and tracking objects that are solely tethered or solely ballistic missiles; however, there are not many methods that follow a more general approach and distinguish simply between Keplerian and non-Keplerian objects. This general approach allows for the quicker detection of non-Keplerian objects. Then, if necessary, a more in-depth analysis can be used to identify whether the object is a tethered system, an object on re-entry, or a maneuvering satellite.

Recommendations for Action

It is recommended that radar sites utilize this computer model in order to filter out space objects that contain a certain level of non-gravitational acceleration. This computer model is not able to track objects for an extended period of time, nor is it able to distinguish between a tethered system and an object on re-entry; however, it is quite useful as a filter. The model is rather effective at determining the state of an object with or without non-gravitational acceleration in just seconds. This is very important when time is of the essence. Depending on the magnitude of the non-gravitational acceleration present, other identification and tracking methods can be used to provide further insight into the object of interest.

Recommendations for Future Research

There are various avenues of future research dealing with the current research and, in general, the area of tracking objects with non-Keplerian motion. The above research was tested and validated using models and simulated data. It would be quite beneficial to test the estimator using real-world data. Perhaps testing the estimator with data from satellites that have on-board accelerometers could be advantageous. This way, there is still some indication as to the true value of non-gravitational acceleration present. Another recommendation would be to develop a model that can detect variable non-gravitational acceleration. The current model only considers constant non-gravitational acceleration.

Another idea to consider is to determine the initial state vector in the space object's coordinate frame instead of working in the inertial IJK coordinate frame. This approach would provide some very important insights. For instance, a lot can be learned about a tethered system simply given the various components of non-gravitational acceleration in the satellite's coordinate frame. Non-gravitational acceleration in a tethered system can only be located in the tangential or radial directions. If the magnitude in both of these directions is zero, then the object is not tethered. If the tangential is zero but the radial is not, then it can be deduced that the system is oriented vertically. If the radial is zero and the tangential is not, then the system is horizontally oriented. If neither component is zero, then the tether orientation angle, magnitude of tether force, apparent gravitational parameter, and distance to the center of mass can be determined (Cicci and others, 2001a: 316-317). Unfortunately, these statements are true

only if the tethered system is moving in the plane. Out of plane motion requires a different approach.

Summary

Objects that contain non-gravitational acceleration; such as tethered systems, thrusting ballistic missiles and maneuvering satellites; can be modeled using a 3rd order Taylor series expansion. The combination of these dynamics with a linear least squares estimator provides the ability to accurately estimate the initial state and covariance of an object in space. The sum of gravitational and non-gravitational acceleration acting on the object is part of the state vector. A non-zero magnitude for non-gravitational acceleration means that the space object is following a non-Keplerian orbit. The current model used for this research is capable of detecting non-gravitational acceleration as small as 10.63 cm/s^2 in magnitude.

The identification and tracking of space objects will remain an important pursuit for years to come. As technology increases, the dynamics of a space object will more than likely not follow a regular Keplerian orbit due to the desire for increased maneuverability for the protection against potential space weapons. Objects such as tethered satellite systems, maneuvering satellites and thrusting ballistic missiles already follow non-Keplerian orbits. Filters, such as the one produced in this research, will become a necessity.

Appendix A: Equations of Motion

Second Order

$$\begin{aligned}\bar{r} &= \bar{r}_0 + \bar{v}_0 \Delta t + \frac{1}{2}(\bar{A}_0 + \bar{g}_0) \Delta t^2 \\ \bar{v} &= \bar{v}_0 + (\bar{A}_0 + \bar{g}_0) \Delta t \\ \bar{a} &= (\bar{A}_0 + \bar{g}_0)\end{aligned}$$

Third Order

$$\begin{aligned}\bar{r} &= \bar{r}_0 + \bar{v}_0 \Delta t + \frac{1}{2}(\bar{A}_0 + \bar{g}_0) \Delta t^2 + \frac{1}{6}(\dot{\bar{A}}_0 + \dot{\bar{g}}_0) \Delta t^3 \\ \bar{v} &= \bar{v}_0 + (\bar{A}_0 + \bar{g}_0) \Delta t + \frac{1}{2}(\dot{\bar{A}}_0 + \dot{\bar{g}}_0) \Delta t^2 \\ \bar{a} &= (\bar{A}_0 + \bar{g}_0) + (\dot{\bar{A}}_0 + \dot{\bar{g}}_0) \Delta t \\ \dot{\bar{a}} &= (\dot{\bar{A}}_0 + \dot{\bar{g}}_0)\end{aligned}$$

Fourth Order

$$\begin{aligned}\bar{r} &= \bar{r}_0 + \bar{v}_0 \Delta t + \frac{1}{2}(\bar{A}_0 + \bar{g}_0) \Delta t^2 + \frac{1}{6}(\dot{\bar{A}}_0 + \dot{\bar{g}}_0) \Delta t^3 + \frac{1}{24}(\ddot{\bar{A}}_0 + \ddot{\bar{g}}_0) \Delta t^4 \\ \bar{v} &= \bar{v}_0 + (\bar{A}_0 + \bar{g}_0) \Delta t + \frac{1}{2}(\dot{\bar{A}}_0 + \dot{\bar{g}}_0) \Delta t^2 + \frac{1}{6}(\ddot{\bar{A}}_0 + \ddot{\bar{g}}_0) \Delta t^3 \\ \bar{a} &= (\bar{A}_0 + \bar{g}_0) + (\dot{\bar{A}}_0 + \dot{\bar{g}}_0) \Delta t + \frac{1}{2}(\ddot{\bar{A}}_0 + \ddot{\bar{g}}_0) \Delta t^2 \\ \dot{\bar{a}} &= (\dot{\bar{A}}_0 + \dot{\bar{g}}_0) + (\ddot{\bar{A}}_0 + \ddot{\bar{g}}_0) \Delta t \\ \ddot{\bar{a}} &= (\ddot{\bar{A}}_0 + \ddot{\bar{g}}_0)\end{aligned}$$

Appendix B: MATLAB Truth Model

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% Author: 2Lt Sandra Rashash, USAF, 17 May 07
%%
%%
%% Outputs:
%%   el_matrix      -matrix of elevation values      rad
%%   az_matrix      -matrix of azimuth values        rad
%%   rho_matrix      -matrix of range values          km
%%   sitlat          -radar site latitude             rad
%%   sitlon          -radar site longitude            rad
%%   sitalt          -radar site altitude             km
%%   julian_matrix   -matrix of julian days
%%   range_noise     -range noise                     km
%%   az_noise        -azimuth noise                   rad
%%   el_noise        -elevation noise                 rad
%%
%% Locals:
%%   r_init          -initial position at epoch in IJK      km
%%   x                -'I' component of position            km
%%   y                -'J' component of position            km
%%   z                -'K' component of position            km
%%   v_init          -initial velocity at epoch             km/s
%%   A                -extra acceleration input             km/s^2
%%   t                -counter for time
%%   jd              -time in julian days
%%   gst              -greenwich sidereal time              rad
%%   local_sidereal_time -local sidereal time                rad
%%   lst              -matrix of local sidereal time        rad
%%   time             -time since epoch                     sec
%%   timematrix       -matrix of times since epoch          sec
%%   geopotential     -geopotential                        m^2/s^2
%%   ax               -'x' component of acceleration        km/s^2
%%   ay               -'y' component of acceleration        km/s^2
%%   az               -'z' component of acceleration        km/s^2
%%   g                -local gravity                       km/s^2
%%   r                -new position vector                  km
%%   v                -new velocity vector                  km/s
%%   i                -index variable
%%   vmatrix          -matrix of velocities                km/s
%%   x_site           -station coordinate for Ellipsoidal Earth km
%%   z_site           -station coordinate for Ellipsoidal Earth km
%%   R_site           -position vector of radar site        km
%%   rot_ijk_sez      -rotation matrix from IJK to SEZ      rad
%%   range_sez        -range vector in SEZ                 km
%%   rho_s            -'s' component of range               km
%%   rho_e            -'e' component of range               km
%%   rho_z            -'z' component of range               km
%%   range_sez_matrix -matrix of range values              km

```

```

%% rho -range km
%% az -azimuth rad
%% el -elevation rad
%% noise -matrix of noise values
%% ww -index variable
%%
%% Note: An extra blank line is present at the end of the output file that must be deleted before sending
the data to the estimator.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

WGS84Data
format long g
global MU J2 RE TwoPI EEarth Rad
radardata1=fopen('radardata1.dat','wt');

% site data
sitlat= Rad*(-7.91);
sitlon=Rad*(-14.4);
sitalt=56.1*10^-3;
range_noise=101.7*10^-3;
el_noise=Rad*(0.0283);
az_noise=Rad*(0.0248);

% initial conditions
r_init =[-6079.6;1837.9;-1596.6];

x = r_init(1,1);
y = r_init(2,1);
z = r_init(3,1);

v_init=[-2.96;-5.65;4.82];

u = v_init(1,1);
v = v_init(2,1);
w = v_init(3,1);

A=[0;0;0];
A_dot=[0;0;0];

% creates a 5 minute matrix of julian days for a given start time
julian_matrix(1,:)= JulianDay(2007,9,13,12,0,0);

for t=1:299
    jd = JulianDay(2007,9,13,12,0,t);
    julian_matrix(t+1,:)=jd;
end

```

```

% determines matrix for the local sidereal time
for index=1:300

    % calculates greenwich sidereal time
    gst = GSTime (julian_matrix(index,:));

    % calculates local sidereal time
    local_sidereal_time = gst + sitlon;

    % calls revcheck to obtain an lst less than 2pi
    local_sidereal_time = revcheck(local_sidereal_time, TwoPI);

    lst(index,:)=local_sidereal_time;

    % converts julian days into time since epoch observation
    time = (julian_matrix(index,:)-julian_matrix(151,1))*86400;

    % creates matrix of observation times
    timematrix(index,:) =time;

end

%% Determination of the gravity vector
% geopotential taking into account J2 and 2-body
% geopotential=(-
MU/sqrt(x^2+y^2+z^2))+((MU*RE^2*J2)/(2*(x^2+y^2+z^2)^(3/2)))*(3*z^2/(x^2+y^2+z^2)-1);
%
% % the negative gradient of the geopotential
%
% ax=-MU/(x^2+y^2+z^2)^(3/2)*x+15/2*z^2*MU*RE^2*J2/(x^2+y^2+z^2)^(7/2)*x-
3/2*MU*RE^2*J2/(x^2+y^2+z^2)^(5/2)*x;
%
% ay=-MU/(x^2+y^2+z^2)^(3/2)*y+15/2*z^2*MU*RE^2*J2/(x^2+y^2+z^2)^(7/2)*y-
3/2*MU*RE^2*J2/(x^2+y^2+z^2)^(5/2)*y;
%
% az=-MU/(x^2+y^2+z^2)^(3/2)*z-
9/2*z*MU*RE^2*J2/(x^2+y^2+z^2)^(5/2)+15/2*z^3*MU*RE^2*J2/(x^2+y^2+z^2)^(7/2);
%
% g=[ax;ay;az]

% 2-body acceleration and its first and second derivatives
a_2body= -MU*r_init*(mag(r_init)^-3);

a_2body_dot= 3*MU*r_init*dot(r_init,v_init)*(mag(r_init)^-5) - MU*v_init*(mag(r_init)^-3);

```

```

% J2 acceleration and its first and second derivatives
a_j2= [15/2*z^2*MU*RE^2*J2/(x^2+y^2+z^2)^(7/2)*x-3/2*MU*RE^2*J2/(x^2+y^2+z^2)^(5/2)*x;...
15/2*z^2*MU*RE^2*J2/(x^2+y^2+z^2)^(7/2)*y-3/2*MU*RE^2*J2/(x^2+y^2+z^2)^(5/2)*y;...
-9/2*z*MU*RE^2*J2/(x^2+y^2+z^2)^(5/2)+15/2*z^3*MU*RE^2*J2/(x^2+y^2+z^2)^(7/2)];

a_j2_dot= -.5*MU*RE^2*J2*[(-15*x*dot(r_init,v_init)*(mag(r_init)^-7)+3*u*(mag(r_init)^-5)...
+105*x*z^2*dot(r_init,v_init)*(mag(r_init)^-9)-30*x*z*w*(mag(r_init)^-7)-15*u*z^2*(mag(r_init)^-
7));...
(-15*y*dot(r_init,v_init)*(mag(r_init)^-7)+3*v*(mag(r_init)^-5)+105*y*z^2*dot(r_init,v_init)*...
(mag(r_init)^-9)-30*y*z*w*(mag(r_init)^-7)-15*v*z^2*(mag(r_init)^-7));...
(-45*z*dot(r_init,v_init)*(mag(r_init)^-7)+9*w*(mag(r_init)^-
5)+105*z^3*dot(r_init,v_init)*(mag(r_init)^-9)-...
45*z^2*w*(mag(r_init)^-7))];

% total acceleration and derivatives
g=a_2body+a_j2
g_dot=a_2body_dot+ a_j2_dot

%%% Equations of Motion for a 5 minute radar track using Taylor series
for w=1:300
    t=timematrix(w,:);

    % finds position
    r = r_init+ v_init*t+((A+g)/2)*t^2+((A_dot+g_dot)/6)*t^3;
    R_ijk_matrix(w,:)=r;

    % finds velocity
    v = v_init+(A+g)*t+((A_dot+g_dot)/2)*t^2;
    vmatrix(w,:)=v;

    % finds acceleration
    a = (A+g) + (A_dot+g_dot)*t;
    amatrix(w,:)=a;

    % finds 1st derivative of accel
    a_dot=(A_dot+g_dot);
    a_dot_matrix(w,:)=a_dot;

end

% calculates x in the R_site equation
x_site = (RE/sqrt(1-EEarth^2*sin(sitlat)^2)+sitalt)*cos(sitlat);

% calculates z in the R_site equation
z_site = ((RE*(1-EEarth^2))/sqrt(1-EEarth^2*sin(sitlat)^2)+sitalt)*sin(sitlat);

% Determines range, azimuth, and elevation matrices
for i=1:300

```

```

% calculates R_site
R_site = [x_site*cos(lst(i)); x_site*sin(lst(i)); z_site];

% rotation matrix from the IJK to SEZ coord. frame
rot_ijk_sez = [sin(sitlat)*cos(lst(i)) sin(sitlat)*sin(lst(i)) -cos(sitlat);...
    -sin(lst(i)) cos(lst(i)) 0; cos(sitlat)*cos(lst(i)) cos(sitlat)*sin(lst(i))...
    sin(sitlat)];

range_sez=rot_ijk_sez*(R_ijk_matrix(i,:)'-R_site);
range_sez_matrix(i,:)=range_sez;

rho_s =range_sez_matrix(i,1);
rho_e =range_sez_matrix(i,2);
rho_z =range_sez_matrix(i,3);

% determines range
rho=sqrt(rho_s^2+rho_e^2+rho_z^2);
rho_matrix(i,:)=rho;

% determines azimuth
az = atan2(rho_e,-rho_s);
if az<0
    az=az+TwoPI;
end
az_matrix(i,:)=az;

% determines elevation
el = asin(rho_z/rho);
el_matrix(i,:)=el;
end

% adds noise to the range,azimuth, elevation data using gaussian random number generator
noise1=randn(300,1)*.1017;
noise2=randn(300,1)*.000433;
noise3=randn(300,1)*.000494;

%outputs data to a file
for ww=1:300

fprintf(radardata1,'%22.15g',sitlat,sitlon,sitalt,julian_matrix(ww),rho_matrix(ww)+noise1(ww),az_matrix(
ww)+noise2(ww),el_matrix(ww)+noise3(ww),range_noise,az_noise,el_noise);
    fprintf(radardata1,'\n');

end

% %%% data with no noise
% for ww=1:300
%
fprintf(radardata1,'%22.15g',sitlat,sitlon,sitalt,julian_matrix(ww),rho_matrix(ww),az_matrix(ww),el_matrix
(ww),range_noise,az_noise,el_noise);
%     fprintf(radardata1,'\n');
%
% end
fclose(radardata1);

```

Appendix C: MATLAB Estimator

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%
%% Author: 2Lt Sandra Rashash, USAF, 17 May 07
%%
%%
%% Locals:
%%   R_ijk_matrix      -matrix of position in IJK coords          km
%%   timematrix        -matrix of observation times              sec
%%   lst              -matrix of local sidereal time             rad
%%   el_matrix        -matrix of elevation                     rad
%%   az_matrix        -matrix of azimuth                       rad
%%   rho_matrix       -matrix of range                          km
%%   t                -time variable                            sec
%%   jdtime           -matrix of julian days
%%   local_sidereal_time -local sidereal time                    rad
%%   R_site           -site position vector in IJK              km
%%   row              -number of rows in position matrix
%%   rot_ijk_sez       -rotation matrix from IJK to SEZ          rad
%%   phi_matrix        -state transition matrix
%%   phi_multi_array   -multi-dimensional array of all phi matrices
%%   w                -index variable
%%   s                -index variable
%%   e                -index variable
%%   v                -index variable
%%   f                -index variable
%%   H                -matrix of the linearized observation relation
%%   T                -observation matrix
%%   T_matrix         -multi-dimensional array of all T
%%   K
%%   K_array
%%   J                -jacobian matrix
%%   J_array          -multi-dimensional array of jacobians
%%   Q_old            -instrumental covariance for rho,az,el
%%   Q_new            -instrumental covariance for position
%%   Q_array          -multi-dimensional array for all Q_new
%%   c_variance       -matrix used for running sums
%%   covariance        -covariance matrix
%%   s_vector         -matrix used for running sums
%%   S                -individual state vector input into summation
%%   P                -individual covariance input into summation
%%   state_vector     -estimated state vector
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

format long g
WGS84Data
global Rad
fid_input = fopen ('radardata1.dat', 'rt');
```

```

i=1;

% scans in data from input file from radar site or in this case the truth model
while ~feof(fid_input)

    sitlat=fscanf(fid_input,'%f',1);
    sitlon=fscanf(fid_input,'%f',1);
    sitalt=fscanf(fid_input,'%f',1);
    jd=fscanf(fid_input,'%f',1);
    rho =fscanf(fid_input,'%f',1);
    az =fscanf(fid_input,'%f',1);
    el =fscanf(fid_input,'%f',1);
    range_noise=fscanf(fid_input,'%f',1);
    az_noise=fscanf(fid_input,'%f',1);
    el_noise=fscanf(fid_input,'%f',1);

    % % for use when elevation and azimuth are given in degrees, range in meters, and time is in
    % year, daynumber, hour, min,sec

    %[sitlat,sitlon,sitalt,rho,az,el,jd,range_noise,az_noise,el_noise]=unit_converter(fid_input);

    % finds position vectors from radar data
    [R_ijk,local_sidereal_time] = position_finder(jd, sitlon, sitlat,sitalt, rho,az, el);

    % creates matrix of position vectors as row vectors
    R_ijk_matrix(i,:)= R_ijk;

    % creates matrix of julian days
    jdtime(i,:)= jd;

    % creates matrix of range values
    rho_matrix(i,:)= rho;

    % creates matrix of azimuth values
    az_matrix(i,:)= az;

    % creates matrix of elevation values
    el_matrix(i,:)= el;

    % creates matrix of lst values
    lst(i,:)=local_sidereal_time;

    i=i+1;
end

% determines size of Position matrix
[row,column] = size(R_ijk_matrix);

for ii=1:row
    % converts julian days into time since initial observation
    time = (jdtime(ii,:)-jdtime(151,1))*86400;

```

```

    % creates matrix of observation times
    timematrix(ii,:)=time;

end

%%%% Linear Least Squares Estimator

%% state transition matrix
% given the equations of motion in the truth model

for w =1:300
    t = timematrix(w,:);

    % defines sub-matrices
    sub0 = eye(3,3);
    sub1 = eye(3,3)*t;
    sub2 = eye(3,3)*(t^2)/2;
    sub3= zeros(3,3);
    sub4 = eye(3,3)*(t^3)/6;

    % defines phi_matrix
    phi_matrix = [sub0 sub1 sub2 sub4; sub3 sub0 sub1 sub2; sub3 sub3 sub0 sub1; sub3 sub3 sub3 sub0];

    phi_multi_array(:,w)=phi_matrix ;

end

[row2, column, depth]=size(phi_multi_array);

%% observation relation

% z = [R_ijk]';
% z = G(X,t)=(sub0 sub3 sub3)X
% G=[sub0 sub3 sub3]*X;

% linearization of G
% H =(partial G) / (partial X)
H = [sub0 sub3 sub3 sub3 ];

% T=H*phi
for f=1:depth
    T = H*phi_multi_array(:,f);
    T_matrix(:,f) = T;
end

% differential correction of intermediates

for s=1:row

```

```

K = [-cos(el_matrix(s))*cos(az_matrix(s)),...
      rho_matrix(s)*cos(el_matrix(s))*sin(az_matrix(s)),...
      rho_matrix(s)*sin(el_matrix(s))*cos(az_matrix(s)),...
      cos(el_matrix(s))*sin(az_matrix(s)),...
      rho_matrix(s)*cos(el_matrix(s))*cos(az_matrix(s)),...
      -rho_matrix(s)*sin(el_matrix(s))*sin(az_matrix(s));
      sin(el_matrix(s)) 0 rho_matrix(s)*cos(el_matrix(s))];
K_array(:, :, s)=K;

% rotation matrix
rot_ijk_sez = [sin(sitlat)*cos(lst(s)) sin(sitlat)*sin(lst(s)) -cos(sitlat);...
               -sin(lst(s)) cos(lst(s)) 0; cos(sitlat)*cos(lst(s)) cos(sitlat)*sin(lst(s))...
               sin(sitlat)];

J = inv(rot_ijk_sez)*K_array(:, :, s);
J_array(:, :, s)=J;

% instrumental covariance
Q_old = [range_noise^2 0 0; 0 az_noise^2 0; 0 0 el_noise^2];
Q_new = J_array(:, :, s)*Q_old*J_array(:, :, s)';
Q_array(:, :, s) = inv(Q_new);

end

c_variance=zeros(12,12);

% sums the values for the state covariance
for v =1:row
    P=T_matrix(:, :, v)'*Q_array(:, :, v)*T_matrix(:, :, v);
    c_variance= c_variance+P;
end

% inverts the covariance
covariance=inv(c_variance);

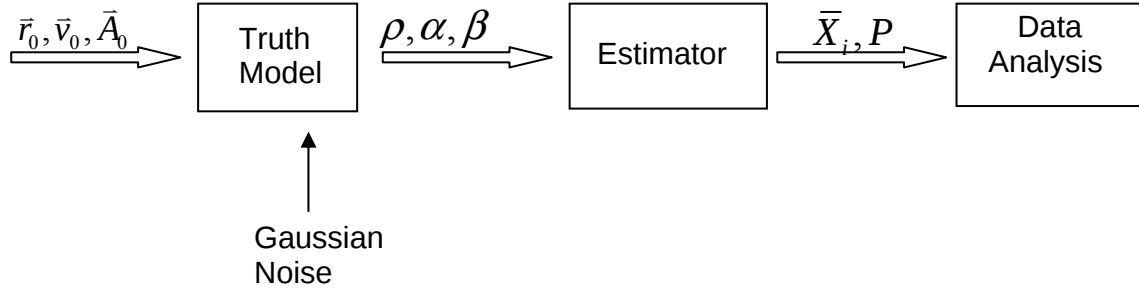
svector=zeros(12,1);

% sums the values for the system estimate at epoch time
for e= 1:row
    S=T_matrix(:, :, e)*Q_array(:, :, e)*R_ijk_matrix(e, :)' ;
    svector= svector+S;
end

% estimate of the state vector at epoch
state_vector=covariance*svector

```

Appendix D: Estimation Process



Bibliography

- Asher, T.A., Boden, D.G., and Tegtmeier, R.J. "Tethered Satellites: The Orbit Determination Problem and Missile Early Warning Systems," presented as AIAA Paper 88-4284 at the AIAA/AAS Astrodynamics Conference, Minneapolis, MN, August 15-17, 1988.
- Bate, Roger R., Muller, Donald D., and White, Jerry E. *Fundamentals of Astrodynamics*. New York: Dover Publications, Inc., 1971.
- Cicci, D.A., Cochran, J.E. Jr, Qualls, C., and Lovell, T.A. "Quick-Look Identification and Orbit Determination of a Tethered Satellite," *The Journal of the Astronautical Sciences*, Vol 50, No 3: 339-353 (July-September 2002).
- Cicci, D.A., Lovell, T.A., Qualls, C. "A Filtering Method for the Identification of a Tethered Satellite," *The Journal of the Astronautical Sciences*, Vol 49, No 2: 309-326 (April – June 2001).
- Cicci, D.A., Qualls, C., Lovell, T.A. "A Look at Tethered Satellite Identification Using Ridge-Type Estimation Methods," *Applied Mathematics and Computation*, Vol 119, Issues 2-3: 297-316 (April 2001).
- Galilei, Galileo. *Dialogues Concerning Two New Sciences*. Trans. Henry Crew and Alfonso de Salvio. New York: Dover Publications, Inc., 1914.
- Hahn, Alexander J. "The Pendulum Swings Again: A Mathematical Reassessment of Galileo's Experiments with Inclined Planes," *Archive for History of Exact Sciences*, Vol 56: 339-361 (2002).
- Hall, Garry L. *Compression of a Radar Track of a Near Earth Satellite into an Earth Centered Inertial State Vector Using Least Squares Differential Correction*. MS thesis, AFIT/GA/ENY/94M-1. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1994 (ADA278498).
- Sellers, Jerry J. *Satellite Tool Kit Astronautics Primer*. Analytical Graphics Inc. <http://www.agi.com/partners/edu/AstroPrimer/primer1.htm>. 10 May 2007.
- Sellers, Jerry J. *Understanding Space: An Introduction to Astronautics* (Second Edition). New York: McGraw-Hill Companies, Inc., 2000.

- Thomas, Paul. "Space Traffic Surveillance," *Space/Aeronautics*, 75-86 (Nov 1967).
- Vallado, David A. *Fundamentals of Astrodynamics and Applications* (Second Edition). Boston: Kluwer Academic Publishers, 2001.
- Weld, Leroy D. *Theory of Errors and Least Squares: A Textbook for College Students and Research Workers*. New York: Macmillan Company, 1916.
- Wiesel, William E. *Modern Astrodynamics*. Beavercreek, OH: Aphelion Press, 2003.
- Wiesel, William E. *Modern Orbit Determination*. Beavercreek, OH: Aphelion Press, 2003.
- Zaninetti, Lorenzo. "Gaussian Distribution Function. Image from website. 2002.
http://www.to.infn.it/~zaninett/javascript/distributions/gaussiana_b.html. 20 June 2007.

Vita

Second Lieutenant Sandra M. Rashash graduated from Jacksonville High School in Jacksonville, North Carolina. She entered undergraduate studies at the United States Air Force Academy in Colorado Springs, Colorado where she graduated with a Bachelor of Science degree in both Space Systems and Basic Sciences in May 2006. Upon graduation, she was commissioned as a Second Lieutenant in the United States Air Force. Following her commissioning, she entered the Graduate School of Engineering and Management, Air Force Institute of Technology. Upon Graduation she will be stationed at Columbus AFB, Mississippi as a student in Undergraduate Pilot Training.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 074-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 13-09-2007		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From – To) Oct 2006-Sept 2007	
4. TITLE AND SUBTITLE Radar Orbit Analysis Tool Using Least Squares Estimator				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Rashash, Sandra M., Second Lieutenant, USAF				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/ENY) 2950 Hobson Way WPAFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GSS/ENY/07-S01	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dr. Robert A. Racca HQ AFSPC/XPY 1150 Vandenberg St. Suite 1105 Peterson AFB, CO 80914 (719) 556-3714				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Most objects tracked in space follow a regular Keplerian orbit; unfortunately, non-Keplerian objects such as maneuvering satellites, tethered systems, and thrusting ballistic missiles are becoming more common. It is important to be able to distinguish between Keplerian and non-Keplerian objects due to the potential risk of a tethered satellite being mistaken for an object on re-entry. This research focused on creating a computer model that can detect the non-gravitational acceleration present in non-Keplerian orbits. A 3 rd order Taylor series expansion was used to model the dynamics and to produce simulated radar data. Linear least squares estimation was used to estimate the initial state of a space object with a state vector composed of position, velocity, acceleration, and its first derivative. Monte Carlo analysis was used to verify that the estimator was unbiased and representative of the uncertainty in the data. The Monte Carlo method detected non-gravitational acceleration as small as 1.12 cm/s ² ; however, a subsequent approach that analyzed the data sets individually only detected acceleration as small as 10.63 cm/s ² . At smaller magnitudes, the estimator was able to detect the presence of non-gravitational acceleration, but was ultimately unable to estimate the true value with statistical accuracy.					
15. SUBJECT TERMS Taylor Series, Least Squares Method, Satellite Tracking, Orbits, Orbit Estimation					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 100	19a. NAME OF RESPONSIBLE PERSON William E. Wiesel, Dr., (ENY)
REPORT U	ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) (937) 255-6565, ext 4312; e-mail: William.wiesel@afit.edu