Space Based Satellite Tracking and Characterization Utilizing Non-Imaging Passive Sensors

THESIS

Bradley R. Townsend, Captain, USA

AFIT/GA/ENY/08-M06

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

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, United States Army, Department of Defense, or the United States Government.

AFIT/GA/ENY/08-M06

Space Based Satellite Tracking and Characterization Utilizing Non-Imaging Passive Sensors

THESIS

Presented to the Faculty

Department of Aeronautical and Astronautical Engineering

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 in Astronautical Engineering

Bradley R. Townsend, B.S.M.E.

Captain, USA

March 2008

AFIT/GA/ENY/08-M06

Space Based Satellite Tracking and Characterization Utilizing Non-Imaging Passive Sensors

Bradley R. Townsend, B.S.M.E.

Captain, USA

Approved:

Dr. Ronald F. Tuttle, PhD (Chairman)	date
Lt Col Nathan A. Titus (Member)	date
Lt Col Kerry D. Hicks (Member)	date

Abstract

A technique is developed to determine the orbit of a sunlight illuminated satellite passing through the field-of-view of a sensor platform in a Highly Elliptical Orbit (HEO) and Geosynchronous orbit (GEO). The technique develops two different methods of initial orbit determination. The first relies on the Gauss initial orbit determination method to develop an estimate of the state from angular data. The second method relies on positional data of the target relative to the Earth's background to determine an estimate of the state. These estimates are then refined in a non-linear least squares routine. This estimate of the state is then used to identify the target from the Air Force Space Command satellite catalog.

It was found that the Gaussian initial orbit determination method produced reasonable estimates of the state given data sets larger than 30 seconds. The second method proved to be useful when only very limited data sets were available and had no limitations on data size. The least squares process was able to compensate for errors of as much as 5 degrees in the initial estimate of the state values. In all HEO test cases the estimate of the state vector provided enough information to identify the target satellite from the satellite catalog. In GEO test cases it proved impossible to identify other objects in GEO given the limited data available. The method developed also proved unreliable in identifying orbits with eccentricities greater than .35. Over 82 percent of the objects listed in the Air Force Space Command satellite catalog can be successfully identified in orbit using this method.

Acknowledgements

I would like to thank my wife for her patience and perseverance for the many hours I spent at and away from home working on the completion of this project. Her support is always instrumental to everything I do. I am also grateful for the support provided to me by Dr. Ron Tuttle from the first day that I arrived at AFIT and sought help concerning the development of a thesis topic that could combine Astrodynamics and MASINT. His support and advice was instrumental in my completion of this topic and of my success in completing the MASINT program. Lastly, I would like to thank LTC Nathan Titus and Dr. William Wiesel for always being available to assist me when I needed it and for their patience in providing assistance whenever I asked for it. Without their assistance I would still be struggling with my code.

Bradley R. Townsend

Table of Contents

Page

List of Figures

List of Tables

List of Symbols

Symbol		Page
$\vec{r}$	Position Vector of Target	5
$\vec{\rho}$	Slant Range	5
$\vec{R}$	Position Vector of Sensor	5
$\hat{L}$	Vector of direction cosines	5
t_0	Some initial time	5
a	Semi-major Axis	7
e	Eccentricity	7
$\vec{h}$	Axis normal to orbital plane	7
Ω	Right ascension of the ascending node	7
w	Argument of Perigee	8
v	True Anomaly	8
λ	An Angle	10
ϕ	An Angle	10
R_i	Component of position vector of the sensor	10
R_j	Component of position vector of the sensor	10
R_k	Component of position vector of the sensor	10
c_1	Constant	14
c_3	Constant	14
f	f series function	14
g	g series function	14
$\vec{r}_i$	Position vector at some time	14
$\vec{v}$	Velocity	14
$\vec{r}_{i\pm j}$	Position vector between times	14
μ	Gravitational parameter	14
τ	Change in time	15

Symbol		Page
$\|\vec{r}_2\|$	Position vector	15
t_1	Time constant	15
t_2	Time constant	15
t_3	Time constant	15
$\hat{L}_1$	Component of line of sight unit vector	15
$\hat{L}_2$	Component of line of sight unit vector	15
$\hat{L}_3$	Component of line of sight unit vector	15
$\vec{r}_{site1}$	Position vector component of observer	15
$\vec{r}_{site2}$	Position vector component of observer	15
$\vec{r}_{site3}$	Position vector component of observer	15
ρ_1	Component of slant range	15
ρ_2	Component of slant range	15
ρ_3	Component of slant range	15
c_2	Constant	16
$\vec{A}$	Intermediate vector in Gauss derivation used for simplification	17
$\vec{B}$	Intermediate vector in Gauss derivation used for simplification	17
$\vec{r}_{site}$	Position of observer	17
A_1^*	Variable defining a function used in Gauss for simplification	17
B_1^*	Variable defining a function used in Gauss for simplification	17
A_3^*	Variable defining a function used in Gauss for simplification	17
B_3^*	Variable defining a function used in Gauss for simplification	17
A_2^*	Variable defining a function used in Gauss for simplification	17
B_2^*	Variable defining a function used in Gauss for simplification	17
h_2	Constant used in Gauss unrelated to angular momentum	18
$\vec{D}$	Vector perpendicular to plane formed by position vectors of target	19

Symbol		Page
$\vec{N}$	Scaled Vector perpendicular to plane formed by position vectors of target	20
$\vec{S}$	Mathematically defined vector used for simplification	20
Δ	Angle relating latitude and azimuth	22
δ	Latitude at Epoch	22
λ_E	Longitude East of meridian	23
θ_{GMST}	Greenwich mean sidereal time	23
$\dot{\theta}$	Radians per second or mean motion	23
μ	Earth gravitational constant	23
R_{Earth}	Radius of Earth	23
h	Altitude	23
δx	Change in trajectory	25
x_0	True system state	25
z_0	True data	25
r	Residual	26
H_i	Observation matrix	26
Q	Covariance matrix	26
σ	Variance	27
$\Phi(t_i, t_0)$	State transistion matrix	27
i	Orbital inclination	28
$\hat{Z}$	Unit vector in ECI frame	30
$\tilde{\Phi}$	Matrix relating position in ECI to state	31
r_e	Distance of target from center of the Earth	31
T	Total data vector	33

List of Abbreviations

Abbreviation		Page
HEO	Highly Elliptical Orbit	1
GEO	Geosynchronous Orbit	1
AFSPC	Air Force Space Command	1
LEO	Low Earth Orbit	1
STK	Satellite Tool Kit	2
AGI	Analytical Graphics Incorporated	2
RAAN	Right Ascension of the Ascending Node	3
TLE	Two Line Element Set	8
ECI	Earth Centered Inertial Frame	9
SEZ	Topocentric Horizon Coordinate System	9
Az	Azimuth	9
El	Elevation	9
NASA	National Aeronautics and Space Administration	12
GSD	Ground Sample Distance	34
MEO	Medium Earth Orbit	52

Space Based Satellite Tracking and Characterization Utilizing Non-Imaging Passive Sensors

I. Introduction

1.1 Objective and Scope

Satellite based sensors looking down at the Earth's surface occasionally observe reflected light from an object passing through the image which is moving too fast relative to the background of the image to be located within the atmosphere. These objects are commonly called fastwalkers. This term refers to any orbital object seen passing through the field of view of an Earth observing sensor which is suspected of being in orbit. The objective of this thesis is to develop a method to determine the orbit of these objects from a sensor in a highly elliptical orbit (HEO) and geosynchronous orbit (GEO) and then match them to an object in the Air Force Space Command (AFSPC) catalog of satellites.

In order to conduct this analysis a few basic assumptions must be made. First, the sensor ephemerids are known exactly since the position of the observer is necessary to relate the observations to a location in space. Second, the targets are assumed to be non-thrusting bodies in low Earth orbit (LEO), meaning they are at or near the Earth's surface below 2000 km in altitude. This assumption is reasonable for a HEO satellite because the number of objects in a HEO orbit is very small, so very few non-LEO spacecraft are likely to appear in the HEO sensors field of view. This assumption is not valid for a GEO sensor. Therefore, we will need to differentiate between targets in LEO and GEO when using observations from a GEO platform. Lastly, since the objects are in a LEO orbit and the sensor field of view is limited, we will assume that the maximum amount of time for which a data track is visible is less than 4 minutes. Given this brief period of observational data, it will be assumed that orbital

perturbations will have no visible effect on the orbit. For this reason the two body equations of motion are sufficient to estimate the orbital element set.

The simulated data for this project was generated by the Satellite Tool Kit (STK) available from Analytical Graphics Incorporated (AGI). This software is currently among the most versatile available and is commonly used for orbital modeling within the Army and Air Force Space Community. Version 7 of STK was used in the generation of all orbital data.

1.2 Research

In his thesis Osedacz [9] states that a database of fastwalker observations exists with data from as far back as 1972. He uses this data from obtained ballistic missile early warning satellites in GEO to attempt to determine the orbit of these fastwalkers. His conclusion was that determining the orbit of other fastwalkers in GEO was impossible. This supported the conclusions of an earlier work conducted by Lem Wong on the inability to determine the orbit of these objects. The research conducted by Wong is no longer available, but according to Osedacz, the problem was found to be unsolvable in a GEO orbit. This conclusion stemmed from the fact that most of the fastwalkers were determined to also be in Geosynchronous orbit with the sensor. The result was an inability to determine the range to the target.

The two previously mentioned works dealing with fastwalkers focused on sensors in GEO orbit. In a recent paper published by Li, Guo, and Zhou at the Chinese National University of Defense Technology, a Kalman filter was used to successfully determine an estimate of the orbital elements of an object visible from a sensor in HEO orbit [7]. The Kalman filter needed over 30 minutes of tracking data to converge on a solution that was accurate to within 4 kilometers. This method was inapplicable to the central problem of this thesis, since a sensor in HEO orbit cannot normally observe an object in LEO orbit for that amount of time. Very little research exists using angles-only data from a space platform and almost nothing on the problem of fastwalker orbit determination.

1.3 Approach

In addition to the sensor ephemerids, it was assumed that the time of each observation, the azimuth and elevation to the target in the sensor frame, and the latitude and longitude of the target against the background of the Earth were available data. The general method of correlating observed tracks to entries in the Air Force Space Command catalog was to use one of two initial orbit determination techniques and then to use a least squares process to refine the estimate of the state vector. The catalog could then be screened for possible matches by successively applying filters based on the estimates of the elements in the state vector.

The state vector consisted of four elements. Three of which are necessary to identify the target. The elements of the state vector are: right ascension of the ascending node (RAAN), inclination of the orbit plane, an argument of latitude which is equal to the mean anomaly of a circular orbit and an estimate of the altitude of the target fastwalker. For the purposes of this research, both the node and the inclination were treated as constants. They were then used to search the catalog, assuming that the most recent catalog is used so that the change in RAAN is small.

The first initial orbit determination technique used the latitude and the longitude of the background image. Spherical trigonometry was then used to determine the state. The second initial orbit determination technique was the classic initial orbit determination technique developed by Gauss. This technique used the azimuth and elevation data to estimate the state vector.

Once an initial estimate of the state vector was obtained, a non-linear least squares process was applied to refine the estimate using the rest of the data points available in a given track. This refined estimate of the state was then used to search the catalog for the right ascension of the ascending node and inclination near the estimate of the state. Once a candidate was found, it could be propagated to the epoch time of the estimated state and it's argument of latitude compared to that of

the estimated state. A match would then allow us to confidently state that we had identified the orbital object.

II. Orbit Generation Equations and Methods

This chapter presents the equations necessary to develop an accurate model of an orbiting object for the purpose of determining its orbit. The element sets, coordinate frames and transformations used will be discussed in detail. This chapter will also focus on describing the problem geometry and develop two different initial orbit determination techniques. The first initial orbit determination technique that will be discussed is the classic Gauss method, which uses three measurements of azimuth and elevation in the sensor frame. The second technique discussed uses latitude and longitude data from the background of the image produced by the sensor. Descriptions of both techniques and their development are discussed in this chapter.

2.1 Problem Geometry

Understanding the geometry of the problem is fundamental to solving it. Both initial orbit determination methods rely on the same fundamental equation. If we let the vector $\vec{r}$ represent the position of the object with respect to the center of the Earth, $\vec{\rho}$ is the range of the target satellite from the sensor, and $\vec{R}$ is the position of the sensor then we can write by observation that

$$\vec{r} = \vec{\rho} + \vec{R} = \rho\hat{L} + \vec{R} \tag{2.1}$$

where $\hat{L}$ is a vector of direction cosines, as shown in Figure 2.1.

2.2 Element Sets

2.2.1 Classical Orbital Elements. To describe an orbit under the assumption of a two-body gravity model, six or more parameters are required in addition to an epoch time. For example, given some time, t_0, and the position and velocity vectors of a satellite, we can fully describe the current state of a satellite. However, due to the limited nature of the data obtained from our observations and the need to limit the number of values in our state vector when using non-linear least squares, the Classical Orbital Element set was chosen as the parameter set in this work. In this element set

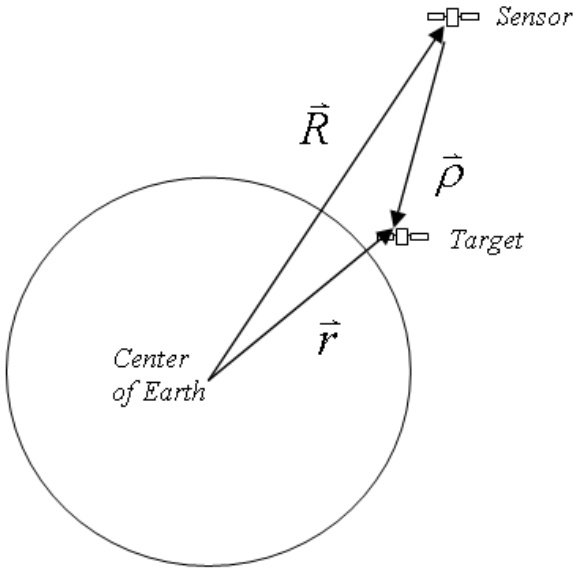

Figure 2.1: Problem Geometry

five of the six elements are constant and one varies periodically through a single orbit of the observed spacecraft. For this reason, the Classical Orbital Elements simplifies the process of developing an estimate of the satellite state. Although many sources define this orbital set, Wiesel [16] provides a solid development, which we will vary from only slightly here.

The classical orbital elements are:

a - The Semimajor Axis

e - Eccentricity

i - Inclination

Ω - Right Ascension of the Ascending Node

w - Argument of Perigee

v - True Anomaly

When describing the Classical Orbital Elements, it is necessary to first determine the shape of orbits around a body. An orbit around a body can be described as a conic section, which is generated by slicing a circular cone with a plane. Five types of conic

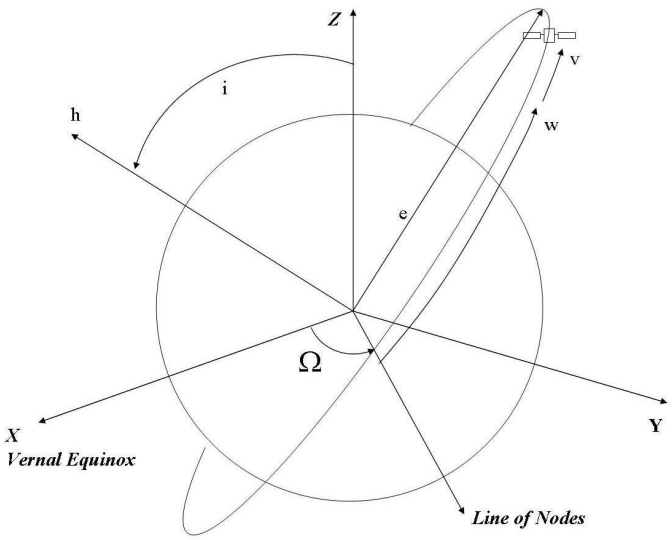

Figure 2.2: The Classical Orbital Elements

sections can be formed in this way, but we are only concerned with two, the circle and the ellipse. The semi major axis, a, is a constant which describes half of the major axis of an ellipse. The shape of the ellipse is defined by the eccentricity, e, which is a value in the interval $0 \leq e \leq 1$ and for a circle $e = 0$.

The inclination of an orbit refers to the tilt of the orbital plane. This angle is measured from the unit vector described by the Z axis to the angular momentum vector, $\vec{h}$, and varies from $0 \leq i \leq 180$ degrees, as shown in Figure 2.2. It can be defined in terms of these quantities by,

$$cos\,(i) = \frac{\hat{Z} \cdot \vec{h}}{\left|\hat{Z}\right| \left|\vec{h}\right|} \tag{2.2}$$

The right ascension of the ascending node, Ω, is the angle in the equatorial plane measured positively from the X axis to the point on the orbit where the satellite crosses the equator moving from South to North. Values for the ascending node range from $0 \leq \Omega \leq 360$ degrees. All inclined orbits also have a descending node where the

satellite crosses the equator moving from North to South. The line connecting these two points is called the Line of Nodes.

The final two elements of the set are interrelated. The Argument of Perigee, w, is the angle formed between the Line of Nodes and the point at which the orbit is closest to the Earth, called perigee. Values for the Argument of Perigee range from 0 to 360 degrees. The True Anomaly, v, is undefined for perfectly circular orbits, which the least squares model in this thesis assumes. For circular orbits this value is replaced with a measurement of the angle between the right ascension of the ascending node and the actual position of the satellite at some epoch time. For this thesis, we will refer to this as the Argument of Latitude. For non-circular orbits, the True Anomaly is not measured from the line of nodes but from the perigee point as pictured in Figure 2.2. Values for both also range from 0 to 360 degrees.

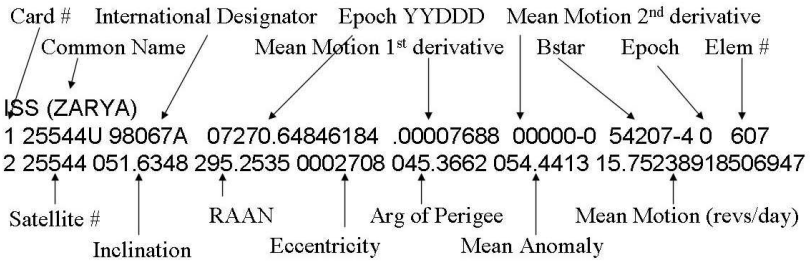

Figure 2.3: Two Line Element Set

2.2.2 Two Line Element Set. While the Classical Orbital Elements are used widely, the Air Force Space Command catalog of satellites is published using a set of elements commonly referred to as the Two Line Element set (TLE). An example of a TLE is shown in Figure 2.3. This system of describing orbits is a relic of the need, before the modern age of computing, to use punch cards in order to have the data read by computers. Space Command's catalog is the most up to date and reliable catalog of objects in orbit around the Earth. For this reason, it is used as the database for our search.

2.3 Coordinate Frames

Several Coordinate Frames are used for the purposes of generating data and describing the orbits of the objects within them. The transformations between these coordinate frames and their use is described.

2.3.1 Earth Centered Inertial Frame. The Earth Centered Inertial Frame (ECI) originates at the center of the Earth and is designated in Figure 2.2 by the letters XYZ. The fundamental plane of this coordinate frame is the Earth's Equator. The X axis points towards the vernal equinox. The Y axis is 90 degrees to the East of the X axis in the equatorial plane. The Z axis extends through the North Pole. Another set of labels that are commonly used for these are IJK.

2.3.2 Sensor Frame. The sensor frame is a slightly modified Topocentric Horizon Coordinate System (SEZ). This system is commonly used for ground based telescopes where the local horizon forms the fundamental plane. The S axis points South from the observer. The E axis points East and is undefined at the poles. The Z axis points outward from the center of the Earth. In the sensor frame the coordinate system is translated along the radial direction from the surface of the Earth to the center of the observing satellite. See Figure 2.1 for an example of the coordinate frame with reference to the problem geometry. Measurements taken in this frame are azimuth (Az) and elevation (El). Azimuth is measured positively clockwise from the -S axis and elevation is measured negatively down from the plane formed by the E and Z axis where the negative Z axis would be -90 degrees. The line-of-sight unit vector to the target in the sensor frame, $\hat{L}_h$, can then be defined as:

$$\begin{bmatrix} L_{hx} \\ L_{hy} \\ L_{hz} \end{bmatrix} = \begin{bmatrix} -cos\,(Az)\,cos\,(El) \\ sin\,(Az)\,cos\,(El) \\ sin\,(El) \end{bmatrix} \tag{2.3}$$

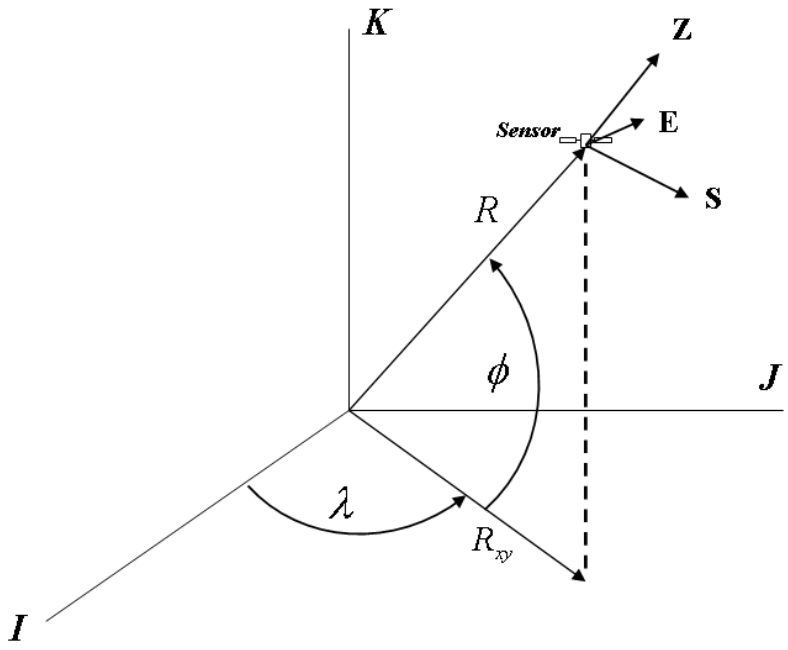

Figure 2.4: Right Ascension and Declination

2.3.3 Right Ascension and Declination Frame. When using the Gauss Initial Orbit Determination Method the measurements are taken as azimuth and elevation in the sensor frame and need to be transformed into right ascension and declination. In order to transform from the sensor frame to the right ascension-declination frame, the sensor frame unit vector found from equation (2.3) is needed. Before transformation between the two coordinate frames is possible, the two angles described by λ and ϕ in Figure 2.4 need to be found. Recall that we assumed we know the position of the sensor, $\vec{R}$. It's ECI components can be defined as R_i, R_j, R_k. Then a vector $\vec{R}_{ij}$ may be defined as

$$\vec{R}_{ij} = \begin{bmatrix} R_i \\ R_j \\ 0 \end{bmatrix} \tag{2.4}$$

The angle ϕ can then be found by:

$$\phi = acos\left(\frac{\vec{R}_{ij}\cdot\vec{R}}{\left\|\vec{R}_{ij}\right\|\left\|\vec{R}\right\|}\right) \tag{2.5}$$

We can then similarly use either the I axis unit vector or the i component of $\vec{R}$ to find λ by:

$$\lambda = acos\left(\frac{\vec{R_{ij}}\cdot\vec{X}}{\left\|\vec{R_{ij}}\right\|\left\|\vec{X}\right\|}\right) \tag{2.6}$$

Care must be taken when using this equation, it requires a quadrant check of the position of the sensor to determine if θ is greater than 180 degrees. With θ now defined it is then possible to transform the sensor frame measurement unit vector $\vec{L}_h$ into the right ascension-declination frame.

$$\begin{bmatrix} L_i \\ L_j \\ L_k \end{bmatrix} = \begin{bmatrix} sin\left(\phi\right)cos\left(\lambda\right) & -sin\left(\lambda\right) & cos\left(\phi\right)cos\left(\lambda\right) \\ sin\left(\phi\right)sin\left(\lambda\right) & cos\left(\lambda\right) & cos\left(\phi\right)sin\left(\lambda\right) \\ -cos\left(\phi\right) & 0 & sin\left(\phi\right) \end{bmatrix} \cdot \begin{bmatrix} L_{hi} \\ L_{hj} \\ L_{hk} \end{bmatrix} \tag{2.7}$$

2.3.4 Geocentric Latitude and Longitude. Measurements of latitude and longitude can be found using the background of images taken by the sensor platform. These measurements can then be used to approximate the position of a low earth orbit satellite's position near the Earth's surface. These measurements are typically given in terms of surface coordinates on the Earth's surface, referred to as geodetic coordinates. Since the Earth is not perfectly spherical and does in fact have a slightly elliptical shape it is necessary to convert these measurements to geocentric coordinates. First we need to define a flattening constant, F_{earth} which describes the elliptical shape.

$$F_{earth} = \frac{1}{298.257223563} \tag{2.8}$$

It is then possible to find the geocentric latitude given the geodetic latitude.

$$Lat_{geocentric} = atan\left(\left(1 - F_{earth}\right)^2 tan\left(Lat_{geodedic}\right)\right) \tag{2.9}$$

2.4 Initial Orbit Determination Techniques

The nature of the data provides no direct measurement of range. We are then limited to angles only initial orbit determination techniques by this. These techniques have classic roots with the methods of Gauss and Laplace with more recent work in the field having been done by R.H. Gooding [4, 5] and in a published technical paper by the National Aeronautics and Space Administration (NASA) [2]. Despite this, the number of unique methods developed since Gauss and Laplace is extremely limited. For over 150 years, there were many refinements of these two unique techniques developed. However, the next significant work was not published until 1959 by R.E. Briggs and J.W. Slowley [1], although mention is made by both B.G. Marsden [8] and L.K Kristensen [6] of a little known technique developed in 1939 by Vaisala for which I could find only broad descriptions. The Briggs and Slowley technique, relying on the recent development of computers, introduced an iterative method of angles only initial orbit determination. This method formed the basis for an entirely new method by P. Escobal [3] called double-r iteration. Most recently, Vallado [15] reviewed the three primary techniques of Gauss, Laplace, and Escobal iteration, recommending them as the current standard in initial orbit determination. This work considered only these three methods and R.H. Gooding's recent work in selecting a suitable initial orbit determination method.

These four techniques represent a broad range of approaches to solving the initial orbit determination problem, but we will focus only on their effectiveness in terms of the number and angular separation of the observations that is necessary for them to function. The Gooding technique is an iterative technique developed specifically for Earth orbiting satellites which relies on angular data developed over multiple revolutions. For our purposes, this method is impractical for the problem considered

in this thesis. A thorough development of the remaining techniques is developed by Escobal [3]. He states that the double-r iteration technique is preferable but it requires data spanning large arcs or a number of revolutions. The standard Laplacian technique is a classic technique optimized for determining the orbits of very distant heliocentric planetary orbits [3]. The Laplacian method is not considered robust or reliable enough for satellite observations [15]. Escobal states that the preferred method for short arcs is Gauss [3]. However, this recommendation is not unanimous. A vigorous condemnation was made of the Gauss technique by L.G. Taff [13], in which he cited its poor radius of convergence based on it's use of series functions but it has also been equally well defended by B.G. Marsden [8].

Two primary initial orbit determination techniques will be used. The first is the classic Gauss Technique which relies on three position vectors and their associated times. The Gibbs or Herrick-Gibbs technique is then used to determine the velocity vector at the central position and obtain a complete description of the orbit. The Gauss method can be modified to use either azimuth and elevation measurements in the sensor frame or geocentric latitude and longitude measurements obtained from the background of the image generated by the sensor. The second technique is less sophisticated and uses the same latitude and longitude measurements applied on a sphere to obtain a partial description of the orbit.

2.4.1 Gauss Angles-Only Initial Orbit Determination. This development and description of the Gauss technique follows that of Escobal [3] and Prussing and Conway [10]. Since the motion of the object occurs in a plane, it is possible to express the position vector at any time as a linear combination of the position vectors at any two other times. Therefore, it should be possible to determine a set of scalars a, b, c not all zero such that

$$a\vec{r}_1 + b\vec{r}_2 + c\vec{r}_3 = 0 \tag{2.10}$$

If we solve for the central date corresponding to $\vec{r}_2$ and denote the new coefficients by c_1 and c_3,

$$\vec{r}_2 = c_1\vec{r}_1 + c_3\vec{r}_3 \tag{2.11}$$

Expressions for the coefficients in the above equation can be found by taking the cross products of the equation with the first and third position vectors.

$$c_1 = \frac{|\vec{r}_2 \times \vec{r}_3|}{|\vec{r}_1 \times \vec{r}_3|} \tag{2.12}$$

$$c_3 = \frac{|\vec{r}_2 \times \vec{r}_1|}{|\vec{r}_3 \times \vec{r}_1|} \tag{2.13}$$

To solve for c_1 and c_3, the f series function and g series function are used. These functions allow us to express the position vector at some time, $\vec{r}_i$, in terms of the velocity, $\vec{v}$, and position, $\vec{r}_{i\pm j}$ at some other time.

$$\vec{r}_1 = f_1\vec{r}_2 + g_1\vec{v}_2 \; ; \; \vec{r}_3 = f_3\vec{r}_2 + g_3\vec{v}_2 \tag{2.14}$$

where f and g are:

$$f_3 = 1 - \frac{(t_3 - t_2)^2}{2}h_2 \; ; \; g_3 = (t_3 - t_2) - \frac{(t_3 - t_2)^3}{6}h_2 \tag{2.15}$$

$$f_1 = 1 - \frac{(t_1 - t_2)^2}{2}h_2 \; ; \; g_1 = (t_1 - t_2) - \frac{(t_1 - t_2)^3}{6}h_2 \tag{2.16}$$

where we can define h_2 using Earth's gravitational parameter μ,

$$h_2 \equiv \frac{\mu}{|\vec{r}_2|^3} \tag{2.17}$$

Now we can define a new time variable, τ, to simplify the time differences in equations (2.15) and (2.16).

$$\tau_1 \equiv t_3 - t_2 \; ; \; \tau_2 \equiv t_3 - t_1 \; ; \; \tau_3 \equiv t_2 - t_1 \tag{2.18}$$

If we combine equation (2.12) and equation (2.14) we get;

$$c_1 = \frac{|\vec{r}_2 \times \vec{r}_3|}{|\vec{r}_1 \times \vec{r}_3|} = \frac{|\vec{r}_2 \times (f_3\vec{r}_2 + g_3\vec{v}_2)|}{|(f_1\vec{r}_2 + g_1\vec{v}_2) \times (f_3\vec{r}_2 + g_3\vec{v}_2)|}$$

$$c_1 = \frac{g_3}{(f_1 g_3 - f_3 g_1)} \approx \frac{\tau_1}{\tau_2}\left[1 + \frac{(\tau_2^2 - \tau_1^2)}{6} h_2\right] \tag{2.19}$$

Similarly:

$$c_3 \approx \frac{\tau_3}{\tau_2}\left[1 + \frac{(\tau_2^2 - \tau_3^2)}{6} h_2\right] \tag{2.20}$$

Combining equation (2.1) with equation (2.11) we obtain:

$$\rho_2 \hat{L}_2 - c_1\rho_1 \hat{L}_1 - c_3\rho_3 \hat{L}_3 = c_1\vec{r}_{site1} + c_3\vec{r}_{site3} - \vec{r}_{site2} \tag{2.21}$$

where:

- c_1 and c_3 are functions of $|\vec{r}_2|$ and t_1,t_2,t_3.
- $\hat{L}_1$,$\hat{L}_2$,$\hat{L}_3$ are observationally determined from either the background latitude and longitude or the observed azimuth and elevation angles from the observer to the target.
- $\vec{r}_{site1}$,$\vec{r}_{site2}$,$\vec{r}_{site3}$ are the position vectors of the sensor platform with respect to the center of the Earth and are known.

Equation (2.21) represents a set of three scalar equations with the four unknowns ρ_1,ρ_2,ρ_3 and $|\vec{r}_2|$. Therefore one more independent equation must be added to create a system that can be solved.

$$|\vec{r}_2|^2 = \rho_2^2 + |\vec{r}_{site2}|^2 + 2\rho_2 \hat{L}_2 \cdot \vec{r}_{site2} \tag{2.22}$$

We can now rewrite equation (2.11) as:

$$c_1\vec{r}_1 + c_2\vec{r}_2 + c_3\vec{r}_3 = 0 \tag{2.23}$$

where c_2=-1 or in terms of equation (2.21),

$$c_2\rho_2\hat{L}_2 + c_1\rho_1\hat{L}_1 + c_3\rho_3\hat{L}_3 = -c_1\vec{r}_{site1} - c_3\vec{r}_{site3} - c_2\vec{r}_{site2} \equiv M \tag{2.24}$$

expanded in matrix form,

$$\begin{bmatrix} L_{1x} & L_{2x} & L_{3x} \\ L_{1y} & L_{2y} & L_{3y} \\ L_{1z} & L_{2z} & L_{3z} \end{bmatrix} \begin{bmatrix} c_1\rho_1 \\ c_2\rho_2 \\ c_3\rho_3 \end{bmatrix} = M = [L] \begin{bmatrix} c_1\rho_1 \\ c_2\rho_2 \\ c_3\rho_3 \end{bmatrix} \tag{2.25}$$

Therefore, we can separate the matrix of line-of-sight unit vectors by inverting it,

$$\begin{bmatrix} c_1\rho_1 \\ c_2\rho_2 \\ c_3\rho_3 \end{bmatrix} = [L]^{-1} M \tag{2.26}$$

where

$$[L]^{-1} \equiv [A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{2.27}$$

At this point we need to redefine (2.19) and (2.20) in terms of new variables,

$$c_1 = \frac{g_3}{f_1 g_3 - f3 g_1} \approx \frac{\tau_1}{\tau_2}\left[1 + \frac{(\tau_2^2 - \tau_1^2)}{6} h_2\right] = A_1 + B_1 h_2 \tag{2.28}$$

$$c_3 \approx \frac{\tau_3}{\tau_2}\left[1 + \frac{(\tau_2^2 - \tau_3^2)}{6} h_2\right] = A_3 + B_3 h_2 \tag{2.29}$$

where we then define the vectors $\vec{A}$ and $\vec{B}$ as:

$$\vec{A}^T = \begin{bmatrix} A_1 & -1 & A_3 \end{bmatrix} \; ; \; \vec{B}^T = \begin{bmatrix} B_1 & 0 & B_3 \end{bmatrix} \tag{2.30}$$

We can also define the site vector, $\vec{r}_{site}$, at all three times such that,

$$\begin{bmatrix} X & \vdots & Y & \vdots & Z \end{bmatrix}^T = \begin{bmatrix} \vec{r}_{site1x} & \vec{r}_{site2x} & \vec{r}_{site3x} \\ \vec{r}_{site1y} & \vec{r}_{site2y} & \vec{r}_{site3y} \\ \vec{r}_{site1z} & \vec{r}_{site2z} & \vec{r}_{site3z} \end{bmatrix} \tag{2.31}$$

If we solve for ρ_2 in equation (2.26) we can see that

$$\rho_2 = A_2^* + B_2^* h_2 \tag{2.32}$$

where

$$\begin{aligned} A_2^* &= a_{21}\vec{A}\cdot\vec{X} + a_{22}\vec{A}\cdot\vec{Y} + a_{23}\vec{A}\cdot\vec{Z}; \\ B_2^* &= a_{21}\vec{B}\cdot\vec{X} + a_{22}\vec{B}\cdot\vec{Y} + a_{23}\vec{B}\cdot\vec{Z} \end{aligned} \tag{2.33}$$

Similarly from (2.26) if we solve for ρ_1 and ρ_3 we find,

$$\rho_1 = \frac{A_1^* + B_1^* h_2}{c_1} \; ; \; \rho_3 = \frac{A_3^* + B_3^* h_2}{c_3} \tag{2.34}$$

where

A_1^*,B_1^*,A_3^*,B_3^* are similar in form to A_2^* and B_2^*

$$\begin{aligned} A_1^* &= -\left(a_{11}\vec{A}\cdot\vec{X} + a_{12}\vec{A}\cdot\vec{Y} + a_{13}\vec{A}\cdot\vec{Z}\right); \\ B_2^* &= -\left(a_{11}\vec{B}\cdot\vec{X} + a_{12}\vec{B}\cdot\vec{Y} + a_{13}\vec{B}\cdot\vec{Z}\right) \end{aligned} \tag{2.35}$$

$$\begin{aligned} A_3^* &= -\left(a_{31}\vec{A}\cdot\vec{X} + a_{32}\vec{A}\cdot\vec{Y} + a_{33}\vec{A}\cdot\vec{Z}\right); \\ B_3^* &= -\left(a_{31}\vec{B}\cdot\vec{X} + a_{32}\vec{B}\cdot\vec{Y} + a_{33}\vec{B}\cdot\vec{Z}\right) \end{aligned} \tag{2.36}$$

However equations (2.32) and (2.34) still contain the term h_2 defined earlier by (2.17) which is still unknown. So in order to determine the magnitude of the radius vector to our target at the central time, r_2, one final equation is needed.

$$r_2^2 = r_{site2}^2 + \left(A - 2^* + B_2^* h_2\right)^2 + 2\left(\hat{L}_2\cdot\vec{r}_{site2}\right)\left(A - 2^* + B_2^* h_2\right) \tag{2.37}$$

Or, in its more recognizable 8th order polynomial form,

$$\begin{aligned} &r_2^8 - r_2^6\left[r_{site2}^2 + A_2^{*2} + 2\left(\hat{L}_2\cdot\vec{r}_{site2}\right)A_2^*\right] \\ &-r_2^3\left[2A_2^*B_2^*\mu + 2\left(\hat{L}_2\cdot\vec{r}_{site2}\right)B_2^*\mu\right] - B_2^{*2}\mu^2 = 0 \end{aligned} \tag{2.38}$$

A key point in the process must be highlighted here. In most forms of the Gauss algorithm, the emphasis is placed on selecting the largest root of the polynomial as the correct value of the magnitude of the radius vector from equation (2.38). This is true when looking out from the origin of your inertial coordinate frame with $|\vec{r}_2| > |\vec{r}_{site2}|$. However, in this problem we are looking in towards the origin $|\vec{r}_2| < |\vec{r}_{site2}|$ and so root selection varies slightly. Selecting the largest root of the polynomial will return the position of the observing sensor. I found that the second largest positive real root is, in fact, the correct value for the magnitude of r_2. With that, it is then possible to use this method to find three position vectors for the target satellite.

2.4.2 Calculating The Velocity Vector. Using the Gauss method discussed in the previous section, it is possible to determine the position vector of the target satellite at three different times. In order to gain a complete understanding of an orbit in three dimensional space, the velocity vector at one of the three position vectors must also be known. To determine the satellite velocity vector there are two primary

techniques which are used depending on the spread of data, the Gibbs and Herrick-Gibbs methods. The Gibbs solution relies on the geometry of the problem to solve for the velocity vector and works best when the spread of data is greater than five degrees, whereas the Herrick-Gibbs method relies on a Taylor series expansion and works best when the spread of data is less than one degree. Between one and five degrees both methods work reasonably well [15] and for the purposes of this thesis four degrees was selected as the crossover point between the two methods. To check the separation between the position vectors for the purposes of determining the appropriate technique to use in determining the velocity vector the following equations were used.

$$cos\left(\alpha_{12}\right) = \frac{\vec{r}_1 \cdot \vec{r}_2}{|\vec{r}_1|\,|\vec{r}_2|} \tag{2.39}$$

$$cos\left(\alpha_{23}\right) = \frac{\vec{r}_2 \cdot \vec{r}_3}{|\vec{r}_2|\,|\vec{r}_3|} \tag{2.40}$$

By taking the sum of these two angles we can determine which method is more appropriate to the data to use.

2.4.3 Gibbs Method. As mentioned earlier, the Gibbs method relies on a geometric solution to solve for the velocity vector. The solution method presupposes that we know three, non-zero, coplanar position vectors, which represent three sequential times of the target satellite in its orbit. From the Gauss method we have the appropriate data on the satellites position and the associated sequential times. When using this method, it is important to know the order of the position vectors in time because it relies on several cross products in its solution. We begin by defining a vector, $\vec{D}$, as:

$$\vec{D} = \vec{r}_1 \times \vec{r}_2 + \vec{r}_2 \times \vec{r}_3 + \vec{r}_3 \times \vec{r}_1 \tag{2.41}$$

So $\vec{D}$ is a vector which is perpendicular to the plane formed by the three position vectors of the target satellite. We also need to define a second vector perpendicular to the plane, $\vec{N}$, as:

$$\vec{N} = r_1 (\vec{r}_2 \times \vec{r}_3) + r_2 (\vec{r}_3 \times \vec{r}_1) + r_3 (\vec{r}_1 \times \vec{r}_2) = \rho \vec{D} \tag{2.42}$$

If the position vectors of the target satellite are not in the same plane then the Gibbs method will fail. However, for non-maneuvering spacecraft, it is a reasonable assumption that the motion is planar. Observational errors can still cause the vectors to appear non-planar. A method of solving for the planar separation between the position vectors is:

$$\alpha_{coplanar} = 90^o - cos^{-1} \left(\frac{(\vec{r}_2 \times \vec{r}_3) \cdot \vec{r}_1}{|\vec{r}_2 \times \vec{r}_3| \, |\vec{r}_1|} \right) \tag{2.43}$$

Since the observational data collected from the focal plane array contains error, this value will never be exactly zero so it is necessary to choose a tolerance for this value. Given the quality of the data analyzed, a tolerance of less than one degree proved to be sufficient. Continuing on with the Gibbs method, we need to define one additional vector quantity before we can find the velocity vector at the central time, $\vec{S}$.

$$\vec{S} = r_1 (\vec{r}_2 - \vec{r}_3) + r_2 (\vec{r}_3 - \vec{r}_1) + r_3 (\vec{r}_1 - \vec{r}_2) \tag{2.44}$$

Finally, we can calculate the velocity vector at the central time and obtain a complete picture of the orbit at our selected epoch time.

$$\vec{v}_2 = \frac{1}{r_2} \sqrt{\frac{\mu}{ND}} \vec{D} \times \vec{r}_2 + \sqrt{\frac{\mu}{ND}} \vec{S} \tag{2.45}$$

2.4.4 Herrick-Gibbs Method. The second method which was used to solve for the velocity vector of the target satellite was the Herrick-Gibbs method. It is used when the vectors are too close to one another for the Gibbs method to work

successfully. This method also uses the three observation times associated with the three sequential position vectors used in the Gibbs method. As in the Gibbs method, it is necessary to conduct a coplanar check of the position vectors using equation (2.43) to determine the validity of the data for use. Before calculating the velocity the change in time between position vectors of the target can be defined as:

$$\begin{aligned} \Delta t_{31} &= t_3 - t_1 \\ \Delta t_{32} &= t_3 - t_2 \\ \Delta t_{21} &= t_2 - t_1 \end{aligned} \tag{2.46}$$

Now using a truncated Taylor series expansion we can solve for the velocity vector at the central epoch.

$$\begin{aligned} \vec{v}_2 = -\Delta t_{32}\left(\frac{1}{\Delta_{21}\Delta t_{31}} + \frac{\mu}{12r_1^3}\right)\vec{r}_1 + (\Delta t_{32} - \Delta t_{21})\left(\frac{1}{\Delta_{21}\Delta t_{32}} + \frac{\mu}{12r_2^3}\right)\vec{r}_2 \\ +\Delta t_{21}\left(\frac{1}{\Delta_{32}\Delta t_{31}} + \frac{\mu}{12r_3^3}\right)\vec{r}_3 \end{aligned} \tag{2.47}$$

The final result of the Gauss method is an estimate of the position and velocity vectors of the target satellite. These values must be converted to the classical orbital elements used in the state vector during the non-linear least squares process that will be discussed in Chapter 3. All of the classical orbital elements are calculated from the estimate provided by Gauss, but only the four values in the state vector are kept. The full element set calculated by Gauss only becomes useful when the target is not identified in the AFSPC catalog as discussed in Chapter 4.

2.4.5 Spherical Trigonometry. The second initial orbit determination technique utilizes a simpler approach to obtain an incomplete picture of the targets orbital elements, but it is enough to identify it within the catalog. This method assumes that the observed satellite is at or near the Earth's surface in a circular orbit. This assumption does introduce error, but it does not significantly affect the information

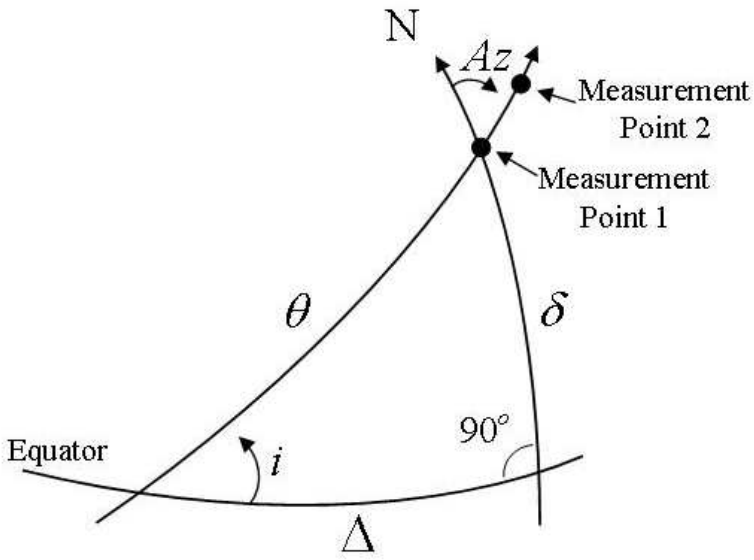

Figure 2.5: Initial Orbit Determination Using Spherical Trigonometry

about the orbital plane, which is defined by right ascension of the ascending node (RAAN), Ω , and inclination. It is these two elements that are initially used to find a match in the AFSPC catalog. Using some simple spherical trigonometry it is possible to obtain enough information to begin the least squares iteration, see Figure 2.5. This method relies on knowing the location of the reflected signature from a target relative to the background of the Earth. Typically this is how fastwalkers are seen on imagery taken from a platform. After being processed the imagery is fitted with a latitude and longitude grid. Using the known location of the target fastwalker at two closely related times we can find the angle between true North and the orbit plane, the azimuth, by:

$$Az = atan\left(\frac{\Delta Latitude}{\Delta Longitude}\right) \tag{2.48}$$

It is then possible to find the angle Δ in Figure 2.5 if we define latitude at epoch as δ through,

$$\Delta = atan\left(sin\left(\delta\right) tan\left(Az\right)\right) \tag{2.49}$$

We can use Δ to find the RAAN. If we define λ_E as the longitude East of the meridian and θ_{GMST} as the Greenwich mean sidereal time, then the the node can be found using equation(2.50).

$$\Omega = \lambda_E + \theta_{GMST} - \Delta \tag{2.50}$$

The argument of latitude, θ , can then be found using the latitude.

$$\theta = acos\left(cos\left(\Delta\right) cos\left(\delta\right)\right) \tag{2.51}$$

The inclination of the orbit is then found easily.

$$i = \frac{sin\left(\delta\right)}{sin\left(\theta\right)} \tag{2.52}$$

If we assume a small change in time between observations we can also find $\dot{\theta}$ by,

$$\dot{\theta} \cong \frac{\sqrt{\left(\Delta Longitude\right)^2 cos^2\left(Lat\right) + \left(\Delta Latitude\right)^2}}{t_2 - t_1} \tag{2.53}$$

Finally using $\dot{\theta}$ we can obtain an estimate of the satellite's altitude above the Earth's surface. If we define μ as the gravitational constant of Earth, R_{Earth} as the radius of the Earth, and h as the altitude we can find altitude using,

$$\dot{\theta} = \sqrt{\frac{\mu}{\left(R_{Earth} + h\right)^3}} \tag{2.54}$$

which is related to the TLE set value of revolutions per day by Kepler's third law.

III. Non-Linear Least Squares

The method used to refine the initial state vector, found through either the Gauss initial orbit determination or using the spherical trigonometry technique, is non-linear least squares. The initial orbit determination techniques discussed earlier are deterministic in their approach. Both methods use an equivalent amount of known values to solve for a set of equivalent unknowns. What both of them lack is an ability to account for less than perfect data and for additional observations outside those necessary to obtain the unknowns. Non-linear least squares allows for the introduction of noise into the data as well as for the incorporation of additional observations to compensate for that noise.

Least squares dates back to the work of Legendre and Gauss at the beginning of the 19th century [14]. Legendre first published a method recognizing the importance of observational errors and proposed the least squares solution as a method for minimizing those errors in 1806. Gauss later published a probabilistic method for using least squares in 1809, but claimed to have developed and used it as early as 1795 and so is given the credit for having originated the method. Regardless of whoever originated the idea, it has undergone much refinement over the two centuries since it was first developed.

3.1 General Method of Non-Linear Least Squares

Non-linear least squares was developed by Gauss to model the non-linearity of real world systems. For the purposes of this thesis we have a non-linear dynamic system and non-linear observation geometry which requires the use of this method.

Since the system dynamics are available as an explicit solution

$$x(t) = h(x(t_0), t) \tag{3.1}$$

which is a function of time and initial conditions, we can linearize the dynamics in the following manner.

$$\delta x\left(t\right)=\Phi\left(t,t_{0}\right)\delta x\left(t_{0}\right) \tag{3.2}$$

Where the quantity δx will be small. Before the dynamics can be successfully linearized an initial trajectory must be obtained to linearize about. The 'true' value of the system state, x_0, is unobtainable so we use a reference trajectory x_{ref}, which is hopefully close to the true system state. The reference trajectory is obtained from using either the Gauss initial orbit determination technique or by utilizing the spherical trigonometry technique.

The condition of the available observational data must also be considered. This data is a non-linear function of the system state at the current time and the observation geometry and can be written as,

$$z_{i}\left(t_{i}\right)=G\left(x\left(t_{i}\right),t_{i}\right) \tag{3.3}$$

The observation relation, described by equation (3.3), can be linearized in a manner similar to equation (3.2). Since our sensor obtains very accurate observations, but not perfect observations of the true state, it does not produce perfect data, z_0. So we can expect that as the error in the observations goes to zero the error in the state also approaches zero. If the error is small enough than we can linearize the difference. If we first describe the difference between the true state and calculated state as,

$$x=x_{0}+\delta x \tag{3.4}$$

we can then describe the difference between the true error and the actual data.

$$\begin{aligned} e &= z-z_{0}=G\left(x,t\right)-G\left(x_{0},t\right) \\ &= G\left(x_{0}+\delta x,t\right)-G\left(x_{0},t\right) \\ &\approx \frac{\partial G}{\partial x}\delta x\left(t\right) \end{aligned} \tag{3.5}$$

Since the observations are not perfect we can equate the true error to the residual in the data, r. The residual has two parts. The first part comes from the error in the reference trajectory from our initial orbit determination method and the second part from the errors in the observations. Using our reference trajectory we can compute the observations that would result from it.

$$r_i = z_i - G\left(x_{ref}\left(t_i\right), t_i\right) \tag{3.6}$$

The residual vector, r_i, is a measure of the difference between the observation and the expected observation. It is related to the change in the reference trajectory, δx, which corrects the reference trajectory to a value close to the true trajectory of the observed target. If we first define a matrix, H_i, which relates the error in the state observation to the error in the reference trajectory as

$$H_i\left(t_i\right) \equiv \frac{\partial G}{\partial x}\left(x_{ref}\left(t_i\right), t_i\right) \tag{3.7}$$

The residuals in the observations can be related to the correction, δx by

$$\begin{aligned} r_i &\approx H_i \delta x\left(t_i\right) = H_i \Phi\left(t, t_0\right) \delta x\left(t_0\right) \\ &= T_i \delta x\left(t_0\right) \end{aligned} \tag{3.8}$$

Finally, a covariance matrix, Q, which compensates for unit compatibility and unit conversions must be defined. Assuming each measurement taken by the sensor is independent, Q can be written as

$$Q = \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma_N^2 \end{pmatrix} \tag{3.9}$$

where σ is the statistical variance in each observation. Combining, we can write that the correction to the reference trajectory can be written as

$$\delta x\left(t_0\right) = \left(T^T Q^{-1} T\right)^{-1} T^T Q^{-1} r \tag{3.10}$$

The associated covariance can then be written.

$$P_{\delta x} = \left(T^T Q^{-1} T\right)^{-1} \tag{3.11}$$

Using equation (3.10), we can then obtain an estimate of the true trajectory.

$$\bar{x}\left(t_0\right) = x_{ref}\left(t_0\right) + \delta x\left(t_0\right) \tag{3.12}$$

A simple modifcation of the algorithm in Wiesel [17] can best describe the entire method.

1. At each observation time t_i, propagate the calculated state vector at that time to the epoch time using the state transition matrix, $\Phi\left(t_i, t_0\right)$.
2. Calculate H_i for the current observation as well as the residual $r_i = z_i - G\left(x\right)$.
3. Calculate $H_i \Phi$.
4. Add this T_i to previous ones to build the matrix $T_i^T Q_i^{-1} T_i$ and the vector $T_i^T Q_i^{-1} r_i$ which should be growing larger in dimension with each data point. Repeat these steps for all data points at the epoch time.
5. Once complete with all data points calculate the covariance $P_{\delta x}$ by using equation (3.11).
6. Then calculate the correction to the reference trajectory using equation (3.10).
7. Correct the reference trajectory using equation (3.12).

8. Determine convergence after the second iteration based on the size of the correction. If the process has not converged continue to repeat the process with the newly corrected reference trajectory until it does so.

Each of the several functions described here will now be developed in detail.

3.2 The State Transition Matrix

In order to improve upon the deterministic reference orbit found through the initial orbit determination techniques, we must utilize a larger quantity of data. Least squares allows us to do this but to do so our observations must be propagated to a single epoch time. Let us begin by defining the state vector

$$\bar{X} = \begin{bmatrix} \Omega \\ i \\ \theta \\ \dot{\theta} \end{bmatrix} \tag{3.13}$$

Where, as before, Ω is the right ascension of the ascending node, i is the inclination of the orbit and θ is the argument of latitude, which is equal to the mean anomaly in a perfectly circular orbit. The value $\dot{\theta}$ can still be used to find an approximation of altitude. Since all the values are constants except θ, which is a function of time, we can define

$$\dot{\bar{X}} = F\left(\bar{X}\right) = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \\ 0 \end{bmatrix} \tag{3.14}$$

The state transition matrix is then simply,

$$\Phi\left(t, t_0\right) = \frac{\partial F}{\partial \bar{X}_0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

3.3 The Observation Function, G

This thesis does not use actual platform observations so they must be generated separately. The observations are azimuth and elevation as measured from the sensor platform in the sensor topocentric frame. Using a orbit modeling software called Satellite Tool Kit developed by Analytical Graphics Incorporated we can obtain the position of the target $\vec{r}$ and the sensor $\vec{R}$ in the Earth Centered Inertial frame. The range between them can be found with

$$\vec{\rho_{ECI}} = \vec{r} - \vec{R} \tag{3.16}$$

This range then needs to be rotated into the sensor frame in order to simulate our observations. The rotation matrix is given by

$$R_{ECI \rightarrow SEZ} = \begin{bmatrix} \hat{S}^T \\ \hat{E}^T \\ \hat{Z}^T \end{bmatrix} \tag{3.17}$$

where we define

$$\hat{Z} = \frac{\vec{R}}{|R|} \tag{3.18}$$

$$\hat{E} = \frac{\hat{k} \times \vec{R}}{|k \times R|} \tag{3.19}$$

$$\hat{S} = \hat{E} \times \hat{Z} \tag{3.20}$$

and $\hat{k} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$ is the unit vector in the $\hat{Z}$ direction in the ECI frame. The range in the SEZ frame can then be found by

$$\vec{\rho}_{SEZ} = R_{ECI \to SEZ} \vec{\rho}_{ECI} \tag{3.21}$$

Then if we separate out the components of the range in the SEZ frame as

$$\vec{\rho}_{SEZ} = \begin{bmatrix} \rho_S \\ \rho_E \\ \rho_Z \end{bmatrix} \tag{3.22}$$

we can write our observations of azimuth and elevation in terms of what we wish to find, the range between our target and sensor.

$$Az = atan\left(\frac{\rho_E}{-\rho_S}\right) \tag{3.23}$$

$$El = asin\left(\frac{\rho_Z}{\sqrt{\rho_S^2 + \rho_E^2 + \rho_Z^2}}\right) \tag{3.24}$$

3.4 Measurement Jacobi Matrix, H

From the derivation of the general method of non-linear least squares, in particular equation (3.7), we can see the need for a relationship between the error in the state and the error in the reference trajectory. We can define this relationship with regard to the observations (obs) as

$$H = \frac{\partial obs}{\partial \vec{r}_{ECI}} = \frac{\partial \vec{obs}}{\partial \vec{\rho}_{SEZ}} \frac{\partial \vec{\rho}_{SEZ}}{\partial \vec{\rho}_{ECI}} \frac{\partial \vec{\rho}_{ECI}}{\partial \vec{r}_{ECI}} \tag{3.25}$$

where

$$\frac{\partial \vec{\rho}_{SEZ}}{\partial \vec{\rho}_{ECI}} = R_{ECI \to SEZ} \tag{3.26}$$

which is the rotation matrix we found previously in equation (3.17) and

$$\frac{\partial \vec{\rho}_{ECI}}{\partial \vec{r}_{ECI}} = I \tag{3.27}$$

is simply equal to an identity matrix. The challenge here is the first term which cannot be directly related but must be further broken into,

$$\frac{\partial \vec{obs}}{\partial \vec{\rho}_{SEZ}} = \frac{\partial obs}{\partial \vec{r}_{ECI}} \frac{\partial \vec{r}_{ECI}}{\partial \bar{X}} \tag{3.28}$$

In order to find the relationship between the position vector in the ECI frame and the state, which we will call, $\tilde{\Phi}$, we first need to define a rotation between the ECI frame and the orbital frame in terms of the state vector. If we first define

$$\vec{r}_{pqw} = \begin{bmatrix} r_e cos\left(\theta\right) \\ r_e sin\left(\theta\right) \\ 0 \end{bmatrix} \tag{3.29}$$

where with μ being the gravitational constant of Earth and r_e is the calculated distance of the target from its center.

$$r_e = \sqrt[3]{\frac{\mu}{\dot{\theta}^2}} \tag{3.30}$$

We can than find $\vec{r}_{ECI}\left(\bar{X}\right)$ by rotating about the node and inclination.

$$\vec{r}_{ECI}\left(\bar{X}\right) = R_3\left(-\Omega\right) R_1\left(-i\right) \vec{r}_{pqw} \tag{3.31}$$

In explicit form

$$\vec{r}_{ECI}\left(\bar{X}\right) = \begin{bmatrix} cos\left(\Omega\right) & sin\left(\Omega\right) & 0 \\ -sin\left(\Omega\right) & cos\left(\Omega\right) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\left(i\right) & sin\left(i\right) \\ 0 & -sin\left(i\right) & cos\left(i\right) \end{bmatrix} \begin{bmatrix} cos\left(\theta\right) \\ sin\left(\theta\right) \\ 0 \end{bmatrix} \sqrt[3]{\frac{\mu}{\dot{\theta}^2}} \tag{3.32}$$

the matrix $\tilde{\Phi}$ is then

$$\frac{\partial \vec{r}_{ECI}}{\partial \bar{X}} = \tilde{\Phi}_{3x4} = \left[\begin{array}{c|c|c|c} \frac{\partial \vec{r}}{\partial \Omega} & \frac{\partial \vec{r}}{\partial I} & \frac{\partial \vec{r}}{\partial \theta} & \frac{\partial \vec{r}}{\partial \dot{\theta}} \end{array} \right]_{3x4} \tag{3.33}$$

$$\frac{\partial \vec{r}}{\partial \Omega} = \begin{bmatrix} r_e \left(-sin\left(\Omega\right) cos\left(\theta\right) - cos\left(\Omega\right) cos\left(i\right) sin\left(\theta\right)\right) \\ r_e \left(cos\left(\Omega\right) cos\left(\theta\right) - sin\left(\Omega\right) cos\left(i\right) sin\left(\theta\right)\right) \\ 0 \end{bmatrix} \tag{3.34}$$

$$\frac{\partial \vec{r}}{\partial i} = \begin{bmatrix} r_e sin\left(\Omega\right) sin\left(i\right) sin\left(\theta\right) \\ -r_e cos\left(\Omega\right) sin\left(i\right) sin\left(\theta\right) \\ r_e cos\left(i\right) sin\left(\theta\right) \end{bmatrix} \tag{3.35}$$

$$\frac{\partial \vec{r}}{\partial \theta} = \begin{bmatrix} r_e \left(cos\left(\Omega\right) sin\left(\theta\right) - sin\left(\Omega\right) cos\left(i\right) cos\left(\theta\right)\right) \\ r_e \left(-sin\left(\Omega\right) sin\left(\theta\right) + cos\left(\Omega\right) cos\left(i\right) cos\left(\theta\right)\right) \\ r_e sin\left(i\right) cos\left(\theta\right) \end{bmatrix} \tag{3.36}$$

$$\frac{\partial \vec{r}}{\partial \dot{\theta}} = \frac{\frac{-2}{3}}{\left(\frac{u}{\dot{\theta}^2}\right)^{\frac{2}{3}}} \cdot \frac{u}{\dot{\theta}^3} \begin{bmatrix} cos\left(\Omega\right) cos\left(\theta\right) - sin\left(\Omega\right) cos\left(i\right) sin\left(\theta\right) \\ sin\left(\Omega\right) cos\left(\theta\right) + cos\left(\Omega\right) cos\left(i\right) sin\left(\theta\right) \\ sin\left(i\right) sin\left(\theta\right) \end{bmatrix} \tag{3.37}$$

Finally, we need the function $\frac{\partial obs}{\partial \vec{r}_{ECI}}$, which we will define as $\tilde{H} \cdot R_{ECI \rightarrow SEZ}$.

$$\tilde{H} = \left[\begin{array}{c|c|c} H_1 & H_2 & H_3 \end{array} \right]_{2x3} \cdot R_{ECI \rightarrow SEZ} \tag{3.38}$$

$$H_1 = \begin{bmatrix} \frac{\rho_E}{\rho_E^2 + \rho_S^2} \\ \frac{-\rho_Z}{\left(\rho_E^2 + \rho_S^2 + \rho_Z^2\right)^{\frac{3}{2}}} \cdot \frac{\frac{\rho_S}{1-\rho_Z^2}}{\left(\rho_E^2 + \rho_S^2 + \rho_Z^2\right)^{\frac{1}{2}}} \end{bmatrix} \tag{3.39}$$

$$H_2 = \begin{bmatrix} \frac{-\rho_S}{\rho_E^2 + \rho_S^2} \\ \frac{-\rho_Z}{\left(\rho_E^2 + \rho_S^2 + \rho_Z^2\right)^{\frac{3}{2}}} \cdot \frac{\frac{\rho_E}{1-\rho_Z^2}}{\left(\rho_E^2 + \rho_S^2 + \rho_Z^2\right)^{\frac{1}{2}}} \end{bmatrix} \tag{3.40}$$

$$H_3 = \begin{bmatrix} 0 \\ \frac{\rho_E^2+\rho_S^2}{\left(\rho_E^2+\rho_S^2+\rho_Z^2\right)^{\frac{3}{2}}} \\ \frac{\rho_E^2+\rho_S^2}{\left(\rho_E^2+\rho_S^2+\rho_Z^2\right)^{\frac{1}{2}}} \end{bmatrix} \tag{3.41}$$

We can then find and build T.

$$T = \tilde{H} \cdot \tilde{\Phi} \cdot \Phi \tag{3.42}$$

This completes the functional derivation of the values needed to execute the non-linear least squares algorithm. The actual implementation is more complicated and as a reference the code is included in Appendix 2.

IV. Data and Results

4.1 Data Accuracy and Precision

An important factor to consider in the development of any model for a real world sensor is the accuracy of its measurements. In order to investigate the affect of error in the measuring sensor on the ability of the method being developed to accurately identify the target satellite, a range of reasonable values for this error must be determined. The primary source of noise in the data was assumed to be the resolution of the focal plane array of the sensor satellite, known as ground sample distance (GSD).

To determine the effects of GSD on the data the position of the sensor is assumed to be known to a level of precision, which for our purposes, we can assume to be perfectly known. Another possible source of error that is assumed to be zero is known as jitter, this refers to the pointing accuracy in the sensor. It exists in any man-made optical platform but is small compared to the error in the known position of the sensor, which is considered to be perfectly known. Since additional sources of error are smaller or have no effect on the method or collection of data, all of these error sources including dead pixels and dark current are ignored. Finally, any error generated by sub pixel processing, centroiding or any other software method of increasing the resolution of the focal plane array is considered to be small, and other than changing the value of the GSD, is ignored.

Any satellite in a highly elliptical orbit spends most of its orbital period near the apogee of its orbit. We will therefore calculate the range of error in the measurements of azimuth and elevation from an altitude at or near apogee for a typical HEO orbit, 36,000 kilometers. The range of reasonable ground sample distances that will be considered for investigation is one kilometer to one meter. Ground sample distance is simply the area on the ground that a sensor sees from orbit for each pixel on the focal plane array. So a ground sample distance of one meter means that each pixel on the focal plane array of the sensor sees a one meter by one meter square of ground. While many civilian sensors working in the visible light wavelengths regularly achieve

ground sample distances of one meter or better, the technology of systems operating outside the visible wavelengths lags far behind. Calculating the range of this error is found simply by

$$Error = tan^{-1}\left(\frac{GSD}{Altitude}\right) \times 2 \tag{4.1}$$

Using this calculation the magnitude of the errors induced by GSD is seen to be quite small from 3.183e-3 to 3e-6 degrees. This level of error is unrealistically small for real time data. Post processing of data can significantly improve the quality of the data to an unknown degree, but may be unnecessary since the methods developed are valid with as much as 1e-2 degrees of error in the data.

4.2 Test Case Selection

The selection of test cases and the validation of the method was done by analyzing the satellite catalog for inclination and right ascension of the ascending node (RAAN). The third value being effectively solved for in the least squares process was the argument of latitude. This value is a measure of where the satellite is in its orbit which is constantly changing and so is not used in an initial search of the catalog for the target satellite. To use the argument of latitude, each item in the catalog would need to be propagated to the epoch time being used. This is simply not a realistic method of conducting an initial search for the target satellite, but can yield useful results once the field of candidate targets is narrowed significantly. Five test cases were selected using satellites from various bands of inclination. Using Figure 4.2 we can clearly see bands of crowded inclination and begin a selection of useful test cases. This plot represents the entire catalog of satellites as generated by Air Force Space Command as of 28 September 2007 and includes 11,231 objects in orbit ranging from fully functional satellites to small pieces of debris.

For the sensor platform in HEO orbit, objects below 40 degrees of inclination do not enter the useful field of view to any significant degree, so we will focus on objects above this inclination for this sensor orbit. The three largest concentrations

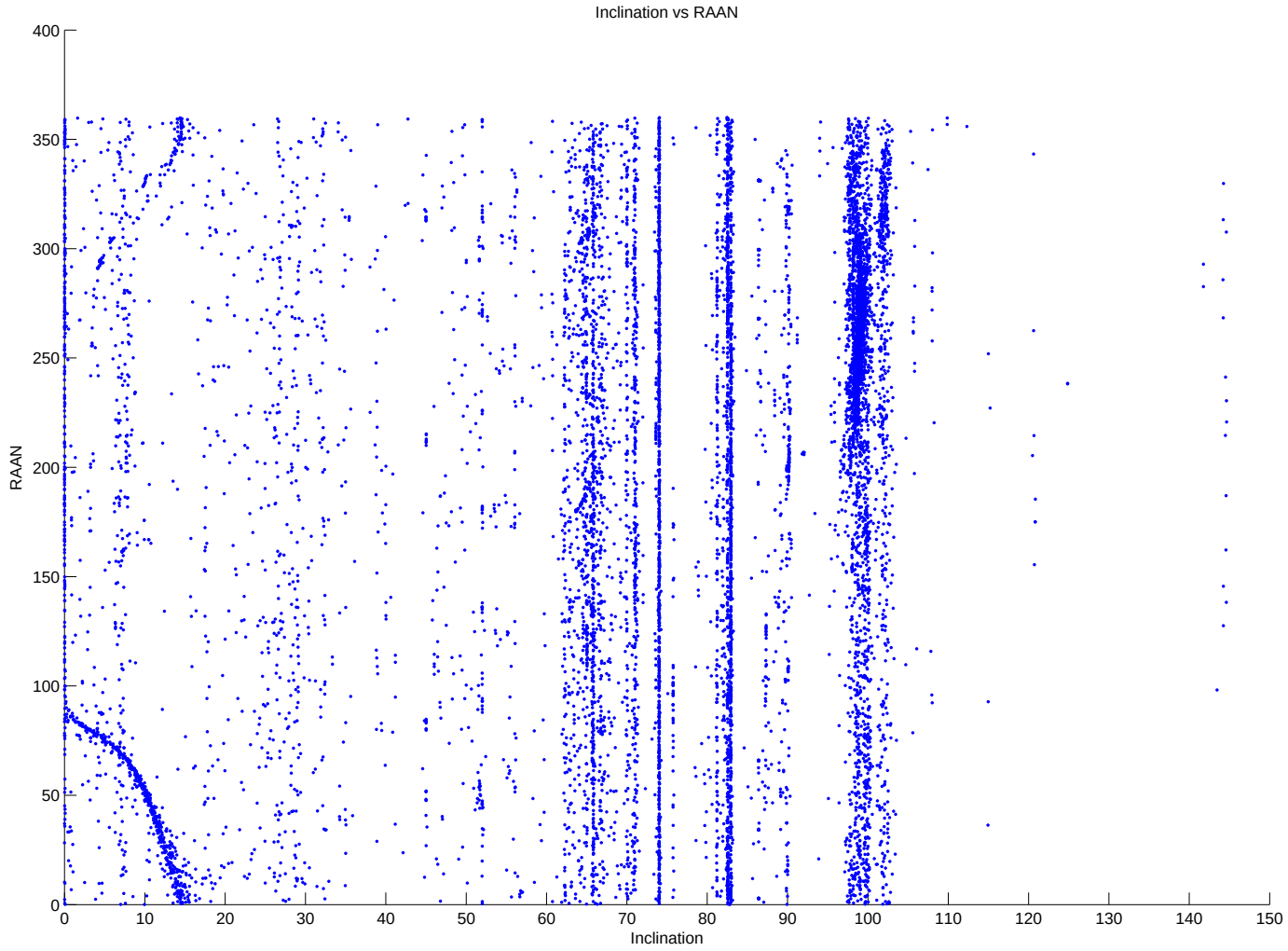

Figure 4.1: Distribution of Space Objects in the AFSPC by Right Ascension of the Ascending Node and Inclination

of satellites can be seen at 83, 74 and 98 degrees of inclination, so a test case within each of these bands was selected. Finally, one test case was selected within the 60-70 degree inclination region and one in the 40-60 degree region. The two line element sets of the selected test cases for the HEO sensor are:

Test Case 1 HEO: 98 degrees inclination LEO
THOR BURNER 2 R/B
1 03271U 68042B 07269.97501982 .00000015 00000-0 29790-4 0 2392
2 03271 098.8574 065.9071 0051791 293.5969 065.9783 14.15994498 29159

Test Case 2 HEO: 82 degrees inclination LEO
COSMOS 2416
1 28908U 05048A 07270.55302940 .00000017 00000-0 40682-4 0 5277
2 28908 082.4733 359.0931 0005782 305.7944 054.2602 12.55368842 80898

Test Case 3 HEO: 74 degrees inclination LEO
SL-8 R/B
1 22676U 93036B 07270.31673174 .00000030 00000-0 20391-4 0 2997
2 22676 074.0421 190.2875 0018551 003.8531 356.2753 14.33042505746561

Test Case 4 HEO: 66 degrees inclination LEO
JASON
1 26997U 01055A 07269.49083870 -.00000038 +00000-0 +10000-3 0 08486
2 26997 066.0446 298.5464 0008096 264.5280 095.4809 12.80929963271384

Test Case 5 HEO: 51 degrees inclination LEO
ISS (ZARYA)
1 25544U 98067A 07270.64846184 .00007688 00000-0 54207-4 0 607
2 25544 051.6348 295.2535 0002708 045.3662 054.4413 15.75238918506947

When selecting test cases for use with a sensor in GEO orbit there are several different factors that need to be investigated. Unlike a HEO orbit, a GEO sensor is much more likely to observe a target that is also in GEO orbit and very close to the sensor, so this case must be investigated. Additionally, a GEO sensor may see targets crossing the equatorial plane that are in HEO or GEO and so they will not be near the surface of the Earth when they are observed. A GEO sensor may also see targets

in highly eccentric orbits and the ability of the model to determine their orbits must be tested. Inclined LEO orbits and polar orbits must also be tested to show that the method continues to work with these orbits as it does when the sensor is in a HEO orbit. Finally, one unique orbit is tested, Chinasat 7 is a satellite with a 28 degree inclination that has a near GEO altitude when it crosses the equatorial plane and its solution is investigated.

Test Case 1 GEO: GEO, 0 degree inclination GEO
SPACEWAY 3
1 32018U 07036A 07268.23359173 -.00000182 +00000-0 +00000-0 0 00393
2 32018 000.1973 288.5111 1772164 041.4316 020.7444 01.00364884000543

Test Case 2 GEO: near GEO, 28 degrees inclination, eccentric orbit
CHINASAT 7
1 24282U 96048A 07264.78888470 -.00000229 00000-0 10000-3 0 1129
2 24282 028.1690 345.6951 3099610 354.9022 002.6532 01.06657574 45729

Test Case 3 GEO: high altitude apogee, 28 degrees inclination, eccentric orbit
N-2 R/B(2)
1 12787U 81012C 07270.74053674 .00006018 00000-0 64665-3 0 1367
2 12787 028.2407 310.6441 6437703 242.5277 039.3189 03.43267090274783

Test Case 4 GEO: 51 degrees inclination LEO
ISS (ZARYA)
1 25544U 98067A 07270.64846184 .00007688 00000-0 54207-4 0 607
2 25544 051.6348 295.2535 0002708 045.3662 054.4413 15.75238918506947

Test Case 5 GEO: 99 degrees inclination LEO
OPS 4467 A
1 00812U 64031A 07270.22049292 -.00000005 00000-0 21485-4 0 510
2 00812 099.8399 266.8336 0005731 143.8671 216.2893 14.23220549243754

Test Case 6 GEO: 20 degrees inclination LEO
DELTA 2 R/B
1 22015U 92039B 07270.49272632 .00000760 00000-0 22968-4 0 3947
2 22015 020.7183 320.6482 0076224 136.6272 224.0191 14.98105611855144

4.3 Data Generation

The data for the analysis of each of these test cases was generated using version 7 of the Satellite Tool Kit (STK). Following the selection of a suitable test case, as described in the previous section, a generic sensor platform was generated either in a GEO or HEO orbit depending on the test case being run. The GEO sensor orbit had an altitude of 36,000 km and exactly 0 degrees of inclination, while the HEO orbit had an apogee of 36,000 km and an inclination of 63.4 degrees. The test case was then propagated forward from the data listed in the AFSPC catalog entry until both the sensor and the target were within a reasonable line of sight of each other with the Earth as a background. The data then generated was the latitude and longitude of the suborbital path of the target, the position of the target in the ECI frame, and the position of the sensor in the ECI frame. The suborbital path data generated by STK was perfectly nadir so care has been taken to explain the weakness of the spherical trigonometry initial orbit determination technique which relies on this perfect data. The positions of the sensor and the target were used to generate the Azimuth and Elevation measurements from the sensor to the target as described in Chapter 3. With this data, as well as the time at which each measurement was made, the analysis was conducted.

4.4 Debris

Debris in orbit is a well known fact. Tracking and accounting for debris has become a significant field of research. The obvious danger that these objects pose to functional satellites in orbit is of great concern. A fundamental assumption of this thesis is that observed fastwalkers are in low Earth orbit. The possibility that debris could represent a fastwalker must be explored.

Only one version of the Space Command catalog was used for this thesis. This issue of the catalog was for the period between 26 and 28 September, 2007 and contained 11,231 total entries. Of these entries 6,294 were classified explicitly as debris, or about 56 percent. Defining what constitutes debris is difficult. Space

Command currently tracks all debris larger than 10 cm in diameter which provides a bottom limit on the size of debris. However, the upper limit remains undefined. Rocket bodies and other large objects are not identified as debris so we can assume that those items classified as debris are items that represent a piece of what was once a single tracked object. One clear and obvious example is the Fengyun 1C satellite. On January 17, 2007 China tested an anti-satellite missile on this satellite. The test was successful and the satellite broke into many smaller pieces. Over 2,000 pieces of debris from this single satellite are being tracked in orbit. If none of these pieces are smaller than 10 cm in diameter, then we can safely assume that not very many of the pieces of this one satellite remaining in orbit are too much larger than that. The possibility that one of these many small pieces of debris could represent a fastwalker observed by our sensor must be investigated.

A fastwalker is defined as a bright object moving across the focal plane array of an Earth observing sensor at a velocity too great to represent an object within the atmosphere. The key word in this definition is the term bright. It means an object that is reflecting enough sunlight in the direction of the sensor to be visible against the background of the Earth. If we assume that the objects remaining from the Fengyun 1C satellite are larger than 10cm in diameter but are irregularly shaped then we can begin to build a picture of their behavior in orbit. Since these objects are no longer stabilized in any way they will rotate about their major axis or align in the orientation which produces the least drag from the atmosphere. Outside the atmoshpere, if a portion of these objects have a reflective surface they will in fact reflect sunlight when exposed to the sun for an amount of time dependent on the period of the object's rotation about it's major axis. This period of time will be very brief and the reflection will be in an irregular direction as the orbit of the debris changes so does it's position in relation to the sun. It is therefore possible for these objects to reflect visible light in the direction a sensor, but it is unlikely that they will do so for a length of time necessary to develop a data track in the absence of drag. If the orbit is not at a high altitude significant drag is present and the object

will align itself in the orientation which produces the least drag. This will create a situation where observation is possible but the small size and irregular reflecting surface of debris will make this unlikely.

As mentioned previously, the catalog has a large number of items classified as debris listed. The surprising fact is that the majority of these listings can be accounted for as pieces of debris from just a handful of satellites like the Fengyun 1C. Many of the items listed as debris must also be the result of debris created during the staging and boosting phases of certain classes of rockets. However, the rocket bodies themselves are not classified as debris and are not eliminated as possible fastwalkers. The reason for this rests in the size and regular shape of a typical rocket body. Since most rocket bodies will be cylindrical and unstabilized they will rotate about their major axis. During a single rotation the cylindrical shape will reflect light once towards any arbitrary point. For this reason, observers will see them not as a constant reflection but they will appear to wink. Due to this ability, as well as the surface area available to reflect sunlight, rocket bodies cannot be discounted as possible targets as we can debris fields like that of the Fengyun 1C satellite.

4.5 Test Case Results

This section will discuss the results using the test cases previously mentioned and the models previously developed. In developing data, several assumptions about the length of time in which the sensor would be able to view the target had to be made. From apogee, a one degree field of view for the sensor allowed it to see a circular area with a diameter of over 1350 kilometers on the Earth's surface. This was assumed to be a reasonable field of view. Hence, the maximum amount of time a target would be visible was for about four minutes assuming the target was detected and tracked early in the field of view. The time between observations was set at one second, although any time period between observations is acceptable as long as they lie within the field of view. A minimum of two observations was required for the spherical trigonometry initial orbit determination technique and three observations

for the Gauss technique. In order for the least squares process to have any significant value we must have more observations than are used in the initial orbit determination technique. Additionally, the Air Force Space Command catalog used was assumed to be the one published nearest the time of the observed fastwalker. Since inclination is essentially constant, but the node changes slowly due to the oblateness of the Earth, they can be considered constant for the duration of the data track. These values are also treated as constants when searching the AFSPC catalog so it is necessary to use the catalog published closest to the epoch time of the observed fastwalker so that the error in node made by this assumption is small. It is not necessary to propagate each TLE in the catalog to the epoch time being considered to successfully identify the target.

4.5.1 Test Case 1 HEO - Thor Rocket Body. The method of using the slope between two points and the geometry of the Earth's surface resulted in an estimate of the state vector which was very good at nadir, an elevation of -90 degrees, for three of the four values in the state vector with two data points one second apart. The state vector consisted of four elements: right ascension of the ascending node, inclination, argument of latitude, and altitude. This method resulted in an estimate of the node as 68.12 degrees and inclination of 100.03. From the two line element set in the previous section, we see that these values are within two degrees of the values needed. The value of the argument of latitude, as discussed previously, is not a constant and need only be used to check that the satellite selected as the target from the catalog is in fact the correct satellite by propagating the element set to the epoch time of the solution. Altitude is necessary for the least squares process, but the accuracy of its value is not critical. However, the trigonometry technique produces extremely poor estimates of the altitude which are too poor to allow the convergence of least squares. While increasing the time between observations improves the estimate of this value, it greatly decreases the accuracy of the other three values. Additionally, the improvement in the estimate of the altitude still remains relatively poor in comparison

to the accuracy of the other three values. The estimate is usually more than several hundred kilometers off of the actual altitude. Relying on an earlier assumption, that the viewed satellite is in a low earth orbit, an average value for these satellites can be selected which is adequate for least squares to function. A value of 500 km was used which is a $\dot{\theta}$ of about 0.0011 radians per second. The values for the performance of the spherical trigonometry technique being discussed here are from nadir observations. Nadir observations are ideal data for this method, but are unlikely to ever be obtained by a real platform. This method suffers greatly from off nadir angles and its usefulness will be discussed along with the least squares process and its ability to compensate for poor observations.

$$X_{Trig} = \left[\begin{array}{cccc} RAAN = 68.1209^\circ & I = 100.0316^\circ & \theta = 57.6837^\circ & \dot{\theta} = 0.00001\, rad/sec \end{array} \right]$$

$$X_{Trig} - X_{Truth} = \left[\begin{array}{ccc} RAAN = 2.2138^\circ & I = 1.1746^\circ & \theta = -.6857^\circ \end{array} \right]$$

The Gauss initial orbit determination technique uses the same angular data necessary for the least squares process. It can be adapted to use the same latitude and longitude of the background data as the previous method, but is no more accurate with that data than using simple trigonometry. When adapted to using azimuth and elevation data, the Gauss technique eliminates the inability to directly observe the altitude of the target as a problem. This method provided an excellent estimate of all four states. The estimated values of the state vector were:

$$X_{Gauss} = \left[\begin{array}{cccc} RAAN = 67.6875^\circ & I = 99.0811^\circ & \theta = 50.8519^\circ & \dot{\theta} = 0.00108\, rad/sec \end{array} \right]$$

$$X_{Gauss} - X_{Truth} = \left[\begin{array}{ccc} RAAN = 1.7804^\circ & I = .2241^\circ & \theta = -6.1461^\circ \end{array} \right]$$

It can be quickly seen from the correct values that Gauss has given us an estimate of inclination accurate to within one degree. It is not as obvious that the value of altitude is also extremely accurate in comparison to the previous technique. The difference between the true altitude of the target and the calculated value was 17.9 km.

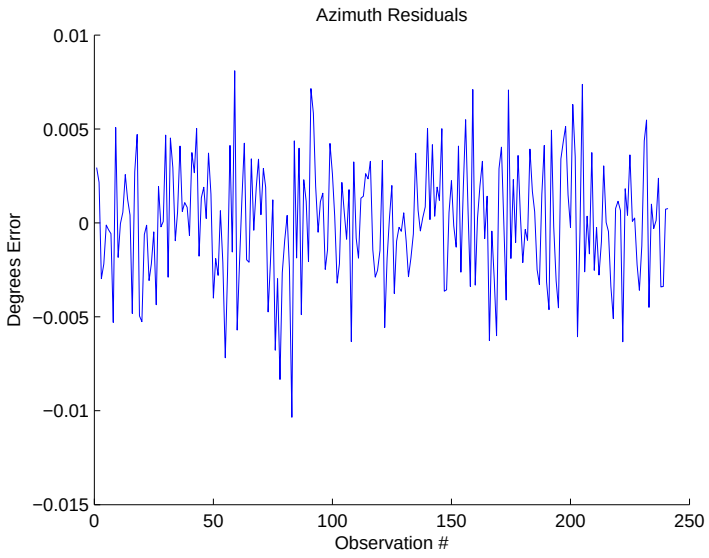

Figure 4.2: Azimuth Residuals With 3.18e-3 Noise for Thor Rocket Body

In comparison to traditional techniques of satellite tracking using radar to calculate range, this is very poor but for our purposes it is more than adequate.

The purpose of the non-linear least squares process is to account for additional observations given an initial estimate of the state in order to reduce the effect of sensor errors. The accuracy of that state estimate is important in that a poor estimate of the initial state vector cannot always be corrected for by the least squares process. On the other extreme, with a very accurate estimate, it does not always improve on the initial guess. This is because the developed non-linear least squares relies on a two-body formulation of the system dynamics, which was judged to be more than adequate for the short time period over which observations occur. Using the results from Gauss as our initial guess the least squares process returns:

$$X_{NLLS} = \left[\begin{array}{cccc} RAAN = 65.3317^\circ & I = 99.1120^\circ & \theta = 52.0130^\circ & \dot{\theta} = 0.000961\, rad/sec \end{array} \right]$$

$$X_{NLLS} - X_{Truth} = \left[\begin{array}{ccc} RAAN = -.7830^\circ & I = .2130^\circ & \theta = -4.985^\circ \end{array} \right]$$

These values were obtained with random noise in the data with a standard deviation of 3.18e-3 degrees which was shown earlier to represent a one kilometer ground sample

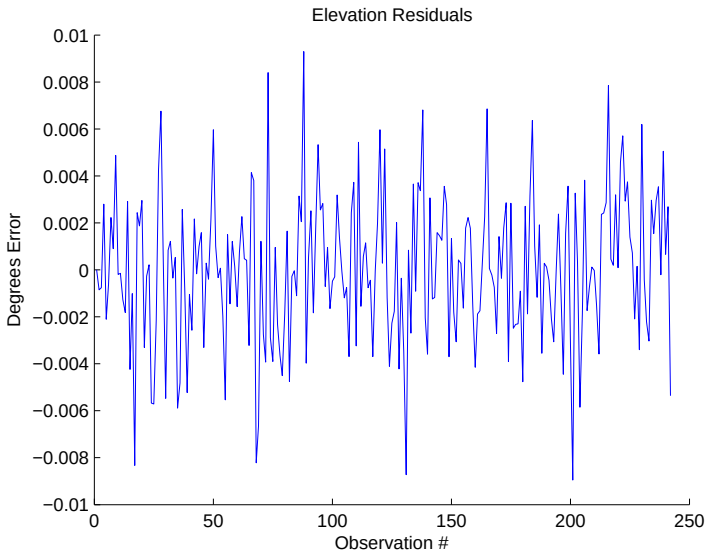

Figure 4.3: Elevation Residuals With 3.18e-3 Noise for Thor Rocket Body

distance. The difference from truth represents the values that would be searched for in the catalog, meaning that the -.7830 degree difference in RAAN represents the difference between the listed value and that in the catalog, not the difference between the RAAN at epoch and this value which is smaller. However, the difference value for θ does represent the error from the θ at epoch since this value is otherwise meaningless. The residuals can be seen to be very small in Figure 4.2 and Figure 4.3. The regular shape of the plot shows that model error is much more significant than sensor data error. Also, using the root mean square as the convergence criteria, we can see the rate of convergence in Figure 4.4 as well as the magnitude of the corrections being made to the initial state. The least squares process converged in 33 iterations in this scenario and as noise increased this number went up slowly, staying below 100 for a successful convergence. One factor to note is that the good initial guess resulted in a least squares data run which was only slightly more accurate than the initial guess.

The covariance matrix is very small, due to the small errors associated with the observations. It was found that the covariance was not a good predictor of the actual error. This is consistent with the fact that the least squares dynamics model used was not as accurate as the model used to generate data. As a result, the search

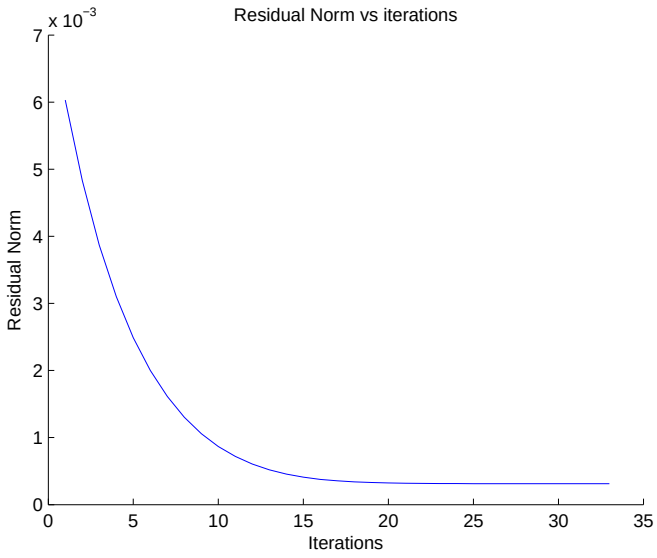

Figure 4.4: Convergence

of the catalog for objects matching the estimate had to look well beyond the range suggested by the covariance values.

$$P = \begin{bmatrix} 0.19080 & -0.06540 & -0.14544 & 0.00096 \\ -0.06540 & 0.03046 & 0.05619 & -0.00032 \\ -0.14544 & 0.05612 & 0.11589 & -0.00077 \\ 0.00096 & -0.00032 & -0.00077 & 0.00011 \end{bmatrix} \times 1e-3 \qquad (4.2)$$

Taking the values from the spherical trigonometry initial orbit determination technique that were more than two degrees off with perfect nadir data, we find that the least squares process with same amount of noise in the data returns the same results for the value of the state as seen above with the Gauss inputs. Investigating the data sets generated for this thesis for a sensor in HEO, the largest off nadir angle of elevation was found to be 5.2 degrees. Since the sensor cannot look at an angle too far off nadir before creating significant orthorectification effects on the imagery from the curvature of the Earth, we can assume a 10 degree off nadir look angle as being the greatest allowable. Using this angle the largest amount of error possible in both latitude and longitude is less than one degree, which will translate into nearly

the same amount of error in the values of node and inclination for the target satellite. The least squares process was found to easily compensate for as much as 5 degrees of random error in the first three states, when the estimate of altitude remained at a fixed 500 km and the level of noise in the data at 3.18e-3 degrees.

The results when we lower the GSD to one meter are worth noting. Both Gauss and the least squares process return answers that are more accurate than with a one kilometer GSD but these improvements are small for the least squares process. Gauss improves significnatly when used with more accurate data. We can see below that the results are nearly similar for Gauss and least squares. What becomes more apparent when the GSD is lowered to one meter is shown by the plots of the residuals. The regular shape of the residuals shown in Figure 4.5 and Figure 4.6 is the result of model error being greater than the error due to noise in the data. This helps explain the poor results from the covariance matrix which fails to account for the model error, a significant factor that is not readily apparent when the noise in the data is 3.18e-3 degrees.

$$X_{Gauss} = \left[\begin{array}{cccc} RAAN = 66.1549^\circ & I = 98.8411^\circ & \theta = 51.4038^\circ & \dot{\theta} = 0.00103\, rad/sec \end{array} \right]$$

$$X_{Gauss} - X_{Truth} = \left[\begin{array}{ccc} RAAN = .2478^\circ & I = -.0159^\circ & \theta = -5.5942^\circ \end{array} \right]$$

$$X_{NLLS} = \left[\begin{array}{cccc} RAAN = 65.8516^\circ & I = 99.0382^\circ & \theta = 51.7056^\circ & \dot{\theta} = 0.000999\, rad/sec \end{array} \right]$$

$$X_{NLLS} - X_{Truth} = \left[\begin{array}{ccc} RAAN = -.0555^\circ & I = .1808^\circ & \theta = -5.2924^\circ \end{array} \right]$$

Concerning noise, one final look at the robustness of the non-linear least squares process is warranted. Due to requests from the sponsor, the effect of ground sample distance was investigated with all other sources of error assumed to be small in comparison. If these assumptions are relaxed and the level of random noise is raised to 10e-3 degrees, representing a GSD of 12.65 km, the least squares process was found to still converge to a reasonable guess at the state, allowing for a match to be found in

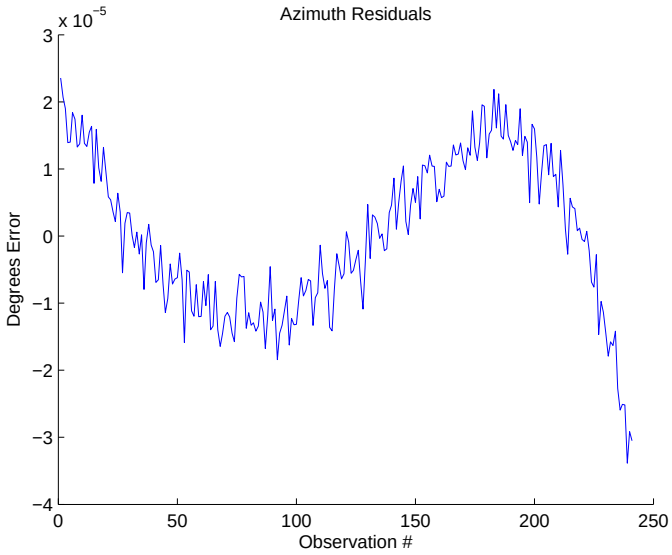

Figure 4.5: Azimuth Residuals With 3e-6 Noise for Thor Rocket Body

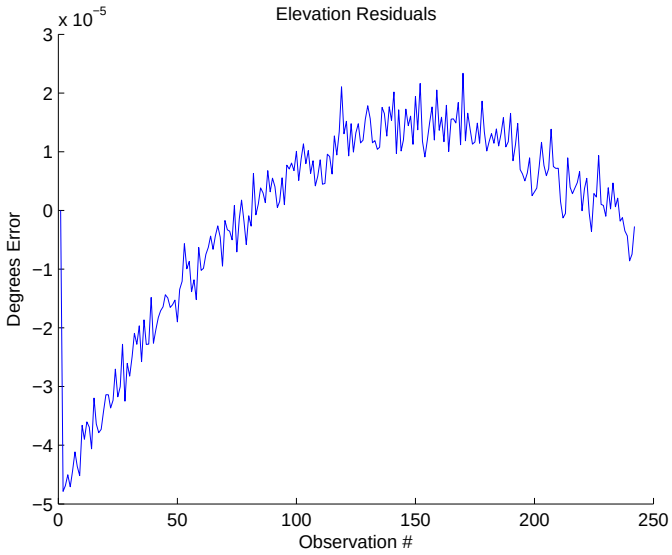

Figure 4.6: Elevation Residuals With 3e-6 Noise for Thor Rocket Body

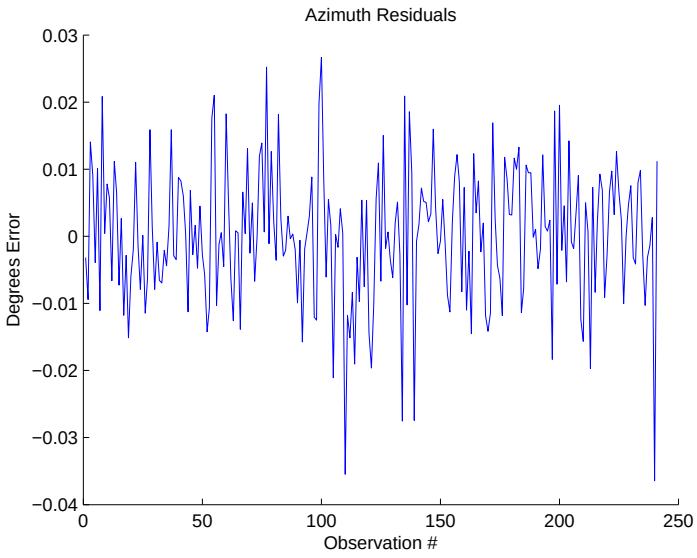

Figure 4.7: Azimuth Residuals With 10e-3 Noise for Thor Rocket Body

the catalog. The limiting factor at this level of noise is the use of Guass as an initial orbit determination technique, 10e-3 represents the error point at which Gauss would begin returning unreasonable estimates of the slant range.

$$X_{NLLS} = \left[\begin{array}{cccc} RAAN = 62.3955^\circ & I = 100.0189^\circ & \theta = 53.9412^\circ & \dot{\theta} = 0.000749\, rad/sec \end{array} \right]$$

$$X_{NLLS-Truth} = \left[\begin{array}{ccc} RAAN = 3.5116^\circ & I = 1.1615^\circ & \theta = -3.0568^\circ \end{array} \right]$$

While still small, the residuals show again that the level of noise in the data has overcome the model error, see Figure 4.7 and Figure 4.8.

The final step in validating the search for the target is to actually determine if the data developed by the methods discussed above is accurate enough to identify the target satellite. As mentioned above, the covariance matrix was extremely small so we need to begin looking in a much larger range. If we look simply at inclinations within one degree of our estimated value from our results with 3.18e-3 degrees of noise a search of the catalog returns 2,276 matches. If we include the right ascension of the ascending node in that search we find only one match, which is our target satellite.

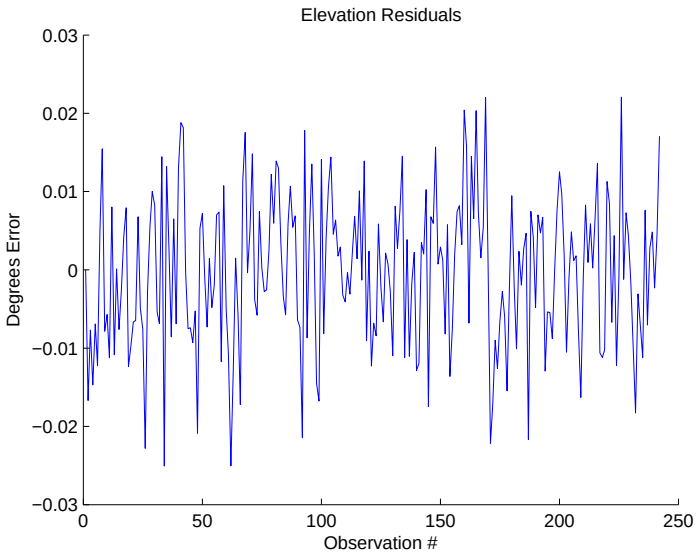

Figure 4.8: Elevation Residuals With 10e-3 Noise for Thor Rocket Body

In this case it proved unnecessary to propagate the orbit to find that this is indeed the correct target, but doing so confirms that it is.

4.5.2 Test Case 2 HEO - Cosmos. This test case dealt with the 82 degree inclination orbit. Both initial orbit determination techniques worked fairly well, with the Gauss technique once again proving to be far superior than the spherical trigonometry technique even when provided with nadir data. The spherical trigonometry technique continued to have extremely poor estimates of the target's altitude. When a search of the catalog was conducted three objects were found to be close matches in both inclination and RAAN. After propagating, the Cosmos satellite was easily identified by the argument of latitude at epoch.

$$X_{Trig} = \left[\begin{array}{cccc} RAAN = 361.839^\circ & I = 83.999^\circ & \theta = 125.469^\circ & \dot{\theta} = 0.00000895\, rad/sec \end{array} \right]$$

$$X_{Gauss} = \left[\begin{array}{cccc} RAAN = 358.2412^\circ & I = 81.6152^\circ & \theta = 122.0505^\circ & \dot{\theta} = .000828\, rad/sec \end{array} \right]$$

$$X_{NLLS} = \left[\begin{array}{cccc} RAAN = 359.1751^\circ & I = 82.1518^\circ & \theta = 121.9621^\circ & \dot{\theta} = .000901\, rad/sec \end{array} \right]$$

$$X_{NLLS} - X_{Truth} = \left[\begin{array}{ccc} RAAN = .2471^\circ & I = -.3162^\circ & \theta = .0111^\circ \end{array} \right]$$

4.5.3 Test Case 3 HEO - SL-8 Rocket Body. This test case was in the 74 degree inclination band. Here least squares improved on Gauss significantly and the result was very close to the actual values. The search of the catalog resulted in three matches within one degree of these values. Only one of which matched after including argument of latitude following propagation to epoch.

$$X_{Trig} = \left[\begin{array}{cccc} RAAN = 191.3505^\circ & I = 74.6582^\circ & \theta = 65.5391^\circ & \dot{\theta} = 0.00000897\, rad/sec \end{array} \right]$$

$$X_{Gauss} = \left[\begin{array}{cccc} RAAN = 189.3430^\circ & I = 74.3784^\circ & \theta = 58.9821^\circ & \dot{\theta} = 0.00099\, rad/sec \end{array} \right]$$

$$X_{NLLS} = \left[\begin{array}{cccc} RAAN = 189.4768^\circ & I = 74.4999^\circ & \theta = 58.9529^\circ & \dot{\theta} = 0.001005\, rad/sec \end{array} \right]$$

$$X_{NLLS} - X_{Truth} = \left[\begin{array}{ccc} RAAN = -.3012^\circ & I = -.3311^\circ & \theta = -6.932^\circ \end{array} \right]$$

4.5.4 Test Case 4 HEO - Jason. The inclination tested here was 66 degrees. In this test case the Gauss technique provided a poor answer in both RAAN and inclination but the least squares process was easily able to converge on the correct solution. As the catalog was searched outward from the least squares results, the first match proved to be the correct target after propagating to confirm the argument of latitude at epoch.

$$X_{Trig} = \left[\begin{array}{cccc} RAAN = 295.9209^\circ & I = 67.1962^\circ & \theta = 122.2495^\circ & \dot{\theta} = 0.00000790\, rad/sec \end{array} \right]$$

$$X_{Gauss} = \left[\begin{array}{cccc} RAAN = 301.4431^\circ & I = 62.7412^\circ & \theta = 113.8477^\circ & \dot{\theta} = 0.001165\, rad/sec \end{array} \right]$$

$$X_{NLLS} = \left[\begin{array}{cccc} RAAN = 297.6072^\circ & I = 65.9347^\circ & \theta = 115.1594^\circ & \dot{\theta} = .0009512\, rad/sec \end{array} \right]$$

$$X_{NLLS} - X_{Truth} = \left[\begin{array}{ccc} RAAN = -.2959^\circ & I = -.0643^\circ & \theta = -6.193^\circ \end{array} \right]$$

4.5.5 Test Case 5 HEO - International Space Station. This test case represented the lowest inclination tested at 51 degrees. It begins to show a slight failing of the use of azimuth and elevation data as the portion of the observed orbit is nearly flat with respect to the Equator and low on the sensor horizon. This resulted in only small

changes in the recorded values for elevation, which resulted in the poor performance of least squares. In this case the spherical trigonometry technique proved to be superior in RAAN and inclination though the least squares and Gauss techniques both still worked well enough to identify the target. A search of the catalog around these values revealed five objects. Two of these objects were debris from the Space Station itself and could be ignored. The other two did not fit when propagated forward. The final object was the target.

$$X_{Trig} = \left[\begin{array}{cccc} RAAN = 294.1198^\circ & I = 51.6595^\circ & \theta = 81.5910^\circ & \dot{\theta} = 0.00000713\, rad/sec \end{array} \right]$$

$$X_{Gauss} = \left[\begin{array}{cccc} RAAN = 293.5153^\circ & I = 51.7434^\circ & \theta = 73.9015^\circ & \dot{\theta} = 0.001132\, rad/sec \end{array} \right]$$

$$X_{NLLS} = \left[\begin{array}{cccc} RAAN = 293.7612^\circ & I = 51.9402^\circ & \theta = 73.7151^\circ & \dot{\theta} = .001125\, rad/sec \end{array} \right]$$

$$X_{NLLS} - X_{Truth} = \left[\begin{array}{ccc} RAAN = .2042^\circ & I = .3662^\circ & \theta = -14.115^\circ \end{array} \right]$$

4.5.6 Test Case 1 GEO - Spaceway 3. When using a sensor in GEO the problem of identifying a target becomes more difficult. A key assumption that was made when working with a sensor in HEO no longer applies. No longer can we assume that the object being viewed is in a LEO orbit. We must also consider whether the object is also in GEO or in a Medium Earth Orbit (MEO). Spaceway 3 is a GEO satellite that allowed us to explore the problem of another satellite passing directly in front of a GEO sensor and here we see the method fail. The spherical trigonometry technique results appear to be close in inclination, but it must be remembered that those results were generated using perfectly nadir data. Nadir data is extremely unlikely in any scenario and becomes even more so as the distance between the observer and the target decreases. The spherical trigonometry method remained valid only when the target was near the Earth's surface and the sensor was in a high enough altitude orbit so as to make errors from viewing at non-nadir angles small. When viewing an object very near the sensor, these non-nadir angles can span the entire

region of space visible to the sensor.

$$X_{Trig} = \left[\begin{array}{cccc} RAAN = -293.188^\circ & I = 0.1923^\circ & \theta = 70.8042^\circ & \dot{\theta} = 0.00002796\, rad/sec \end{array} \right]$$

$$X_{Trig} - X_{Truth} = \left[\begin{array}{ccc} RAAN = -221.699^\circ & I = 0.0005^\circ & \theta = 1.541^\circ \end{array} \right]$$

Identifying the altitude of the target is impossible with the spherical trigonometry technique but Gauss gives us an estimate of the target satellite's altitude that is at least accurate enough to determine that it is also in a GEO or near GEO orbit. In this case, using Gauss, we can determine that the altitude of the target was greater than 30,000 kilometers, making it a target which we will be unable to identify.

$$X_{Gauss} = \left[\begin{array}{cccc} RAAN = 182.2929^\circ & I = 171.2489^\circ & \theta = 171.5152^\circ & \dot{\theta} = 0.00009442\, rad/sec \end{array} \right]$$

$$X_{Gauss-Truth} = \left[\begin{array}{ccc} RAAN = -106.218^\circ & I = 8.5538^\circ & \theta = 99.1702^\circ \end{array} \right]$$

The problem of attempting to identify satellites in GEO from GEO was attempted in much greater detail in Osedacz [9]. One difference must be noted. Osedacz declared that it was impossible to determine if the object being viewed was in GEO or LEO. The Gauss method used here was able to provide an estimate of the position vector of the target that was more than adequate to determine the difference between a GEO and a LEO satellite. Since we can differentiate the two using Gauss, either of the initial orbit determination techniques provides an adequate enough estimate of the state vector for a LEO target to apply the least squares process. In fact, while the method performs poorly with all satellites in GEO orbit, it performs very well with low inclination circular orbits near LEO. Finally, when attempting to use the initial estimates of the state calculated above for the target in GEO the least squares algorithm failed to converge. Based on this case, it is clear that the methods being developed in this thesis cannot be used to identify GEO or near GEO satellites.

4.5.7 Test Case 2 GEO - Chinasat 7. Chinasat 7 is a unique satellite in that it was the highest inclined orbit with a GEO altitude in the catalog that was used. It was selected to test the response of Gauss and the least squares process to an inclined GEO orbit. Once again the spherical trigonometry technique is unreliable for the reasons stated earlier but Gauss was somewhat surprising. The Gauss method failed to identify the satellite as being in an elliptical orbit and found the satellite to have a parabolic path. With the failure of both initial orbit determination methods the use of the least squares process is no longer possible. The Chinasat 7 test case shows that identifying any object too close to the sensor with angles-only data is extremely difficult no matter what inclination it is at.

4.5.8 Test Case 3 GEO - N-2 R/B(2). This test case represented a target satellite with a relatively low inclination, a high altitude apogee, and large eccentricity. At the point in which it crosses the equatorial plane, the satellite is close to LEO altitudes, but still several thousand kilometers from the surface of the Earth. In this test case, the method barely succeeds. As you can see below, the least squares process returned an answer that was actually farther from the truth than Gauss due to the failure of the circular orbit assumption upon which the least squares method relies. Using Gauss, we can quickly identify it but with the poor estimate of inclination provided by the least squares process 15 possible targets had to be propagated before the target satellite could be identified by the mean anomaly.

$$X_{Trig} = \left[\begin{array}{cccc} RAAN = 309.756^\circ & I = 30.6425^\circ & \theta = 175.8093^\circ & \dot{\theta} = 0.000797\, rad/sec \end{array} \right]$$

$$X_{Gauss} = \left[\begin{array}{cccc} RAAN = 313.8226^\circ & I = 27.7682^\circ & \theta = 169.1596^\circ & \dot{\theta} = 0.0009025\, rad/sec \end{array} \right]$$

$$X_{NLLS} = \left[\begin{array}{cccc} RAAN = 309.4777^\circ & I = 34.0038^\circ & \theta = 176.0905^\circ & \dot{\theta} = 0.0007839\, rad/sec \end{array} \right]$$

$$X_{NLLS} - X_{Truth} = \left[\begin{array}{ccc} RAAN = -.6063^\circ & I = 5.7978^\circ & \theta = 1.0965^\circ \end{array} \right]$$

4.5.9 Test Case 4 GEO - ISS(Zarya). In this test case Gauss provided us with a better estimate of the true value of inclination than the least squares process, but the target was easily identified using either method. For the HEO test cases, the ISS represented the lowest inclination test case and in that situation we saw Gauss and least squares perform equally well. With accurate data, Gauss has proved to be an excellent initial orbit determination technique, but it is deterministic, and a single piece of poor data would greatly affect its result.

$$X_{Trig} = \left[\begin{array}{cccc} RAAN = 294.224^\circ & I = 54.2655^\circ & \theta = 19.1739^\circ & \dot{\theta} = 0.001088\, rad/sec \end{array} \right]$$

$$X_{Gauss} = \left[\begin{array}{cccc} RAAN = 292.9583^\circ & I = 51.6694^\circ & \theta = 14.1904^\circ & \dot{\theta} = 0.001176\, rad/sec \end{array} \right]$$

$$X_{NLLS} = \left[\begin{array}{cccc} RAAN = 292.8512^\circ & I = 52.0571^\circ & \theta = 22.1802^\circ & \dot{\theta} = 0.001314\, rad/sec \end{array} \right]$$

$$X_{NLLS} - X_{Truth} = \left[\begin{array}{ccc} RAAN = -.0161^\circ & I = .448^\circ & \theta = 2.188^\circ \end{array} \right]$$

4.5.10 Test Case 5 GEO - OPS 4467 A. In this test case we see significant improvement using the least squares process. This follows with what we saw in the HEO test cases. The target was easily identified with the given state vector. The ability of this method to identify the target has been aided by the fact that the method performs very well at higher inclinations where the orbits are much more crowded than at lower inclinations but tend to be more circular.

$$X_{Trig} = \left[\begin{array}{cccc} RAAN = 268.6538^\circ & I = 103.1968^\circ & \theta = 22.7718^\circ & \dot{\theta} = 0.001044\, rad/sec \end{array} \right]$$

$$X_{Gauss} = \left[\begin{array}{cccc} RAAN = 267.3318^\circ & I = 100.0285^\circ & \theta = 15.0649^\circ & \dot{\theta} = 0.001011\, rad/sec \end{array} \right]$$

$$X_{NLLS} = \left[\begin{array}{cccc} RAAN = 267.4795^\circ & I = 98.6737^\circ & \theta = 20.4633^\circ & \dot{\theta} = 0.000931\, rad/sec \end{array} \right]$$

$$X_{NLLS-Truth} = \left[\begin{array}{ccc} RAAN = .3515^\circ & I = -1.119^\circ & \theta = -1.6118^\circ \end{array} \right]$$

4.5.11 Test Case 6 GEO - DELTA 2 R/B. This test case shows the ability of both initial orbit determination techniques and the least squares process to work well with a LEO satellite at low inclination. Using Gauss, the approximate altitude of the orbit can be used to determine that the target is not at or near GEO. Once it has been determined to be in a LEO orbit, the target can be identified. There are only 49 satellites in LEO at inclinations below 28 degrees, the highest inclination at which we are likely to see an object near GEO. This is a small number of satellites in comparison with the total number of objects being tracked from 0 to 28 degrees of inclination. It is likely that because most objects in this inclination rage are not in LEO most observed fastwalkers in this inclination range will be unidentifiable.

$$X_{Trig} = \left[\begin{array}{cccc} RAAN = 317.111^\circ & I = 22.0569^\circ & \theta = 15.7482^\circ & \dot{\theta} = 0.00101\, rad/sec \end{array} \right]$$

$$X_{Gauss} = \left[\begin{array}{cccc} RAAN = 315.5738^\circ & I = 20.6494^\circ & \theta = 9.6695^\circ & \dot{\theta} = 0.00114\, rad/sec \end{array} \right]$$

$$X_{NLLS} = \left[\begin{array}{cccc} RAAN = 316.3434^\circ & I = 20.6253^\circ & \theta = 15.9865^\circ & \dot{\theta} = 0.0000103\, rad/sec \end{array} \right]$$

$$X_{NLLS-Truth} = \left[\begin{array}{ccc} RAAN = -4.3046^\circ & I = -.0927^\circ & \theta = -1.4085^\circ \end{array} \right]$$

4.5.12 Direct Orbit Prediction. It has been made clear several times that the general method being developed is dependent entirely on the assumption that the target object is located in the catalog. However, since mistakes and omissions can be made, what can be done in the case where the object is not located within the catalog to identify its orbit? We have developed Gauss to provide us with a specific state vector, but it can also provide us with an estimate of the orbital elements of the target object. As has been shown in the test cases previously mentioned, Gauss can provide us with results that are accurate enough to determine the identity of the target using the catalog and given a data track spread over more than 30 seconds for a LEO object, about .5 degrees of arc. Gauss however, provides us with a poor direct estimate of the orbital position of a satellite.

ISS Orbital Element Comparison			
Element	ISS Actual	ISS Gauss	Difference
Semi-major Axis (km)	6716.786	6760.078	-43.292
Eccentricity	0.000482	0.008314	-.007832
Inclination (deg)	51.574	51.545	.029
RAAN (deg)	293.557	293.530	.027
Arg of Perigee (deg)	87.830	90.634	2.804
Mean Anomaly (deg)	354.044	343.568	10.476

Table 4.1: ISS Orbital Element Comparison

Table 4.1 is an example of the quality of the orbital elements estimate provided by Gauss. These results show that the largest error lies in the argument of perigee and the mean anomaly. The net effect of these errors is that the predicted position of the target satellite will lag far behind that of the actual. Initially, there are 930 km separating the calculated versus actual position, though they are following very similar suborbital paths. This error grows with each orbit as the error in the periods between the predicted and actual satellite orbits increases. After one full day, the error between the predicted and actual position is nearly 1/4 of an orbit. While the direct calculation of the orbit does not provide us with an accurate position of the target, it does provide enough information to reacquire the target after only a few orbits. As the entire method relies on the fact that very few satellites share both an inclination and a RAAN, it would be possible to reacquire the target and develop additional tracking data to refine the orbit. While this would require the dedication of satellite resources to reacquiring the target, it would be possible with an additional data track to develop a far superior direct calculation of the orbital elements by using either the methods developed by Gooding [5] or Osedacz [9].

V. Conclusions

Fastwalker orbit determination can be performed with a single data set from a HEO orbit and, with some limitations, a GEO orbit. Contrary to the broad assertions in a previous thesis on the topic that orbit determination of fastwalkers is not possible with a single data set [9], it is possible to characterize LEO orbits from a sensor platform in a HEO and GEO orbit using orbit matching. Data collected from future occurrences can be used to determine the orbit of the target object.

Two different initial orbit determination techniques were discussed. Both were successful in determining an estimate of the state vector necessary for additional analysis. The Gauss method was far more accurate than the spherical trigonometry technique over data sets that spanned several minutes, but as the observation time decreased the accuracy of the Gauss method dropped off sharply. Gauss suffered from poor estimates of the slant range when the noise level in the data was as large as .01 degrees. Over data sets spanning less than .5 degrees of arc the Gauss technique began to fail entirely and it becomes necessary to use the spherical trigonometry technique. The spherical trigonometry technique did suffer from poor results when looking off nadir, but as long as measurements remained within 10 degrees of nadir the least squares process was easily able to compensate.

The effects of ground sample distance on the ability of these methods to effectively determine the orbit of a fastwalker was also analyzed. It was found that for any reasonable value the level of noise in the data that would be imparted by a large ground sample distance did not have a significant affect on the results. Since commercially available platforms operating in the near infrared spectrum are available with ground sample distances of 10 meters or less from LEO, it is reasonable to assume that future platforms placed in a HEO or GEO orbit will not have ground sample distances much larger than one kilometer. In fact, the level of noise imparted by large ground sample distance does not prevent identification of the target satellite until these values are greater than 12.65 kilometers.

Non-linear least squares proved to be an effective method of refining the estimate of the state obtained from the initial orbit determination process. It allowed for the use of additional data and proved robust enough, even with very short data sets, to correct for several degrees of error in the initial guess of the state. Although the Gauss initial orbit determination method sometimes provided an initial guess that was very accurate over data sets spanning several minutes, its accuracy could not be confirmed without taking into account the additional data through the least squares process. This made the non-linear least squares process necessary to confirm the accuracy of the state vector developed using Gauss. When the target was in a highly eccentric orbit, the method did not fail but performed poorly even when the target was relatively close to the surface of the Earth. The method was unsuccessful in identifying targets at or near GEO.

Searching for a fastwalker target based on its right ascension of the ascending node and its inclination proved to be an effective way of identifying the orbit. This method is heavily dependent on using an Air Force Space Command catalog published during the period that the fastwalker was observed. While inclination is nearly constant, the node is not, and using a catalog published even days apart could invalidate the method. However, these catalogs are published several times a week and older copies are readily available. The argument of latitude was effective as the final step in confirming the identity of the fastwalker and even at the most crowded inclinations there was no difficulty in identifying the target.

The method developed was not intended to determine the orbit of the target alone. It was effective in identifying the target fastwalker from the data in the Space Command catalog. In not attempting to fully determine the orbital elements of the target fastwalker directly, this method circumvented some of the problems which could have arisen. Over very short data tracks with only angular data, an accurate estimate of the orbital elements of the target would have been impossible. In the recent paper by the Chinese National Defense Institute [7] over 30 minutes of data was needed in order to obtain even a poor independent estimate of the target satellite from a

HEO or GEO orbit, excluding the fact that it is unlikely that a LEO satellite would be visible to a satellite in HEO orbit for that amount of time. Using much shorter data spans, it was possible to take advantage of the existence and accuracy of the Space Command catalog and using only a few orbital elements identify the correct satellite in the catalog when the orbit is determined to be near LEO. From the two line element set in the catalog the orbit is then known to an acceptable standard.

5.1 Recommendations

The topic of fastwalker orbit determination has not received much attention in recent years. It was determined to be impossible for single data tracks in a geosynchronous orbit and no other orbit types were investigated. The advantage in working in a HEO orbit was the fact that these orbits are not crowded by other platforms, so the key assumption that the object was located near the Earth's surface was made. In Osedacz's thesis [9], he determined that the majority of fastwalkers visible to the platform he was working with were in fact also in geosynchronous orbit. This led to his inability to determine the range between the sensor and the target.

The focus of this thesis was on determining the orbit of the target fastwalker. Very little attention was given to the actual nature of the fastwalker phenomenon. Fastwalkers represent objects large enough to reflect enough light in one direction to be clearly visible from a HEO or GEO orbit. When these events are likely to occur and under what circumstances needs further analysis. In order to do so, significant amounts of data on actual fastwalker observations would be necessary. The time of the observation and the location of the observer would be the needed to even begin a thorough analysis. Since this data was not available for this thesis the topic was only briefly discussed. This is an area where a significant amount of future work could be conducted.

There are several areas where future work could be conducted to further the results obtained here. Obtaining actual fastwalker data and attempting to conduct a successful orbit matching using the techniques developed in this thesis would be a

useful follow on project. Additionally, future work could be conducted by attempting to use the same orbit matching idea and developing a matching technique which accounts for model error. This would improve the values of the covariance matrix. If the accuracy of these values is improved than the entire process from data entry into the system until target identification could be automated removing the need for a user search of the catalog.

During the development of this thesis, I was also impressed by the almost complete lack of modern research into angles-only initial orbit determination techniques. Gooding, [4, 5] is the only person currently active in researching new techniques, although Chris Sabol [11, 12] is working on applications in improving the accuracy of entries in the Air Force Space Command catalog with angular data. For my purposes, I found the Gauss method when used in conjunction with the Herrick-Gibbs method provided an estimate of the location of the orbit which was more than adequate. Additional experimentation on refining the least squares process is needed to attempt to account for model error in the covariance matrix. With model error accounted for, it would be possible to automate the entire process and remove the manual searching of the catalog currently necessary by the user to characterize the target. Despite the success of this method, determining the exact orbit of another satellite in GEO from a GEO sensor using angles-only data may still remain impossible. As it currently stands this method allows for the identification of over 82 percent of the objects listed in the AFSPC catalog.

Appendix A. Residual Plots

This Appendix includes all of the residual charts referenced in chapter 4. The residual plots shown are all with 10e-7 degrees of error. For examples of residual plots with larger errors please refer to chapter 4.

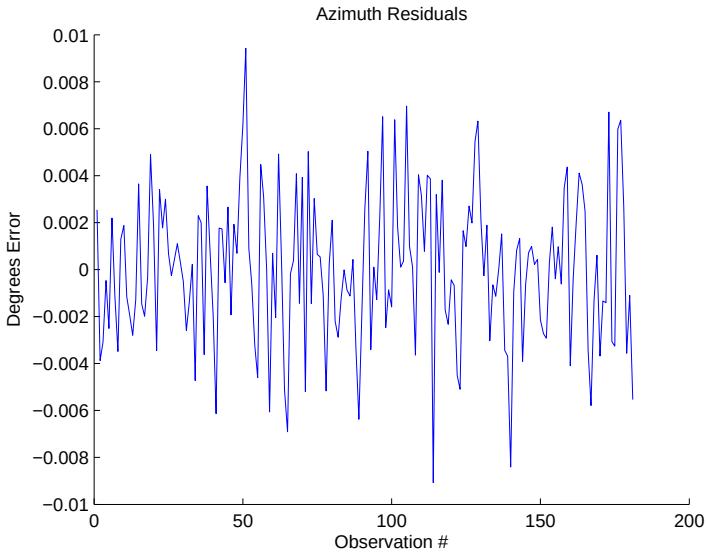

Figure A.1: Azimuth Residuals With 3.18e-3 Noise for Cosmos

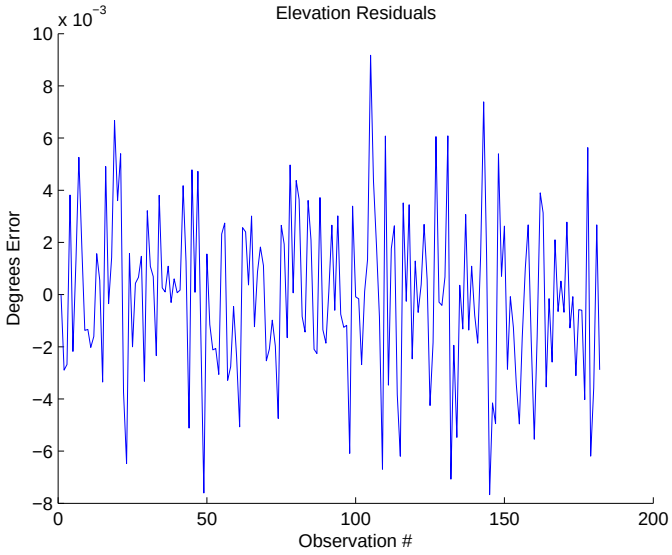

Figure A.2: Elevation Residuals With 3.18e-3 Noise for Cosmos

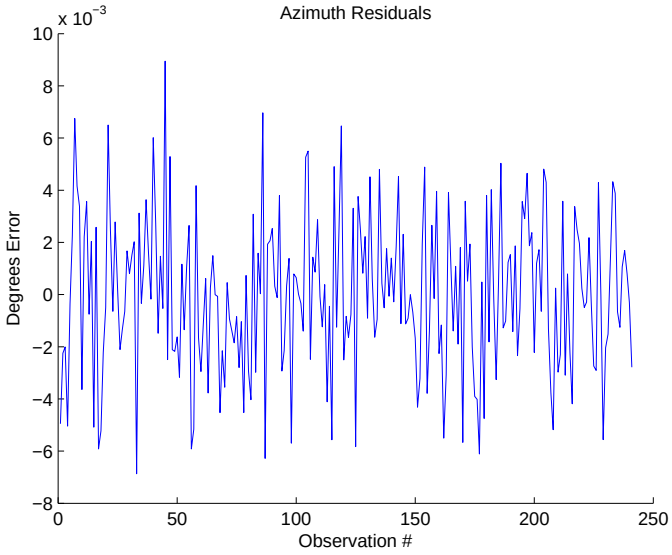

Figure A.3: Azimuth Residuals With 3.18e-3 Noise for SL-8 R/B

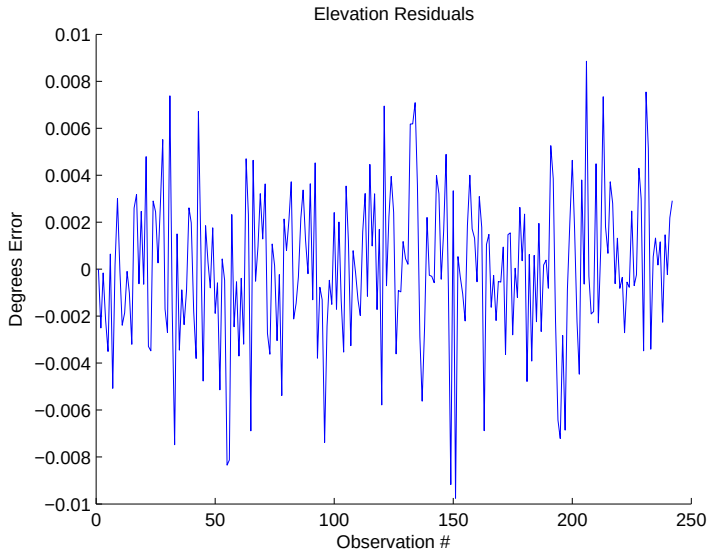

Figure A.4: Elevation Residuals With 3.18e-3 Noise for SL-8 R/B

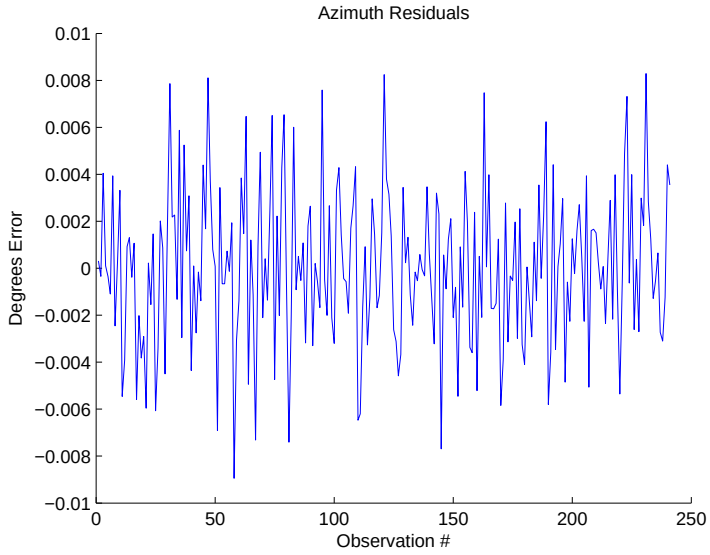

Figure A.5: Azimuth Residuals With 3.18e-3 Noise for Jason

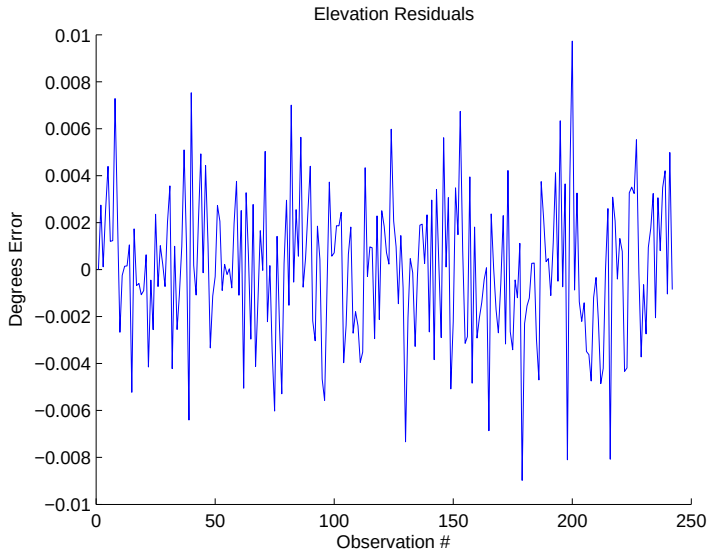

Figure A.6: Elevation Residuals With 3.18e-3 Noise for Jason

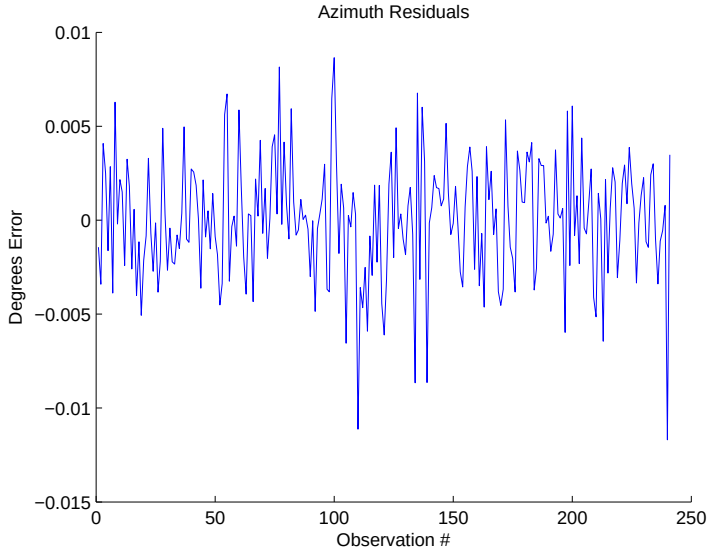

Figure A.7: Azimuth Residuals With 3.18e-3 Noise for ISS

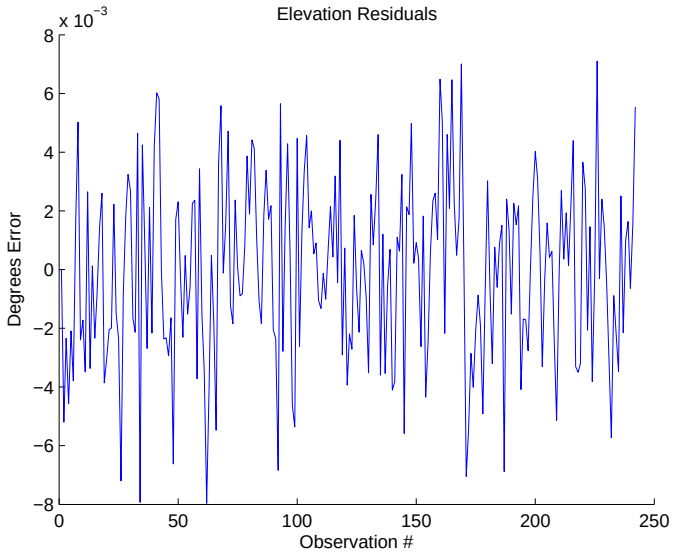

Figure A.8: Elevation Residuals With 3.18e-3 Noise for ISS

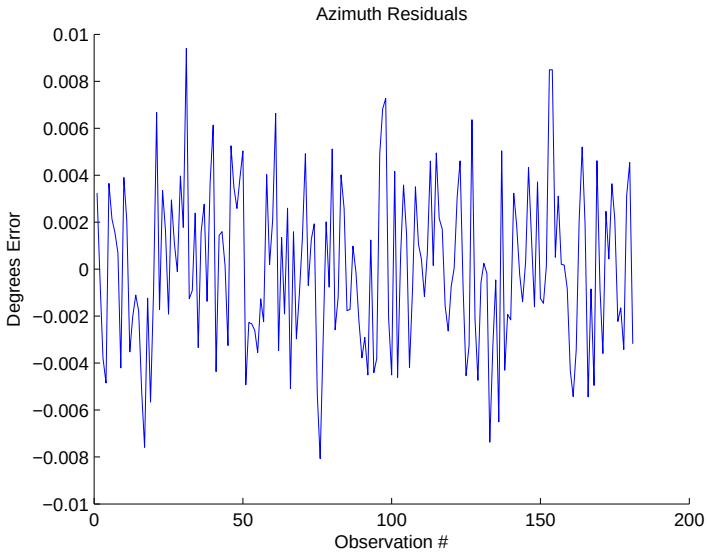

Figure A.9: Azimuth Residuals With 3.18e-3 Noise for N2 RB

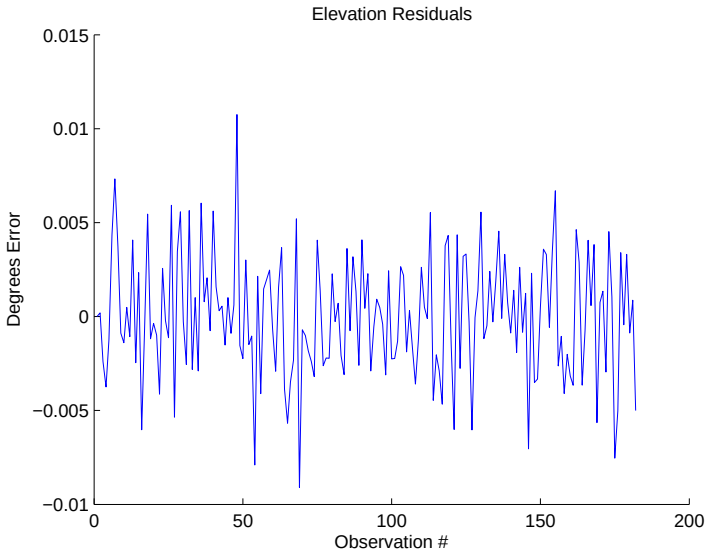

Figure A.10: Elevation Residuals With 3.18e-3 Noise for N2 RB

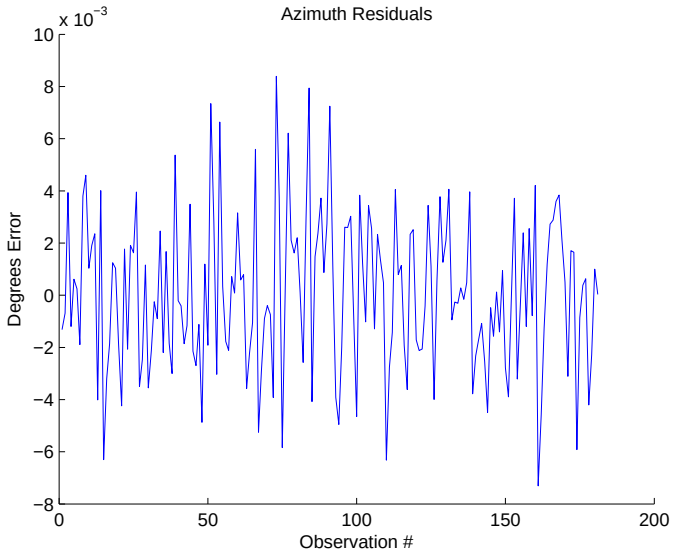

Figure A.11: Azimuth Residuals With 3.18e-3 Noise for ISS

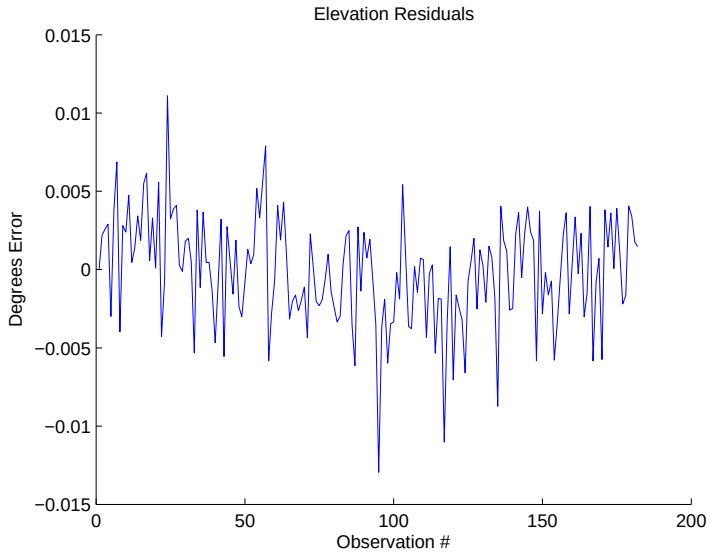

Figure A.12: Elevation Residuals With 3.18e-3 Noise for ISS

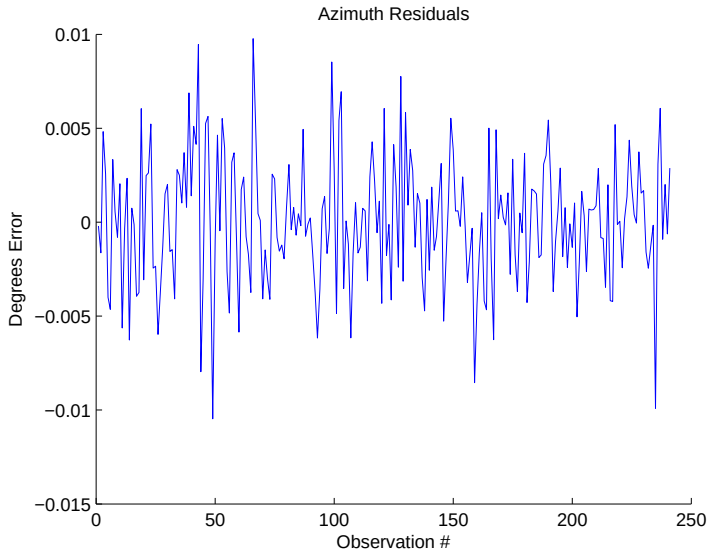

Figure A.13: Azimuth Residuals With 3.18e-3 Noise for OPS 4467 A

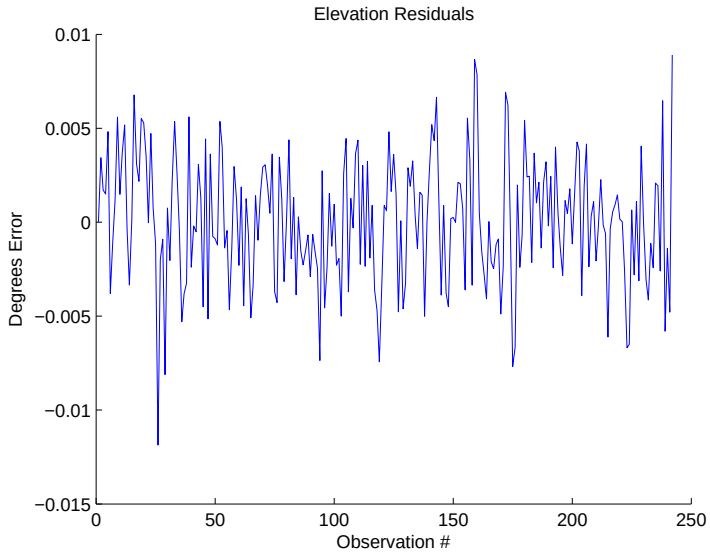

Figure A.14: Elevation Residuals With 3.18e-3 Noise for OPS 4467 A

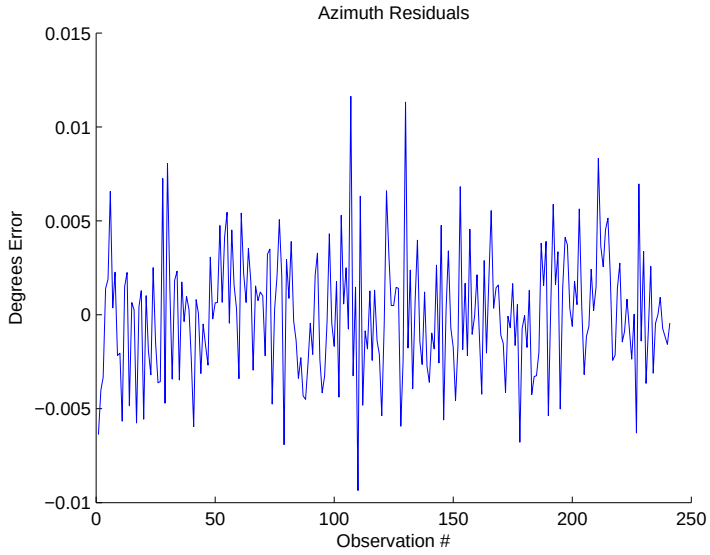

Figure A.15: Azimuth Residuals With 3.18e-3 Noise for Delta 2 R/B

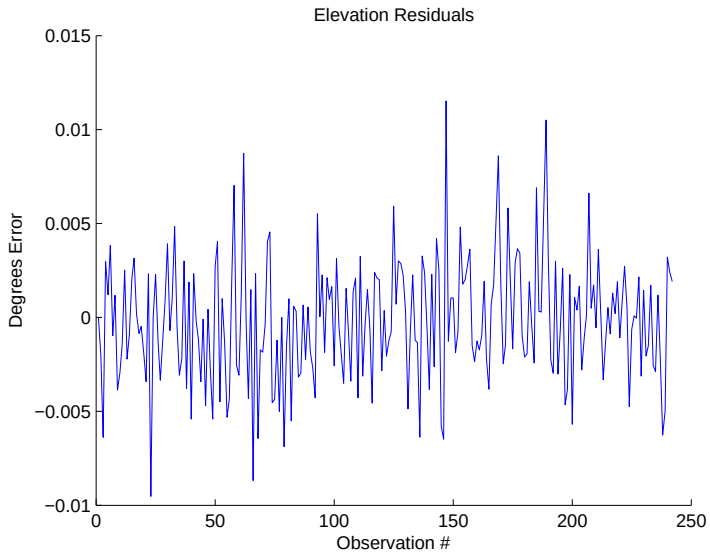

Figure A.16: Elevation Residuals With 3.18e-3 Noise for Delta 2 R/B

This appendix contains the core code elements. All code was written and functions in Matlab® .

B.1 Gauss Initial Orbit Determination

Listing B.1: The Gaussian initial orbit determination code. (appendix2/GaussIOD.m)

```
% ...
    --------------------------------------------------------------------

%
%                          Gauss_IOD.m
%
%  this code is the Gauss initial orbit determination
%
%  inputs          description                    range / units
%  year            Of observation                 Time
%  month            "     "
%  day
%  hour1-3         3 hours of observation
%  s1-3            3 seconds of observation
%  alpha           Azimuth                        radians
%  sigma           Elevation                      radians
%  r_site1-3       location of sensor ECI         kms
%
%  outputs         r2,v2,                         km/ km/s
 %                 argument of lat,omega (RAAN)   degrees
%                   i (inclination),
%                  Thetadot (alt)                 NLLS input term
% ...
    --------------------------------------------------------------------


clc
clear all
close all
%Constants
u_earthkm3persec2=398600.4415;
radius_earth=6378.1363

%for Cosmos
%Time of Observations
year=2007;month=9;day=27;
hour1=15;hour2=15;hour3=15;
min1=48;min2=49;min3=51;
s1=0.74;s2=59.74;s3=0.74;
%R_site
%data
%Azimuth
```

```
alpha=[0.601631211134485;1.057947399382605;1.306829387830625];
%Elevation
sigma=[-1.498307748963798;-1.506215139106105;-1.504976881780819];
r_site1=[-17470.7644479361      12124.4096188952        ...
    27495.1031194714]';
r_site2=[-17582.718937935       11991.5070697076        ...
    27779.4780612869]';
r_site3=[-17639.2080231445      11922.7654824776        ...
    27923.8294989853]';

%Line of Sight Unit Vector
L1=[-cos(sigma(1,1))*cos(alpha(1,1));cos(sigma(1,1))*sin(alpha...
    (1,1));sin(sigma(1,1))];
L2=[-cos(sigma(2,1))*cos(alpha(2,1));cos(sigma(2,1))*sin(alpha...
    (2,1));sin(sigma(2,1))];
L3=[-cos(sigma(3,1))*cos(alpha(3,1));cos(sigma(3,1))*sin(alpha...
    (3,1));sin(sigma(3,1))];

r_site=[r_site1,r_site2,r_site3];
%-------------------------------------------------------
%Quadrant check of sensor platform in ECI frame (assumeing all ...
    r_sites are
%in same quadrant)
if r_site1(1,1) < 0 && r_site1(2,1) < 0
    display('Sensor in quadrant 3')
    modifier=1;
elseif r_site1(1,1) > 0 && r_site1(2,1) > 0
    display('Sensor in quadrant 1')
    modifier=0;
elseif r_site1(1,1) < 0 && r_site1(2,1) > 0
    display('Sensor in quadrant 2')
    modifier=0;
elseif r_site1(1,1) > 0 && r_site1(2,1) < 0
    display('Sensor in quadrant 4')
    modifier=1;
end

%-------------------------------------------------------
% CONVERSION OF AZ EL TO RIGHT ASCENSION AND DECLINATION
if modifier == 0
x=[1,0,0]';
r_site1xy=[r_site1(1,1),r_site1(2,1),0]';

phi=acos(dot(r_site1xy,r_site1)/((norm(r_site1xy)*norm(r_site1))))...
    ;
theta=acos(dot(r_site1xy,x)/((norm(r_site1xy)*norm(x))));
SEZ=[sin(phi)*cos(theta),-sin(theta),cos(theta)*cos(phi);...
    sin(phi)*sin(theta),cos(theta),sin(theta)*cos(phi);...
    -cos(phi),0,sin(phi)];
L1=SEZ*L1;
r_site2xy=[r_site2(1,1),r_site2(2,1),0]';
```

```
    phi=acos(dot(r_site2xy,r_site2)/((norm(r_site2xy)*norm(r_site2))))...
        ;
    theta=acos(dot(r_site2xy,x)/((norm(r_site2xy)*norm(x))));
    SEZ=[sin(phi)*cos(theta),-sin(theta),cos(theta)*cos(phi);...
 86      sin(phi)*sin(theta),cos(theta),sin(theta)*cos(phi);...
         -cos(phi),0,sin(phi)];
    L2=SEZ*L2;
    r_site3xy=[r_site3(1,1),r_site3(2,1),0]';
    phi=acos(dot(r_site3xy,r_site3)/((norm(r_site3xy)*norm(r_site3))))...
        ;
 91 theta=acos(dot(r_site3xy,x)/((norm(r_site3xy)*norm(x))));
    SEZ=[sin(phi)*cos(theta),-sin(theta),cos(theta)*cos(phi);...
         sin(phi)*sin(theta),cos(theta),sin(theta)*cos(phi);...
         -cos(phi),0,sin(phi)];
    L3=SEZ*L3;
 96 elseif modifier == 1
         x=[1,0,0]';
    r_site1xy=[r_site1(1,1),r_site1(2,1),0]';

    phi=acos(dot(r_site1xy,r_site1)/((norm(r_site1xy)*norm(r_site1))))...
        ;
101 theta=2*pi-acos(dot(r_site1xy,x)/((norm(r_site1xy)*norm(x))));
    SEZ=[sin(phi)*cos(theta),-sin(theta),cos(theta)*cos(phi);...
         sin(phi)*sin(theta),cos(theta),sin(theta)*cos(phi);...
         -cos(phi),0,sin(phi)];
    L1=SEZ*L1;
106 r_site2xy=[r_site2(1,1),r_site2(2,1),0]';

    phi=acos(dot(r_site2xy,r_site2)/((norm(r_site2xy)*norm(r_site2))))...
        ;
    theta=2*pi-acos(dot(r_site2xy,x)/((norm(r_site2xy)*norm(x))));
    SEZ=[sin(phi)*cos(theta),-sin(theta),cos(theta)*cos(phi);...
111      sin(phi)*sin(theta),cos(theta),sin(theta)*cos(phi);...
         -cos(phi),0,sin(phi)];
    L2=SEZ*L2;
    r_site3xy=[r_site3(1,1),r_site3(2,1),0]';
    %x=[r_site3(1,1),0,0]';
116 phi=acos(dot(r_site3xy,r_site3)/((norm(r_site3xy)*norm(r_site3))))...
        ;
    theta=2*pi-acos(dot(r_site3xy,x)/((norm(r_site3xy)*norm(x))));
    SEZ=[sin(phi)*cos(theta),-sin(theta),cos(theta)*cos(phi);...
         sin(phi)*sin(theta),cos(theta),sin(theta)*cos(phi);...
         -cos(phi),0,sin(phi)];
121 L3=SEZ*L3;
    end
    %-------------------------------------------------------
    L=[L1,L2,L3];
    %-Figure out angles-
126 dec=asin(L3(3,1))*180/pi;
    csigmat=sqrt(1-L3(3,1));
    ra=asin(L3(2,1)/csigmat)*180/pi;
```

```
%Beginning Gaussian Method
tau1=(hour1*60+min1+s1/60)-(hour2*60+min2+s2/60);
tau3=(hour3*60+min3+s3/60)-(hour2*60+min2+s2/60);
tau1_sec=tau1*60;
tau3_sec=tau3*60;

a1=tau3/(tau3-tau1);
a3=-tau1/(tau3-tau1);
%----------------
%convert time to seconds
a1u=(tau3_sec*((tau3_sec-tau1_sec)^2-tau3_sec^2))/(6*(tau3_sec-...
    tau1_sec));
a3u=-(tau1_sec*((tau3_sec-tau1_sec)^2-tau1_sec^2))/(6*(tau3_sec-...
    tau1_sec));

invL=inv(L);
M=invL*r_site;
d1=M(2,1)*a1-M(2,2)+M(2,3)*a3;
d2=M(2,1)*a1u+M(2,3)*a3u;
C=L2'*(r_site2);
%Calculate the 8th order polynomial's roots
poly=[1 0 -(d1^2+2*C*d1+(r_site2'*r_site2)) 0 0 (-2*...
    u_earthkm3persec2*(C*d2+d1*d2)) 0 0 -u_earthkm3persec2^2*d2^2];
r2=roots(poly);
format long
display(r2)
r2=input('Enter the correct root choice numerically:  ');
u1=u_earthkm3persec2/r2^3;
c=[a1+a1u*u1;-1;a3+a3u*u1];
p=inv([c(1,1) 0 0;0 c(2,1) 0;0 0 c(3,1)])*M*-c;
pcheck=inv([c(1,1) 0 0;0 c(2,1) 0;0 0 c(3,1)])*M*-c;
magp=(sqrt(p(1,1)^2+p(2,1)^2+p(3,1)^2));

%we now have first guess at position vectors r1,r2,r3
r=[p(1,1)*L1+r_site1,p(2,1)*L2+r_site2,p(3,1)*L3+r_site3];
r1=r(:,1);
r2=r(:,2);
r3=r(:,3);


%We now use either the Gibbs or Herrick-Gibbs method to calculate ...
    the velocity vector
alpha1=acos((r1'*r2)/((sqrt(r2(1,1)^2+r2(2,1)^2+r2(3,1)^2))*(sqrt(...
    r1(1,1)^2+r1(2,1)^2+r1(3,1)^2))));
alpha3=acos((r3'*r2)/((sqrt(r2(1,1)^2+r2(2,1)^2+r2(3,1)^2))*(sqrt(...
    r3(1,1)^2+r3(2,1)^2+r3(3,1)^2))));
alpha_total=alpha1*180/pi+alpha3*180/pi;

if alpha_total < 5 %Herrick-Gibbs
    tau31=(hour3*3600+min3*60+s3)-(hour1*3600+min1*60+s1);
     tau21=(hour2*3600+min2*60+s2)-(hour1*3600+min1*60+s1);
    Z23=cross(r2,r3);
```

```
176         magr1=(sqrt(r1(1,1)^2+r1(2,1)^2+r1(3,1)^2));
        magr2=sqrt(r2(1,1)^2+r2(2,1)^2+r2(3,1)^2);
        magr3=(sqrt(r3(1,1)^2+r3(2,1)^2+r3(3,1)^2));
        %coplanar check
        alphacoplanar=90*pi/180-acos((Z23'*r1)/((sqrt(Z23(1,1)^2+Z23...
            (2,1)^2+Z23(3,1)^2))*(sqrt(r1(1,1)^2+r1(2,1)^2+r1(3,1)^2)))...
            );
181     if alphacoplanar > .001
            display('Vectors are not Coplanar!')
            break
        end
        v2=-tau3*(1/(tau21*tau31)+u_earthkm3persec2/(12*magr1^3))*r1+(...
            tau3-tau21)*(1/(tau21*tau3)+u_earthkm3persec2/(12*magr2^3))...
            *r2+tau21*(1/(tau3*tau31)+u_earthkm3persec2/(12*magr3^3))*...
            r3
186
    elseif alpha_total >= 5 %Gibbs
        Z12=cross(r1,r2);
        Z23=cross(r2,r3);
        Z31=cross(r3,r1);
191     %coplanar check
        alphacoplanar=90*pi/180-acos((Z23'*r1)/((sqrt(Z23(1,1)^2+Z23...
            (2,1)^2+Z23(3,1)^2))*(sqrt(r1(1,1)^2+r1(2,1)^2+r1(3,1)^2)))...
            );
        if alphacoplanar > .001
            display('Vectors are not Coplanar!')
            break
196     end
        N=((sqrt(r1(1,1)^2+r1(2,1)^2+r1(3,1)^2))*Z23+(sqrt(r2(1,1)^2+...
            r2(2,1)^2+r2(3,1)^2))*Z31+(sqrt(r3(1,1)^2+r3(2,1)^2+r3(3,1)...
            ^2))*Z12); %ER^3
        magr1=(sqrt(r1(1,1)^2+r1(2,1)^2+r1(3,1)^2));
        magr2=sqrt(r2(1,1)^2+r2(2,1)^2+r2(3,1)^2);
        magr3=(sqrt(r3(1,1)^2+r3(2,1)^2+r3(3,1)^2));
201     D=Z12+Z23+Z31;
        S=(magr2-magr3)*r1+(magr3-magr1)*r2+(magr1-magr2)*r3;
        B=cross(D,r2);
        Lg=sqrt(u_earthkm3persec2/(N'*D));
        v2=(Lg/magr2)*B+Lg*S; %ER/TU
206 end

    %needed orbital elements from r and v to get p1
    h=cross(r2,v2); %angular momentum
    magv2=(sqrt(v2(1,1)^2+v2(2,1)^2+v2(3,1)^2));
211 sigma=((magv2^2)/2)-(u_earthkm3persec2/magr2);
    e=1/u_earthkm3persec2*((magv2^2-(u_earthkm3persec2/magr2))*r2-(r2...
       '*v2)'*v2);
    mage=(sqrt(e(1,1)^2+e(2,1)^2+e(3,1)^2));
    a=1;
    p1=1;
216 if mage < 1
        a=-u_earthkm3persec2/(2*sigma);
```

```
        p1=a*(1-mage^2);
    elseif e >= 1
        display('Orbit is parabolic with eccentricity equal to or ...
           greater than 1');
221 end



    %now that we can determine the semi-parameter p1 we can get a new ...
       estimate
226 %of the slant range
    deltav1=-acos((r1/magr1)'*r2/magr2);
    deltav3=acos((r3/magr3)'*r2/magr2);
    v11=acos((e'*r1)/(mage*magr1));
    v22=acos((e'*r2)/(mage*magr2));
231 v33=acos((e'*r3)/(mage*magr3));
    f1=1-(magr1/p1)*(1-cos(deltav1));
    f3=1-(magr3/p1)*(1-cos(deltav3));
    g1=((magr1)*(magr2)*sin(deltav1))/sqrt(u_earthkm3persec2*p1);
    g3=((magr3)*(magr2)*sin(deltav3))/sqrt(u_earthkm3persec2*p1);
236 c1=g3/(f1*g3-f3*g1);
    c3=-g1/(f1*g3-f3*g1);
    p=inv([c1 0 0;0 c(2,1) 0;0 0 c3])*M*-c;
    pcheck1=inv([c1 0 0;0 c(2,1) 0;0 0 c3])*M*-c
    magp1=(sqrt(p(1,1)^2+p(2,1)^2+p(3,1)^2));
241

    %now that we have two guesses at slant range we need to iterate to...
        refine
    %the initial estimate of slant ranges
    count=1;
246 while abs(pcheck(1,1)-pcheck1(1,1)) > .1 && abs(pcheck(3,1)-...
       pcheck1(3,1)) > .1
        pcheck1=pcheck;
        r=[p(1,1)*L1+r_site1,p(2,1)*L2+r_site2,p(3,1)*L3+r_site3];
        r1=r(:,1);
        r2=r(:,2);
251     r3=r(:,3);
        %We now use either the Gibbs or Herrick-Gibbs method to ...
           calculate the velocity vector
        alpha1=acos((r1'*r2)/((sqrt(r2(1,1)^2+r2(2,1)^2+r2(3,1)^2))*(...
           sqrt(r1(1,1)^2+r1(2,1)^2+r1(3,1)^2))));
        alpha3=acos((r3'*r2)/((sqrt(r2(1,1)^2+r2(2,1)^2+r2(3,1)^2))*(...
           sqrt(r3(1,1)^2+r3(2,1)^2+r3(3,1)^2))));
        alpha_total=alpha1*180/pi+alpha3*180/pi;
256
        if alpha_total < 5 %Herrick-Gibbs
            tau31=(hour3*3600+min3*60+s3)-(hour1*3600+min1*60+s1);
            tau21=(hour2*3600+min2*60+s2)-(hour1*3600+min1*60+s1);
            Z23=cross(r2,r3);
261         %coplanar check
```

```
        alphacoplanar=90*pi/180-acos((Z23'*r1)/((sqrt(Z23(1,1)^2+...
            Z23(2,1)^2+Z23(3,1)^2))*(sqrt(r1(1,1)^2+r1(2,1)^2+r1...
            (3,1)^2))));
        if alphacoplanar > .0001
            display('Vectors are not Coplanar!')
            break
266     end
        v2=-tau3*(1/(tau21*tau31)+u_earthkm3persec2/(12*magr1^3))*...
            r1+(tau3-tau21)*(1/(tau21*tau3)+u_earthkm3persec2/(12*...
            magr2^3))*r2+tau21*(1/(tau3*tau31)+u_earthkm3persec2...
            /(12*magr3^3))*r3

    elseif alpha_total >= 5 %Gibbs
        Z12=cross(r1,r2);
271     Z23=cross(r2,r3);
        Z31=cross(r3,r1);
        %coplanar check
        alphacoplanar=90*pi/180-acos((Z23'*r1)/((sqrt(Z23(1,1)^2+...
            Z23(2,1)^2+Z23(3,1)^2))*(sqrt(r1(1,1)^2+r1(2,1)^2+r1...
            (3,1)^2))));
        if alphacoplanar > .0001
276         display('Vectors are not Coplanar!')
            break
        end
        N=((sqrt(r1(1,1)^2+r1(2,1)^2+r1(3,1)^2))*Z23+(sqrt(r2(1,1)...
            ^2+r2(2,1)^2+r2(3,1)^2))*Z31+(sqrt(r3(1,1)^2+r3(2,1)^2+...
            r3(3,1)^2))*Z12); %ER^3
        magr1=(sqrt(r1(1,1)^2+r1(2,1)^2+r1(3,1)^2));
281     magr2=sqrt(r2(1,1)^2+r2(2,1)^2+r2(3,1)^2);
        magr3=(sqrt(r3(1,1)^2+r3(2,1)^2+r3(3,1)^2));
        D=Z12+Z23+Z31;
        S=(magr2-magr3)*r1+(magr3-magr1)*r2+(magr1-magr2)*r3;
        B=cross(D,r2);
286     Lg=sqrt(u_earthkm3persec2/(N'*D));
        v2=(Lg/magr2)*B+Lg*S; %ER/TU
    end
    %needed orbital elements from r and v to get p1
    h=cross(r2,v2); %angular momentum
291 magv2=(sqrt(v2(1,1)^2+v2(2,1)^2+v2(3,1)^2));
    sigma=((magv2^2)/2)-(u_earthkm3persec2/magr2);
    e=1/u_earthkm3persec2*((magv2^2-(u_earthkm3persec2/magr2))*r2...
        -(r2'*v2)'*v2);
    mage=(sqrt(e(1,1)^2+e(2,1)^2+e(3,1)^2));
    a=1;
296 p1=1;
    if mage < 1
        a=-u_earthkm3persec2/(2*sigma);
        p1=a*(1-mage^2);
    elseif e >= 1
301     display('Orbit is parabolic with eccentricity equal to or ...
            greater than 1');
    end
```

```
        %now that we can determine the semi-parameter p1 we can get a ...
           new estimate
        %of the slant range
        deltav1=-acos((r1/magr1)'*r2/magr2);
306     deltav3=acos((r3/magr3)'*r2/magr2);
        v11=acos((e'*r1)/(mage*magr1));
        v22=acos((e'*r2)/(mage*magr2));
        v33=acos((e'*r3)/(mage*magr3));
        f1=1-(magr1/p1)*(1-cos(deltav1));
311     f3=1-(magr3/p1)*(1-cos(deltav3));
        g1=((magr1)*(magr2)*sin(deltav1))/sqrt(u_earthkm3persec2*p1);
        g3=((magr3)*(magr2)*sin(deltav3))/sqrt(u_earthkm3persec2*p1);
        c1=g3/(f1*g3-f3*g1);
        c3=-g1/(f1*g3-f3*g1);
316     p=inv([c1 0 0;0 c(2,1) 0;0 0 c3])*M*-c;
        pcheck1=inv([c1 0 0;0 c(2,1) 0;0 0 c3])*M*-c
        magp1=(sqrt(p(1,1)^2+p(2,1)^2+p(3,1)^2));
        count=count+1;
    end
321 [p,a,ecc,incl,omega,argp,nu,m,m_arglat] = rv2coe (r2,v2);
    incl=incl*180/pi
    omega=omega*180/pi
    m=m*180/pi
    argp=argp*180/pi
326 m_arglat=m_arglat*180/pi
    %calc estimated alt
     h_target=norm(r2)
     thetadot=sqrt(u_earthkm3persec2/(h_target)^3)
```

B.2 Spherical Trigonometry Initial Orbit Determination

Listing B.2: The spherical trigonometry initial orbit determination code. (appendix2/SphericaltrigIODm.m)

```
 1 % ...
     ------------------------------------------------------------------------

    %
    %                          Spherical_trig_IOD.m
    %
    %  this code is the initial orbit determination method using lat ...
       long data.
 6  %
    %  inputs          description                    range / units
    %       day=dataimport(:,1);
    %       month=dataimport(:,2);
    %       year=dataimport(:,3);
11  %       hour=dataimport(:,4);                       TIME
    %       min=dataimport(:,5);
    %       sec=dataimport(:,6);
    %       Lat=dataimport(:,7);                        GEODEDIC
    %       Long=dataimport(:,8);
16  %      CASE1: Circular ascending                    SYMBOL
```

```
%       CASE2: Circular descending
%       CASE3: Polar ascending
%       CASE4: Polar desending
%  outputs        :Omega = node
%                      I= inclination
%                      Theta=Argument of latitude
%                      Thetadot=change in theta
%                    INITIAL STATE
%              [Omega I Theta Thetadot]
%need JDAY.M, GSTIME.M, MODULO.M
% ...
    ------------------------------------------------------------------------


clc
format long

fid=fopen('ThorRB_latlong.txt','r');
dataimport = fscanf(fid, '%g %g %g %g %g %g %g %g', [8 inf]);
dataimport=dataimport';% It has two rows now.
fclose(fid)
day=dataimport(:,1);
month=dataimport(:,2);
year=dataimport(:,3);
hour=dataimport(:,4);
min=dataimport(:,5);
sec=dataimport(:,6);
Lat=dataimport(:,7);
Long=dataimport(:,8);


for i=1:length(Lat)
    degree_lat=Lat(i,1);
    degree_lat=degree_lat*pi/180;
    flatening=1/298.257223563;
    u=degree_lat;
    geocentric_lat=atan((1-flatening)^2*tan(u));
    geocentric_lat1=geocentric_lat*180/pi;
    Lat(i,1)=geocentric_lat1;
end

%There are 4 cases of data that we can consider using here.  We ...
    have
%CASE1: Circular ascending
%CASE2: Circular descending
%CASE3: Polar

% These cases are defined in terms of the track crossing the ...
    equator in the
% northern hemisphere

i=1;
```

```
A=zeros((length(Long)-1),4);
b=[];
%----------------CHOOSE ORBIT DATA TYPE-----------------
%syms case1 case2 case3
% ie choice=case1;
choice=3
%-------------------------------------------------------


if choice==1
for i=1:(length(Long)-1)
    deltalat=Lat(i+1,1)-Lat(i,1);
    deltalong=(Long(i+1,1)-Long(i,1))*cos(Lat(i,1)*pi/180);
    alpha=atan2(deltalat,deltalong);
    b(i,5)=alpha;
    az=90-(alpha*180/pi);
    b(i,3)=az;
    az=az*pi/180;

    %step 1 %Omega calc here
    delta=atan(sin(Lat(i,1)*pi/180)*tan(az));
    JD = JDAY(year(i,1), month(i,1), day(i,1), hour(i,1), min(i,1)...
        , sec(i,1));
    [gst] = GSTIME (JD);
    b(i,1)=gst*180/pi;
    b(i,2)=delta*180/pi;
    %need JDAY.M, GSTIME.M, MODULO.M
    Omega=Long(i,1)+gst*180/pi-delta*180/pi;
    A(i,1)=Omega;
    b(i,4)=Omega;
    %step 2
    theta=acos(cos(delta)*cos(Lat(i,1)*pi/180));
    theta=theta*180/pi;
    A(i,3)=theta;

    %step 3
    inclination=asin(sin(Lat(i,1)*pi/180)/sin(theta*pi/180));
    inclination=inclination*180/pi;
    A(i,2)=inclination;

    %step 4
    thetadot=sqrt((deltalong*pi/180)^2*(cos(Lat(i,1)*pi/180))^2+(...
       deltalat*pi/180)^2)/((hour(i+1,1)*3600+min(i+1,1)*60+sec(i...
       +1,1))-(hour(i,1)*3600+min(i,1)*60+sec(i,1)));
    A(i,4)=thetadot;
    i=i+1;
end
elseif choice==2
    for i=1:(length(Long)-1)
    deltalat=Lat(i+1,1)-Lat(i,1);
    deltalong=(Long(i+1,1)-Long(i,1))*cos(Lat(i,1)*pi/180);
    alpha=atan2(deltalat,deltalong);
```

```
        b(i,5)=alpha*180/pi;
        az=90-(alpha*180/pi);
116     beta=90+(alpha*180/pi);
        b(i,6)=beta;
        b(i,3)=az;
        az=az*pi/180;

121     %step 1 %Omega calc here
        delta=atan(sin(Lat(i,1)*pi/180)*tan(beta*pi/180));
        delta=pi-delta;
        JD = JDAY(year(i,1), month(i,1), day(i,1), hour(i,1), min(i,1)...
            , sec(i,1));
        [gst] = GSTIME (JD);
126     b(i,1)=gst*180/pi;
        b(i,2)=delta*180/pi;
        %need JDAY.M, GSTIME.M, MODULO.M
        Omega=Long(i,1)+gst*180/pi-delta*180/pi;
        A(i,1)=Omega;
131     b(i,4)=Omega;
        %step 2
        theta_tilde=acos(cos(delta)*cos(Lat(i,1)*pi/180));
        theta=theta_tilde*180/pi;
        A(i,3)=theta;
136
        %step 3
        inclination=asin(sin(Lat(i,1)*pi/180)/sin(theta_tilde));
        inclination=inclination*180/pi;
        A(i,2)=inclination;
141
        %step 4
        thetadot=sqrt((deltalong*pi/180)^2*(cos(Lat(i,1)*pi/180))^2+(...
            deltalat*pi/180)^2)/((hour(i+1,1)*3600+min(i+1,1)*60+sec(i...
            +1,1))-(hour(i,1)*3600+min(i,1)*60+sec(i,1)));
        A(i,4)=thetadot;
        i=i+1;
146     end
    elseif choice== 3
      for i=1:(length(Long)-1)
        deltalat=Lat(i+1,1)-Lat(i,1);
        deltalong=(Long(i+1,1)-Long(i,1))*cos(Lat(i,1)*pi/180);
151     alpha=atan2(deltalat,deltalong);
        b(i,5)=alpha;
        az=90-(alpha*180/pi);
        b(i,3)=az;
        az=az*pi/180;
156
        %step 1 %Omega calc here
        delta=atan(sin(Lat(i,1)*pi/180)*tan(az));
        JD = JDAY(year(i,1), month(i,1), day(i,1), hour(i,1), min(i,1)...
            , sec(i,1));
        [gst] = GSTIME (JD);
161     b(i,1)=gst*180/pi;
```

```
        b(i,2)=delta*180/pi;
        %need JDAY.M, GSTIME.M, MODULO.M
        Omega=Long(i,1)+gst*180/pi-delta*180/pi;
        A(i,1)=Omega;
166     b(i,4)=Omega;
        %step 2
        theta=acos(cos(delta)*cos(Lat(i,1)*pi/180));
        theta=theta*180/pi;
        A(i,3)=theta;
171
        %step 3
        inclination=asin(sin(Lat(i,1)*pi/180)/sin(theta*pi/180));
        inclination=180-inclination*180/pi;
        A(i,2)=inclination;
176
        %step 4
        thetadot=sqrt((deltalong*pi/180)^2*(cos(Lat(i,1)*pi/180))^2+(...
            deltalat*pi/180)^2)/((hour(i+1,1)*3600+min(i+1,1)*60+sec(i...
            +1,1))-(hour(i,1)*3600+min(i,1)*60+sec(i,1)));
        A(i,4)=thetadot;
        i=i+1;
181   end

    end% end if loop for choice
    display(A(i-1,:))
    display(['The epoch time is ' ,num2str(hour(i-1,1)), ':',num2str(...
        min(i-1,1)), ':' ,num2str(sec(i-1,1)),])
```

B.3 Non-Linear Least Squares

Listing B.3: The least squares code.(appendix2/NLleastsquares.m)

```
% ...
    ------------------------------------------------------------------------

%
%                       Non_linear_leastsquares.m
4 %
%  this code is the master Non-Linear Least Squares
%
%  inputs          description                    range / units
%       day=dataimport(:,1);
9 %       month=dataimport(:,2);
%       year=dataimport(:,3);
%       hour=dataimport(:,4);                       TIME
%       min=dataimport(:,5);
%       sec=dataimport(:,6);
14 %      Xo = INITIAL STATE=[Omega I Theta Thetadot]
%      EPOCH TIME
%  outputs       :Omega = node
%                    I= inclination
%                    Theta=Argument of latitude
19 %                 Thetadot=change in theta
```

```
%                   INITIAL STATE
%              [Omega I Theta Thetadot]
% FILES REQUIRED TO
% RUN;testdatasensor4.txt;testdatatarget4.txt;G.M;r_target.M;...
    cooridnate_tra
% nsform.m
% ...
    ------------------------------------------------------------------------

close all
clear all
clc
meanr=10;
%Cosmos
Xo=[358.9086913581474*pi/180,82.491055652040558*pi...
    /180,121.9992275169298*pi/180,.0009138421573219655];
hour_epoch=15;
 min_epoch=49;
 sec_epoch=59.74;

Xo=Xo';

%Gather Data
% Target
fid = fopen('Cosmos_p.txt', 'r');
dataimport = fscanf(fid, '%g %g %g %g %g %g %g %g %g', [9 inf]);
dataimport=dataimport';% It has two rows now.
fclose(fid)
day=dataimport(:,1);
month=dataimport(:,2);
year=dataimport(:,3);
hour=dataimport(:,4);
min=dataimport(:,5);
sec=dataimport(:,6);
rx=dataimport(:,7);
ry=dataimport(:,8);
rz=dataimport(:,9);

%Sensor
fid = fopen('Cosmos_sensor_p.txt', 'r');
dataimport = fscanf(fid, '%g %g %g %g %g %g %g %g %g', [9 inf]);
dataimport=dataimport';% It has two rows now.
fclose(fid)
day=dataimport(:,1);
month=dataimport(:,2);
year=dataimport(:,3);
hour=dataimport(:,4);
min=dataimport(:,5);
sec=dataimport(:,6);
Rx=dataimport(:,7);
Ry=dataimport(:,8);
Rz=dataimport(:,9);
```

```
[Az_ob, El_ob]=G(rx,ry,rz,Rx,Ry,Rz);
max_count=200;
count=1;
res_norm_old=100000;
res_norm_hist=[];
temp=size(Az_ob);
sigma = 0.0000001;
nvec1=sigma*randn(temp);
nvec2=sigma*randn(temp);
Az_ob=Az_ob+nvec1;
El_ob=El_ob+nvec2;
Q=diag(sigma^2*ones(length(Az_ob)*2,1));
Qinv=inv(Q);
ANS=2
  while ANS > 1
      count=count+1
%Begin Non Linear Least Squares
for i=1:length(Az_ob)

    % Compute current state from inital state & delta_t (time ...
       since epoch)
    delta_t=-(hour_epoch*3600+min_epoch*60+sec_epoch)+(hour(i,1)...
       *3600+min(i,1)*60+sec(i,1));
    phi=[1 0 0 0;0 1 0 0;0 0 1 delta_t;0 0 0 1];
    Xn=phi*Xo;

    %Calculate pseudo_phi using Xn
    Omega=Xn(1,1);
    I=Xn(2,1);
    Theta=Xn(3,1);
    Thetadot=Xn(4,1);
    t=delta_t;
    mue= 398601; % Earth Gravitational Parameter (km^3/sec^2)
    x1=Omega;
    x2=I;
    x3=Theta;
    x4=Thetadot;
    pseudo_phi = [(mue/x4^2)^(1/3)*(-sin(x1)*cos(x3)-cos(x1)*cos(...
       x2)*sin(x3)),...
        (mue/x4^2)^(1/3)*sin(x1)*sin(x2)*sin(x3),(mue/x4^2)^(1/3)...
           *...
        (-cos(x1)*sin(x3)-sin(x1)*cos(x2)*cos(x3)), -2/3/(mue/x4...
           ^2)^(2/3)...
        *(cos(x1)*cos(x3)-sin(x1)*cos(x2)*sin(x3))*mue/x4^3;...
                  (mue/x4^2)^(1/3)*(cos(x1)*cos(x3)-sin(x1)*cos(...
                     x2)*sin(x3)),...
                  -(mue/x4^2)^(1/3)*cos(x1)*sin(x2)*sin(x3), (mue...
                     /x4^2)^(1/3)...
                  *(-sin(x1)*sin(x3)+cos(x1)*cos(x2)*cos(x3)),...
                  -2/3/(mue/x4^2)^(2/3)*(sin(x1)*cos(x3)+cos(x1)...
                     ...
```

```
                    *cos(x2)*sin(x3))*mue/x4^3;...
                    0, (mue/x4^2)^(1/3)*cos(x2)*sin(x3),...
114                 (mue/x4^2)^(1/3)*sin(x2)*cos(x3),...
                    -2/3/(mue/x4^2)^(2/3)*sin(x2)*sin(x3)*mue/x4...
                       ^3];

        %Using Xn calculate r_target
        r_target1 = r_target(Xn);
119     Ro=[Rx(i,1);Ry(i,1);Rz(i,1)];
        rho_eci=r_target1-Ro;
        R=Ro;
        R_transform = coordinate_transform(R);
        rho_sez=R_transform*rho_eci;
124     ps=rho_sez(1,1);
        pe=rho_sez(2,1);
        pz=rho_sez(3,1);

        %Obtain Az_calc and El_calc
129     Az=atan2(pe,-ps);
        El=asin(pz/sqrt(ps^2+pe^2+pz^2));
        Az_calc(i,1)=Az;
        El_calc(i,1)=El;

134     %Calculate H
        H_tilde= [pe/(pe^2+ps^2), -ps/(pe^2+ps^2), 0;...
                 -pz/(pe^2+ps^2+pz^2)^(3/2)*ps/(1-pz^2/(pe^2+ps^2+pz...
                    ^2))^(1/2), -pz/(pe^2+ps^2+pz^2)^(3/2)*pe/(1-pz...
                    ^2/(pe^2+ps^2+pz^2))^(1/2),(pe^2+ps^2)/(pe^2+ps^2+...
                    pz^2)^(3/2)/((pe^2+ps^2)/(pe^2+ps^2+pz^2))^(1/2)];
        H=H_tilde*R_transform;

139     %Calculate T
        T=H*pseudo_phi*phi;
        if i==1
            BigT=T;
        else
144         BigT=[BigT;T];
        end;

        %Calculate residuals
        residual=zeros(2,1);
149     residual(1,1)=Az_ob(i,1)-Az_calc(i,1);
        residual(2,1)=El_ob(i,1)-El_calc(i,1);
        if i==1
            BigR=residual;
        else
154         BigR=[BigR;residual];
        end
    end

    % Update State Estimate (at epoch)
159 P=inv(BigT'*Qinv*BigT);
```

```
delta_X=P*BigT'*Qinv*BigR;
Xo=Xo+delta_X.*[0.2 0.2 0.2 0.2]';

% Check for convergence by looking at percent change in the ...
    residual norm
res_norm=norm(BigR)
res_norm_hist=[res_norm_hist;res_norm];
if abs(res_norm-res_norm_old)/res_norm<0.0001
    ANS =1;
    disp('converged')
end

if count>max_count
    ANS=1;
    disp('Exceeded Max Count')
end

res_norm_old=res_norm;

  end
  Xo(1,1)=Xo(1,1)*180/pi;
  Xo(2,1)=Xo(2,1)*180/pi;
  Xo(3,1)=Xo(3,1)*180/pi;
display(Xo)
display(P)

%The loops below are seperating out azimuth and elevation ...
    residuals
for i=1:length(BigR)
    odd=i*2-1;
    odd1(i,1)=odd;
    even=i*2-2;
    even1(i,1)=even;
end

for i=1:length(BigR)
    for a=1:length(even1)
    if i==even1(a,1)
       el=BigR(i,1);
       el1(a,1)=el;
      end
    end
end
for i=1:length(BigR)
    for a=1:length(odd1)
    if i==odd1(a,1)
       Az=BigR(i,1);
       Az1(a,1)=Az;
      end
    end
end
```

This appendix contains all function code referenced in the core code in appendix 1.

C.1 Observation Function

Listing C.1: The observation function G used in NLleastsquares.m. (appendix3/G.m)

```
% ...
    ------------------------------------------------------------------------

%
%                           function G.m
%
%  this is the observation funtion for least squares.
%
%  inputs          description                    range / units
%    rx,ry,rz    - position of target             ECI KM
%    Rx,Ry,Rz    -position of sensor              ECI KM
%  outputs       :
%    Az_ob          Observed Azimuth              radians
%    El_ob          Observed Elevation            radians

% [Az_ob, El_ob]=G(day,month,year,hour,sec,rx,ry,rz,Rx,Ry,Rz)
% files needed to run: coordinate_transform

%[Az_ob, El_ob]=G(rx,ry,rz,Rx,Ry,Rz)
% ...
    ------------------------------------------------------------------------



function [Az_ob, El_ob]=G(rx,ry,rz,Rx,Ry,Rz)


%Form vectors in ECI
R=[Rx,Ry,Rz];
r=[rx,ry,rz];

%rho in the eci
for i=1:length(R)
    A=r(i,:)-R(i,:);
    rho_eci(i,:)=A;
end

%rho in the sez frame
for i=1:length(rho_eci)
    R_transform = coordinate_transform(R(i,:)');
    B=R_transform*rho_eci(i,:)';
    rho_sez(i,:)=B;
end
```

```
%generate az and el data
ps=rho_sez(:,1);
pe=rho_sez(:,2);
pz=rho_sez(:,3);
for i=1:length(rho_sez)
    F=atan2(pe(i,1),-ps(i,1));
    Az(i,1)=F;
end

for i=1:length(rho_sez)
    B=asin(pz(i,1)/sqrt(ps(i,1)^2+pe(i,1)^2 + pz(i,1)^2));
    El(i,1)=B;
end
Az_ob=Az;
El_ob=El;
```

C.2 Target function

Listing C.2: The target function.(appendix3/rtarget.m)

```
% ...
    ------------------------------------------------------------------------

%
%                           function r_target.m
%
%  this function finds the calculated position of the target ...
   Satellite from Xn.
%
%  inputs          description                    range / units
%    Xn            State Vector at time n         varied
%    r             height value being used        KM
%
%  outputs       :
%    r_target      target position in ECI fram    KM

% r_target = r_target(Xn,r,mue)
% ...
    ------------------------------------------------------------------------


function r_target1 = r_target(Xn)

    mue=398601;
    Omega=Xn(1,1);
    I=Xn(2,1);
    Theta=Xn(3,1);
    Thetadot=Xn(4,1);
RIn=[1 0 0;
    0 cos(I) -sin(I);
    0 sin(I) cos(I)];
```

```
ROmegan=[cos(Omega) -sin(Omega) 0;
    sin(Omega) cos(Omega) 0;
    0 0 1];
R_pqw=[cos(Theta);
    sin(Theta);
    0];
n=(mue/Thetadot^2)^(1/3);
r_target1=ROmegan*RIn*R_pqw*n;
```

C.3 Julian Day

Listing C.3: The julian day function for reference.(appendix3/JDAY.m)

```
% ...
    ------------------------------------------------------------------------

%
%                           function jday.m
%
%  this function finds the julian date given the year, month, day,...
    and time.
%
%  inputs          description                    range / units
%    year        - year                           1900 .. 2100
%    mon         - month                          1 .. 12
%    day         - day                            1 .. 28,29,30,31
%    hr          - universal time hour            0 .. 23
%    min         - universal time min             0 .. 59
%    sec         - universal time sec             0.0d0 .. 59.999...
    d0
%
%  outputs       :
%    jd          - julian date                    days from 4713 ...
    bc

% jd = jday(year, month, day, hour, min, sec)
% ...
    ------------------------------------------------------------------------


function JD = jday(year, month, day, hour, min, sec)

JD=367*year-floor((7*(year+floor((month+9)/12)))/4)+floor((275*...
    month)/9)+day+1721013.5+((((sec/60)+min)/60)+hour)/24;
```

C.4 Greenwich Mean Sidereal Time

Listing C.4: The θ_{GMST} function .(appendix3/GSTIME.m)

```
% ...
    ------------------------------------------------------------------------

%                           function gstime
```

```
%
%  this function finds the greenwich sidereal time.
%
%  inputs          description                    range / units
%    jdut1       - julian date of ut1             days from 4713 ...
    bc
%
%  outputs       :
%    gst         - greenwich sidereal time        0 to 2pi rad
%
%  locals        :
%    temp        - temporary variable for reals   rad
%    tut1        - julian centuries from the
%                  jan 1, 2000 12 h epoch (ut1)
%  references    :
%    vallado       2001, 191, eq 3-45
% gst = gstime(JD);
% ...
    ------------------------------------------------------------------------


function gst = gstime(JD);
        twopi       = 2.0*pi;
        deg2rad     = pi/180.0;

        % ------------------------  implementation   ...
            ------------------
        tut1= ( JD - 2451545.0 ) / 36525.0;

        temp = - 6.2e-6 * tut1 * tut1 * tut1 + 0.093104 * tut1 * ...
           tut1  ...
               + (876600.0 * 3600.0 + 8640184.812866) * tut1 + ...
                 67310.54841;

        temp = modulo( temp*deg2rad/240.0,twopi );

        % ------------------------ check quadrants ...
           ---------------------
        if ( temp < 0.0 )
            temp = temp + twopi;
          end

        gst = temp;
```

C.5 Coordinate Transform

Listing C.5: The coordinate transform function.
(appendix3/coordinatetransform.m)

```
% ...
    ------------------------------------------------------------------------

%
```

```
%                           function coordinate_transform.m
%
%  this function finds the coordinate transform from ECI to SEZ.
%
%  inputs          description                    range / units
%     R              position vector of sensor      KM ECI
%
%  outputs       :
%    R_transform     coordinate transfrom          unitless

% R_transform = coordinate_transform(R)
% ...
    ------------------------------------------------------------------------.


function R_transform = coordinate_transform(R)

%Coordinate Transform
z=R/norm(R);
kvec=[0;0;1];
b=cross(kvec,R);
b1=norm(b);
e=b/b1;
s=cross(e,z);
C=[s';e';z'];
R_transform=C;
```

C.6 Modulus

Listing C.6: Simple but necessary modulus function.(appendix3/modulo.m)

```
% ...
    ------------------------------------------------------------------------..

%
%                             modulo function
%
%  this function finds the modulous value. The result is positive ...
   or negative.
%
%  inputs          description                    range / units
%    x           - input argument
%    xv          - input argument to mod with
%
%  outputs       :
%    y           - answer
% y = modulo (x,xv);
% ...
    ------------------------------------------------------------------------..


function y = modulo (x,xv);
        y = x - xv * fix(x / xv);
```

C.7 Search Algorithm

Listing C.7: The search algorithm uses a modified single line version of the catalog. (appendix3/simplesearch.m)

```
clear all
clc
format long

%fid = fopen('2ndlinecatalog.txt', 'r');%the one without debris
fid = fopen('catalog2ndlinewdebris.txt', 'r');%the one with debris
dataimport = fscanf(fid, '%g %g %g %g %g %g %g %g', [8 inf]);
dataimport=dataimport';% It has two rows now.
fclose(fid)
line=dataimport(:,1);
designator=dataimport(:,2);
inclination=dataimport(:,3);
raan=dataimport(:,4);
eccentricity=dataimport(:,5);
argument_of_perigee=dataimport(:,6);
mean_anomaly=dataimport(:,7);
mean_motion=dataimport(:,8);

fid = fopen('2ndlinecatalog_epoch.txt', 'r');
dataimport = fscanf(fid, '%14g', [1 inf]);
dataimport=dataimport';% It has two rows now.
fclose(fid)
Epoch=dataimport(:,1);
figure
hold on
plot(inclination,raan,'.')
title('Inclination vs RAAN')
xlabel('Inclination')
ylabel('RAAN')
set(gca,'XTick',0:10:150)
hold off


found=0;
for i=1:length(line)
    if inclination(i,1) > 98.5 && inclination(i,1) < 99.5 && raan(...
       i,1) > 65.4 && raan(i,1) < 66.4
        found=found+1;
        match(found)=i;
    end
end
if found==0, disp('No matches'),end
if found~=0
    disp(['Found ',num2str(found),' matches.'])
    disp([' '])
    for i=1:found
        disp(['Satellite Number:   ', num2str(designator(match(1,i...
           ),1))])
```

```
47          disp(['Inclination:          ', num2str(inclination(match(1,...
                i),1))])
            disp(['RAAN:                 ', num2str(raan(match(1,i),1))...
                ])
            disp(['Eccentricity:         ', num2str(eccentricity(match...
                (1,i),1))])
            disp(['Argument of Perigee:', num2str(argument_of_perigee(...
                match(1,i),1))])
            disp(['Mean Anomaly:         ', num2str(mean_anomaly(match...
                (1,i),1))])
52          disp(['Mean Motion:          ', num2str(mean_motion(match(1,...
                i),1),'%2.8f')])
            disp([' '])
        end
    end
```

Bibliography

1. Briggs, RE and JW Slowley. "An Iterative Method of Orbit Determination from Three Observations of a Nearby Satellite". *SAO Special Report*, 27(part 1), 1959.
2. Burns, R.E. "Solution of the Angles-Only Satellite Tracking Problem". *NASA Technical Paper*, 3667:25, 1997.
3. Escobal, P.R. *Methods of Orbit Determination*. Wiley New York, 1965.
4. Gooding, RH. "A New Procedure for Orbit Determination Based on Three Lines of Sight (Angles Only)". *DRA/RAE Technical Report*, 93004, 1993.
5. Gooding, RH. "A new procedure for the solution of the classical problem of minimal orbit determination from three lines of sight". *Celestial Mechanics and Dynamical Astronomy*, 66(4):387–423, 1996.
6. Kristensen, LK. "Orbit Determination by Four Observations". *ASTRONOMISCHE NACHRICHTEN*, 316(4):261, 1995.
7. Li, Q., F. Guo, J. Li, and Y. Zhou. "Research of Satellite-to-Satellite Passive Tracking Using Bearings-Only Measurements In J2000 ECI Frame". *Radar, 2006. CIE'06. International Conference on*, 1–4, 2006.
8. Marsden, BG. "Initial orbit determination- The pragmatist's point of view". *The Astronomical Journal*, 90:1541, 1985.
9. OSEDACZ, R. "Orbit determination of sunlight illuminated objects detected by overhead platforms(M. S. Thesis)". 1989.
10. Prussing, J.E. and B.A. Conway. *Orbital Mechanics*. Oxford University Press New York, 1993.
11. SABOL, C., K.T. ALFRIEND, and D. WIESE. "Angles-only orbit updates for low-earth-orbit satellites". *Journal of guidance, control, and dynamics*, 30(2):314–321, 2007.
12. Sabol, C. and R. Culp. "Improved angular observations in geosynchronous orbit determination". *Journal of Guidance, Control, and Dynamics*, 24(1):123–130, 2001.
13. Taff, LG. "On initial orbit determination". *Astronomical Journal*, 89:1426–1428, 1984.
14. Tapley, B., B. Schutz, and G. Born. *Statistical Orbit Determination*. Elsevier Academic Press, Boston, 2004.
15. Vallado, D.A. *Fundamentals of Astrodynamics and Applications*. Kluwer Academic Pub, 2001.

16. Wiesel, William E. *Spaceflight Dynamics*. McGraw-Hill, Boston, 1989.

17. Wiesel, William E. *Modern Methods of Orbit Determination*. Apahelion Press, BeaverCreek Ohio, 2003.

Index

The index is conceptual and does not designate every occurrence of a keyword. Page numbers in bold represent concept definition or introduction.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
10–03–2008	Master's Thesis	Sept 2007 — Mar 2008

4. TITLE AND SUBTITLE

Space Based Satellite Tracking and Characterization Utilizing Non-Imaging Passive Sensors

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Bradley R. Townsend, CPT, USA

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GA/ENY/08-M06

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

A technique is developed to determine the orbit of a sunlight illuminated satellite passing through the field-of-view of a sensor platform in a Highly Elliptical Orbit (HEO) and Geosynchronous orbit (GEO). The technique develops two different methods of initial orbit determination. The first relies on the Gauss initial orbit determination method to develop an estimate of the state from angular data. The second method relies on positional data of the target relative to the Earth's background to determine an estimate of the state. These estimates are then refined in a non-linear least squares routine. This estimate of the state is then used to identify the target from the Air Force Space Command satellite catalog.

15. SUBJECT TERMS

Gauss, fastwalker, Angles only, Orbit Determination, Highly Elliptical Orbit, Geosynchronous orbit

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			LTC Nathan Titus
U	U	U	UU	111	19b. TELEPHONE NUMBER *(include area code)* (937) 785–3636, ext 7469