3-9-2004

Data Sorting and Orbit Determination of Tethered Satellite Systems

Mark J. Faulstich

DATA SORTING AND ORBIT DETERMINATION OF TETHERED SATELLITE SYSTEMS

THESIS

Mark J. Faulstich, Captain, USAF

AFIT/GSS/ENY/04-M03

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, Department of Defense, or the U.S. Government.

AFIT/GSS/ENY/04-M03

DATA SORTING AND ORBIT DETERMINATION OF TETHERED SATELLITE SYSTEMS

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science (Space Systems)

Mark J. Faulstich, BS

Captain, USAF

March 2004

AFIT/GSS/ENY/04-M03

DATA SORTING AND ORBIT DETERMINATION OF TETHERED SATELLITE SYSTEMS

Mark J. Faulstich, BS

Captain, USAF

Approved:

//signed//	9 March 04
Dr. Steven Tragesser (Chairman)	date
//signed//	9 March 04
Dr. William Wiesel (Member)	date
//signed//	9 March 04
Joerg Walter, Maj, USAF (Member)	date

Abstract

Tethered satellite system end masses do not obey the normal laws of motion developed for determining their orbits. In addition, tethered satellite systems cause unique problems for satellite tracking because there are potentially two or more objects which may be tracked.

This thesis provides insight into these issues by developing a method of sorting out observation data of tethered satellite systems into their appropriate end mass and providing an estimate on the center of mass orbit of the tethered satellite system. The method used to accomplish both of these tasks is optimization of an estimated simulated orbit. This orbit estimate is optimized to provide the minimum difference between the end mass position estimates and the observations obtained from one or more tracking sites. This methodology also helps provide a baseline for tracking tethered satellite systems more accurately in the future.

Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Dr. Steven Tragesser, for his guidance and support throughout the course of this thesis effort. The insight and experience was certainly appreciated. I would also like to thank my sponsor, Dr. Robert Racca, from the Space Warfare Center Analysis and Engineering Division for both the support and latitude provided to me in this endeavor.

Mark J. Faulstich

.

Table of Contents

Page

Abstract ……..iv

Acknowledgements ……v

List of Figures ……..viii

I. Introduction ……1

Background ……1
Problem Statement ……..2
Research Objectives ……3
Methodology ……...4
Preview ……...4

II. Literature Review ……..6

Tethered Satellite Orbit Determination ……...6
Identification of Tethered Satellite Systems ……...7
TSS Data Sorting Problems ……8

III. Data Sorting Preliminary Analysis ……..……..10

Elevation and Libration Analysis on TS Slant Range and EC Range ….……10
Tracking Site Error Analysis ……15
Data Sorting Preliminary Analysis Conclusion …….……...19

IV. Estimating the TSS Orbit and Sorting the Data ……21

Assumptions ……..22
Obtaining an Initial Estimate of the COEs of the CM ……...……...25
Calculating Estimated EC pqw Position Vectors …….....32
Propagating Estimated CM COEs for all Observation Times ……32
Converting Estimated CM COEs into
Estimated CM IJK Position Vectors ……35
Calculating Estimated TSS End Mass pqw Position Vectors ……...36
Converting Observation Data into pqw Position Vectors …….……39
Minimizing Observed and Estimated Differences ……40
Calculating TSS Position Residuals ……40
Optimizing the COEs and Data Sorting ……...……42
Sources of TSS Observation Data ……45

Page

Estimation and Data Sorting Conclusion …..……..50

V. Results ……..……52

Baseline Case ……52
Baseline Case Parameters ……53
Baseline Case Estimation ……..54
Comparing Baseline Case Results and Tracking Site Error Analysis …..58
Parametric Studies of Different TSS and Tracking Site Parameters ……60
Time/Number of Observations Variations ……61
Tether Length/Elevation Angle Variations ……..……..62
Single Object Only Data ……..……..63
TETHERSIM^{TM} Results ……..……..65
Unknown TSS ……...……67
Single-Body Satellite System ……...……67
Unknown Baseline Case TSS ……...……68
Single End Mass Observations for an Unknown TSS ……...……70
Real-World TiPS Results ……...…73

VI. Conclusion ……...……..78
Operational Implications ……...78
Future Research ……79

Appendix: Primary MATLAB Programs ……..82

Bibliography ……...110

List of Figures

Figure Page

1. $\hat{u}_1$ Coordinate Component Defined ……...…….11

2. Slant Range from Tracking Site to End Masses ……..12

3. Affect of Elevation Angle on s_r …….17

4. Affect of Larger Slant Range Error on s_r ……..18

5. Relationship Between $\hat{u}_1$ and IJK Coordinate Frame …….37

6. Baseline Case Plot of Observed versus Estimated EC Position Vector Magnitudes ...58

7. Error Analysis for Lowest Elevation Tracking Data for Baseline Case ……..59

8. Bottom Mass Only Data for Baseline Case ……....63

9. Bottom Mass Only Data for Baseline Case Showing Best of 3 Optimizations ……...64

10. TETHERSIM^{TM} Known TSS Mixed Data Results ……....66

11. Baseline Case Unknown TSS Results ……....69

12. 20 km Unknown TSS Results ……71

13. TiPS Tracking Site Error Analysis ……74

14. TiPS EC Position Vector Magnitude Plot ……..75

DATA SORTING AND ORBIT DETERMINATION OF TETHERED SATELLITE SYSTEMS

I. Introduction

Background

A Tethered Satellite System (TSS) can provide unique capabilities over a single-body satellite. Some of the potential applications of a TSS include power generation and orbital reboost (Beletsky and Levin., 1993:20). In recent years, some of these ideas have started to become reality as tethered satellites are deployed in space. As TSS's become more of a reality, it is important to understand how to do proper orbit determination (OD) for these objects. In order to accomplish this it is also important to understand how to properly identify which end mass is being tracked since there are generally two or more end masses connected to a TSS.

Orbit determination of a TSS requires different techniques than orbit determination of a single-body satellite. The normal Keplerian equations of motion that apply to a single satellite do not work for the end masses of a TSS. In fact, if normal

Keplerian techniques are applied to a TSS, under the right conditions it may appear as if one of the end masses is on a suborbital trajectory with the Earth (Lovell et al., 2000:1). For obvious reasons this provides motivation for wanting to properly identify the orbit of a TSS.

Although the subject of tethered satellite OD has received much attention in recent years, data sorting of TSS tracking data has received less attention. But, if TSS tracking data is not tagged to the appropriate end mass then calculating an accurate estimate of a TSS orbit becomes extremely difficult, if not impossible. For this reason, data sorting and OD of a TSS are closely related tasks.

Data sorting and OD of a TSS is more difficult if it is not known whether the satellite system is a TSS, or if the key parameters of the TSS are unknown. Again, if normal techniques of orbit determination are applied to a TSS without some knowledge about the system the orbit prediction will not be very accurate.

With all of these items in mind, this research effort was undertaken to provide insight into a possible method of data sorting and doing OD for a known or unknown TSS only utilizing radar tracking data. This type of data is the most commonly available measurement for satellite tracking purposes.

Problem Statement

There are two main problems this research addresses. The first problem takes a known TSS and determines if it is possible to allocate a given radar measurement to the

appropriate end mass and provide a good estimate of the TSS orbit. The second problem is similar to the first, but the distinction is the TSS is now an unknown system. Both problems use raw radar observation data to include slant range, elevation, and azimuth, but not range rate.

Research Objectives

As given in the problem statement, there are two main research objectives for this thesis. The first objective is to sort out raw radar observation data of either a known or unknown TSS into the appropriate end mass; i.e. assign an identification tag to each radar observation. Typically, this is not an issue with single-body satellite systems because they only have one object to observe. As far as a TSS is concerned, however, the end masses may not be easily distinguishable depending on the length of the TSS.

The second research objective of this thesis is to determine an approximate orbit for the TSS. This does not mean identifying an orbit for each end mass, but, instead, determining an orbit for the Center of Mass (CM) of the TSS. Identifying the CM orbit of a TSS is extremely useful because the CM orbit essentially follows a normal Keplerian orbit (Cochran et al., 2000:478). Therefore, if the CM orbit is identified, standard orbit propagation techniques can be applied to determine where the CM will be at some time in the future. Even though knowing the position of the CM at some time in the future does not precisely determine where each of the end masses is located at that time, it significantly narrows down the search space where the end masses might be located.

Methodology

The methodology used in this research focused mainly on determining estimated Classical Orbital Elements (COEs) of the CM of the TSS. The estimated, or best-fit, COEs of the CM were determined using parameter optimization. The COEs for the CM were optimized by determining the best-fit of all available observations of the end masses compared to the estimated location of the CM and end masses.

As a part of this estimation process the next part of the research was accomplished by determining which end mass each observation represented. This was done by sorting the observations based on each observation's Earth-Centered (EC) position vector magnitude and the estimated EC position vector magnitudes of the end masses at any particular time. Since this method of sorting the observations relies on the accuracy of the estimated COEs of the CM, then, in general, a better COE estimate provides better sorting results.

Preview

This thesis is divided into four additional chapters. Chapter two reviews some of the most pertinent literature related to the topic of tethered satellites. Specifically, the type of literature reviewed will concentrate on showing what research efforts have been accomplished in the past relating to data sorting and OD of TSS data. Chapter three covers the preliminary analysis done to determine the best parameter to use for data

sorting. Chapter four covers in-depth the methodology used in accomplishing the research goals of data sorting and OD for TSS data. This methodology will discuss the main assumptions and the specific techniques used during the estimation process. Chapter five discusses and analyzes the results of several specific cases looked at during this research effort. The results show the strengths of this technique for data sorting and TSS OD, and they also show where this technique starts to break down. Finally, chapter six concludes by discussing the operational relevance of this particular research effort and also covers potential areas of further research on this topic.

II. Literature Review

Tethered Satellite Orbit Determination

There are two main approaches to modeling a TSS for OD. The first approach uses the full equations of motion for tethered satellites to include all of the libration and tension affects. The second approach attempts to simplify the equations of motion by trying to separate the orbital motion from the attitude dynamics of the system.

Numerous people have analyzed the full equations of motion of tethered satellite systems. One of the most commonly accepted sets of dynamical equations for a TSS comes from Beletsky and Levin's *Dynamics of Space Tether Systems*, Volume 83 in the *Advances in the Astronautical Sciences*, written in 1993. This publication covers in-depth the equations of motion and it also covers many other items of concern related to tethers to include perturbations and some of the potential uses of a TSS (Beletsky and Levin, 1993:20). The most significant understanding that comes from this publication is the coupling which occurs between orbital motion and attitude motion for a TSS. This coupling of the equations is what causes most of the difficulties in TSS orbit and attitude determination. Specifically, coming up with approximate analytical solutions for TSS OD is extremely difficult, if not impossible in some cases.

The difficulties caused by the coupling of the dynamics of TSS's are the cause for many people looking at ways to decouple the attitude dynamics from the orbital

dynamics. One of the most commonly accepted methods of accomplishing this decoupling is to assume the TSS remains nadir-oriented at all times. This assumption ignores the librational motion of the TSS. This technique has been used in the past with some success for an actual TSS. The Tether Physics and Survivability (TiPS) experiment was launched on 20 June 1996 and still continues to fly today. A team of people at the Naval Research Laboratory usedd an assumed nadir-oriented tether initially to help analyze the orbital motion of TiPS separately from the attitude (Purdy et al., 1997:3).

Identification of Tethered Satellite Systems

Extensive research has been accomplished in the last few years in trying to determine a method for identifying whether an object in space is part of a TSS. Due to the importance of this task, several methods of identifying tethered satellites have arisen. These methods often use different types of filters to try and determine specific parameters that help identify whether a tracked object is part of a TSS. For example, one method attempts to calculate the radial and tangential force components on the tracked object that would be caused by a tether which is attached to the object (Cicci et al., 2001:314). If these additional forces are calculated to be above a certain level, then it is assumed that the object is part of a TSS because a single-body satellite should not have these additional forces present.

The aforementioned method of TSS identification, as well as several other methods currently in existence, usually assume the data points fed into the filter all come

from the same object. If this is true, these methods may show promise in identifying a TSS. However, if some of the data points happen to come from the other end mass of the TSS, then the calculations may be thrown off enough to improperly identify whether an object is part of a TSS. This is why data sorting becomes extremely important.

TSS Data Sorting Problems

The data sorting problem, as it relates to TSS's, was not discussed in-depth until the TiPS experiment was launched and tracked in 1996. Since one of the main purposes of TiPS was to help provide understanding of the long-term dynamics of a TSS, a large amount of tracking data was collected over the years following its launch. During the analysis of this tracking data, specifically radar tracking data, it became apparent that a significant portion, approximately 30%, of the data was not tagged, or sorted, properly into the appropriate end mass (Barnds and Coffey, 1999:1846). This occurred when it was known that the system was a TSS, and all of the key parameters of the TSS were known. Presumably with an unknown TSS the possibility of inappropriately sorting tracking data becomes greater.

The analysis team involved in sorting out the TiPS tracking data was able to approach the problem with much more certainty because of the use of many other sources of data available to them. They were able to take sources such as Satellite Laser Ranging data and optical tracking data to come up with a more complete picture of what was going on with the satellite (Barnds and Coffey, 1999:1845). Their analysis of multiple

tracking sources helped the team come up with a very accurate estimate of the TiPS orbit as well as determine an approximation on the libration and libration rate associated with TiPS over time.

The results of the TiPS analysis shows that data sorting and OD for TSS's can be done with very good accuracy, but these results were achievable only with multiple types of tracking data, a large set of data, and much was known about the system. Coming up with a way of doing data sorting on a much shorter timeframe with only radar tracking data is still an issue.

III. Data Sorting Preliminary Analysis

Two parameters were initially considered as possibilities for accomplishing data sorting of TSS observations: tracking site (TS) slant range and Earth-Centered (EC) position vector magnitude. For brevity, EC position vector magnitude and EC range will be synonomous for the rest of this paper. An analysis was done to determine which of these two parameters more easily distinguishes the two end bodies of a TSS.

When a TS tracks an object in space it generally obtains a reading for slant range, elevation angle, and azimuth angle. Slant range is the distance from the site to the object being tracked. Elevation is defined as the angle between the TS's local horizon and the location of the satellite, and it can assume any value between 0 and 90 degrees. Finally, azimuth is an angle measured clockwise from north to the direction where the TS is tracking. This means azimuth can assume any angle between 0 and 360 degrees (Bate et al., 1971:84).

Elevation and Libration Analysis on TS Slant Range and EC Range

The first parameter we analyzed to determine if it had good properties for data sorting was the TS slant range. If it can be shown that the lower end mass (mass 1) of a TSS almost always has a smaller TS slant range than the upper end mass (mass 2) then

this would be a good parameter to use for data sorting. This requires us to do a comparison between the TS slant ranges for both end masses. In order to calculate the TS slant ranges we must define tether libration. Figure 1 shows a coordinate component that defines the tether libration.

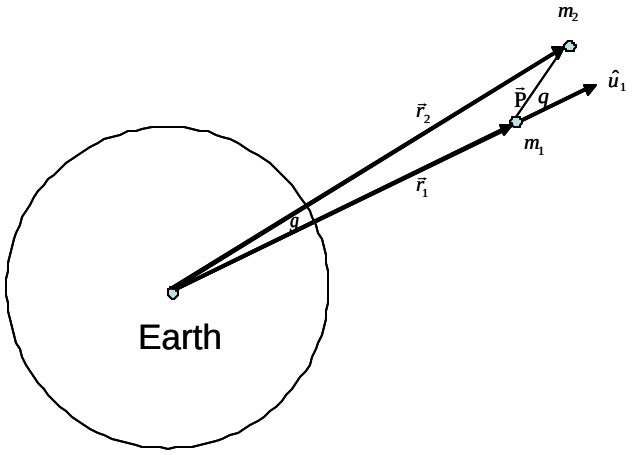

Figure 1- $\hat{u}_1$ Coordinate Component Defined

The $\hat{u}_1$ coordinate component is defined such that its origin is located at the center of the Earth and $\hat{u}_1$ goes from the origin at the center of the Earth and passes through the lower end mass, m_1, of the TSS (Cicci et al., 2001:311). Therefore, $\vec{r}_1$, the vector from the center of the Earth to m_1 is only defined in the $\hat{u}_1$ direction. Also, $|\vec{r}_1|$ is the EC range for m_1, and $|\vec{r}_2|$ is the EC range for m_2. The angle, ?, defines the angle between the $\hat{u}_1$ direction at m_1 and mass m_2. This angle, ?, defines the libration angle. Libration angles can have a component in the plane of the orbital plane, and a component out of the plane of the orbital plane. Figure 1 only shows the in-plane portion of the angle. For example, if ? equals zero degrees then the TSS is perfectly nadir-oriented (aligned vertically in the

$\hat{u}_1$ direction) and $\vec{r}_2$ simply becomes $|\vec{r}_1|$ plus the magnitude of the length of the tether, $|\vec{P}|$, in the $\hat{u}_1$ direction. Finally, g defines the angle between $\vec{r}_1$ and $\vec{r}_2$, which is a function of libration angle, tether length, and altitude of the TSS.

The next step is to look at how TS slant range is affected by the libration angle and the TS elevation angle. Another figure helps show how important the tracking site's elevation angle and the TSS libration angle are in determining TS slant range to a TSS. Figure 2 is similar to Figure 1 but it includes a tracking site located on the surface of the Earth.

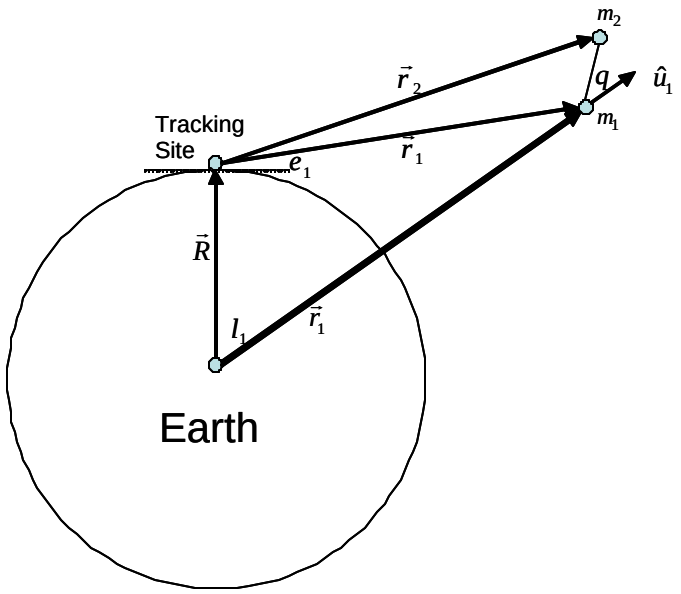

Figure 2- Slant Range from tracking Site to End Masses

$\vec{R}$ defines the vector from the center of the Earth to the tracking site. Each of the $\vec{r}$ terms describes the slant range vector from the site to one of the end masses of the TSS. The TS slant ranges to the end masses, r_1 and r_2, are the magnitudes of the TS slant range vectors. The dashed line at the site defines the local horizon for the TS. The e_1 term defines the TS elevation angle to the lower end mass of the TSS. A second

elevation angle, e_2, represents the TS elevation angle to the upper end mass of the TSS. Finally, l_1 represents the central earth angle between $\vec{R}$ and $\vec{r}_1$. There is also an angle, l_2, which defines the central earth angle between $\vec{R}$ and $\vec{r}_2$ (not shown for clarity).

Using Figures 1 and 2 together, the TS slant range to mass m_1, r_1, can be obtained if $\vec{R}$, $\vec{r}_1$, and e_1 are known:

$$|\vec{r}_1| = \sqrt{|\vec{r}_1|^2 - |\vec{R}|^2 + 2 * |\vec{r}_1| * |\vec{R}| * \cos(90° + e_1)} \tag{1}$$

Then by using the libration angle, q, and the length of the tether, $|\vec{P}|$, the TS slant range to mass m_2, r_2, can also be obtained. This is accomplished by first determining the central earth angle between the site and the lower mass:

$$l_1 = \cos^{-1}\left(\frac{|\vec{R}|^2 + |\vec{r}_1|^2 - |\vec{r}_1|^2}{2 * |\vec{R}| * |\vec{r}_1|} \right) \tag{2}$$

Next, determine $|\vec{r}_2|$ by using the law of cosines:

$$|\vec{r}_2| = \sqrt{|\vec{r}_1|^2 + |\vec{P}|^2 + 2 * |\vec{r}_1| * |\vec{P}| * \cos(q)} \tag{3}$$

The angle between $|\vec{r}_1|$ and $|\vec{r}_2|$ is then calculated by:

$$g = \cos^{-1}\left(\frac{|\vec{r}_1|^2 + |\vec{r}_2|^2 - |\vec{P}|^2}{2 * |\vec{r}_1| * |\vec{r}_2|} \right) \tag{4}$$

The Earth central angle to m_2, l_2, is obtained simply by subtracting g from l_1. Finally, the slant range from the tracking site to the upper end mass is determined with:

$$|\vec{r}_2| = \sqrt{|\vec{R}|^2 + |\vec{r}_2|^2 - 2 * |\vec{R}| * |\vec{r}_2| * \cos(l_2)} \quad (5)$$

This sequence of equations allows us to compare the slant range from a tracking site to each of the end masses. Using this tool, it can be shown that at lower elevations, smaller librations can cause the upper mass to have a smaller slant range than the lower mass. For example, a 4 kilometer tethered satellite with 40 degrees of libration viewed at approximately 27 degrees elevation or lower, will cause the upper end mass to have a shorter slant range than the lower end mass. In stark contrast to this, it is obvious that at 90 degrees elevation the libration angle must be 90 degrees before the slant range for the two end masses is equal. This analysis shows that the lower the elevation angle that a TSS is tracked, the less libration is needed to confuse the two objects apart solely on slant range alone.

Comparing the effects of libration and elevation angle on TS slant range and EC range requires a look at how libration and elevation angle affects EC range for each of the end masses. It is apparent that the TS elevation angle does not affect the EC range because the EC range is only related to the location of the TSS and the center of the Earth and has nothing to do with the location of the tracking site. Therefore, 90 degrees of libration will always be required before the upper mass and lower mass have equal EC ranges.

This analysis led our research to focus on EC range instead of TS slant range as a key parameter which could be used to tell TSS data points apart. However, even though EC range seems to be a better indicator for TSS data sorting based on the previous, there

are some issues related to real-world tracking site errors that have an affect on data sorting, which leads us to our next preliminary analysis.

Tracking Site Error Analysis

In the perfect world, our previous analysis indicates EC range is a better parameter to use for data sorting than TS slant range. Unfortunately, in the real world problems like tracking site errors need to be considered. These real-world problems led us to analyze tracking site error impacts on TS slant range and EC range to help determine the potential impact on sorting out TSS data points.

Analyzing tracking site errors on TS slant range is straightforward. Since radar sites obtain slant range directly as a measurement, the slant range error is incorporated directly. For example, if a tracking site has a potential slant range error of 0.02 km, and it obtains a reading on an object of 2000 km, then there is 68% confidence that the actual slant range of the object is between 1999.98 km and 2000.02 km. The errors in the elevation and azimuth of the tracking site do not affect the slant range measurement in any way.

However, analyzing the tracking site errors on the EC range, $|\vec{r}|$, requires more complicated computations. Since EC range error is not measured directly by a tracking site, it must be calculated from the tracking site errors for elevation and slant range. The tracking site azimuth error does not affect the EC range error because EC range is

calculated using the law of cosines by rearranging Equation (1) to solve for $|\vec{r}|$ and azimuth does not appear in this equation. From now on, instead of writing out the full form of the magnitude of any particular vector, it will be assumed that r equals $|\vec{r}|$, and so forth for any other magnitude calculation. To calculate the uncertainty of the EC range, s_r, given the uncertainties in slant range, s_r, and elevation, s_e, we use:

$$s_r^2 = J * \begin{Bmatrix} s_r^2 & 0 \\ 0 & s_e^2 \end{Bmatrix} * J^T \tag{6}$$

Where

$$J = \begin{bmatrix} \frac{\partial r}{\partial r} & \frac{\partial r}{\partial e} \end{bmatrix} \tag{7}$$

The elements $\frac{\partial r}{\partial r}$, and $\frac{\partial r}{\partial e}$ are obtained by taking the partial derivatives of Equation (1) with respect to r, and e, respectively. These partials are:

$$\frac{\partial r}{\partial r} = \frac{r - R * \cos(90° + e)}{\sqrt{R^2 + r^2 - 2 * R * r * \cos(90° + e)}} \tag{8}$$

$$\frac{\partial r}{\partial e} = \frac{-R * r * \sin(90° + e)}{\sqrt{R^2 + r^2 - 2 * R * r * \cos(90° + e)}} \tag{9}$$

These equations provide the error covariance in the EC range as a function of the TS elevation angle and slant range error covariances. For example, assume a tracking site has the following errors for slant range and elevation: $s_r = 0.021km$ and $s_e = 0.023°$. Next, assume a 2 km nadir-oriented tethered satellite has EC ranges for the bottom and top masses respectively of: $r_1 = 7399.5km$ and $r_2 = 7401.5km$. This leads to the covariances in Figure 3. The dotted line indicates the *actual* upper mass EC range,

and the curved dash-dot lines represent the 1 s_r error bars for the upper mass EC range. The lower solid line indicates the *actual* lower mass EC range, and the curved dashed lines represent the 1 s_r error bars for the lower mass EC range.

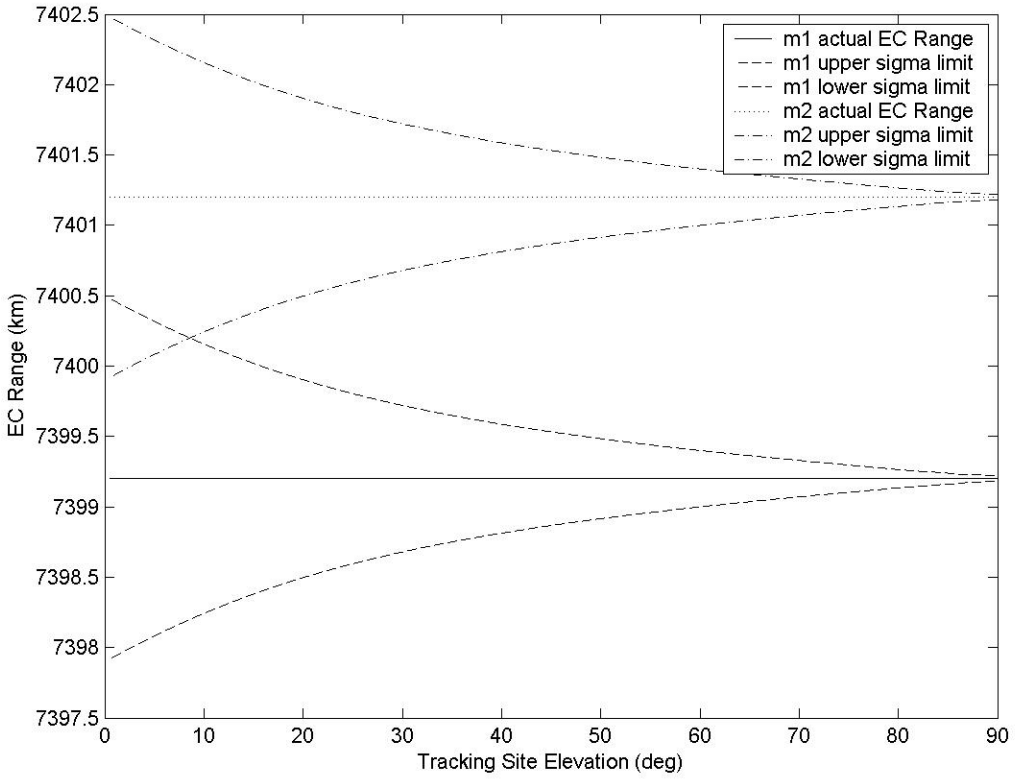

Figure 3- Affect of Elevation Angle on s_r

This figure shows how the potential EC range error grows larger as the elevation angle becomes smaller. Figure 4 shows how increasing the TS slant range error to $s_r = 0.5km$ affects the EC range error.

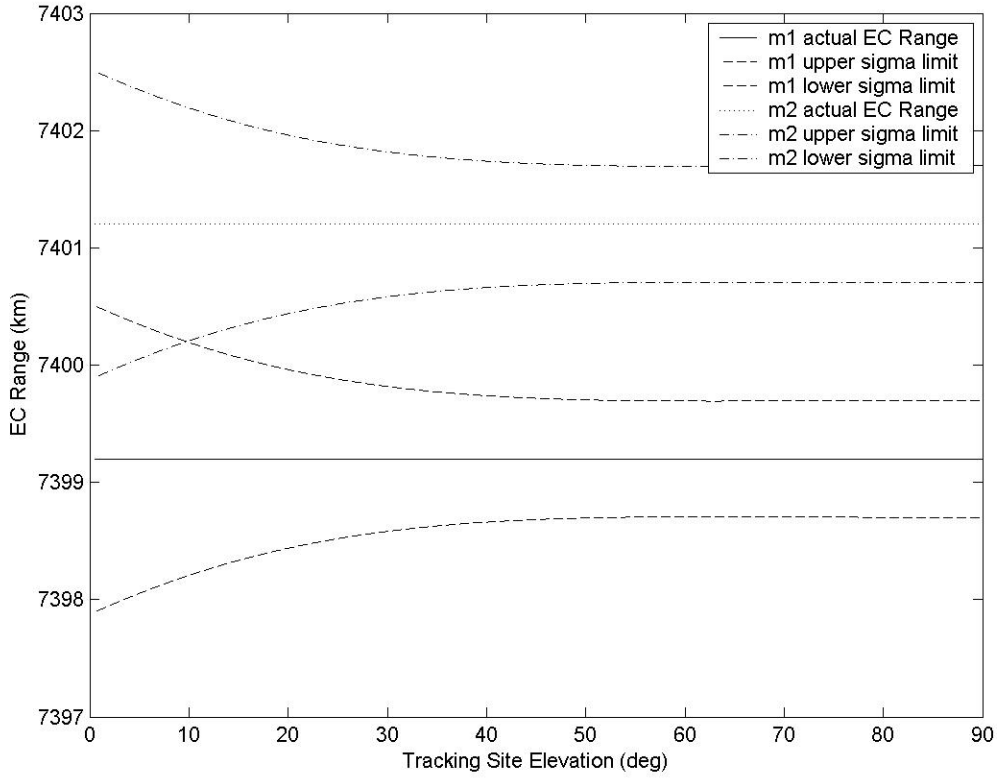

Figure 4- Affect of Larger Slant Range Error on s_r

These figures can be used to draw two conclusions. First, elevation angle and elevation angle error play a major role in determining the *actual* EC range of an object in space. The TS slant range error is not as big of a factor as elevation error. However, it is apparent that if a tracking site does have a larger slant range error then the error width even at 90 degrees is proportionately larger to incorporate the larger slant range error.

The second important point to note concerning both figures is the point at which the lines cross determines where the end masses may be confused for each other and an unsuccessful data sort may occur. This means that for these conditions and the previously mentioned tracking site errors, at approximately 10 degrees in elevation there

is an 84% chance that a data point will be properly identified if the actual position is known in advance (Bayer, 1991:497). Figure 4 shows how the slant range error has only a minor affect on the elevation at which data points may be confused. This shows that the elevation angle and elevation angle error are the main factors in determining a data point's *actual* EC range.

The plots shown in both figures apply only for the specified TSS, TSS orbit, and tracking site parameters. This important fact must be taken into consideration when analyzing the point at which a tracking site will probably start to have problems sorting out TSS tracking data.

This second analysis shows once again how elevation angle plays a role in being able to identify data points properly. The EC range is more susceptible to error than the slant range, particularly at low elevation angles.

Data Sorting Preliminary Analysis Conclusion

After looking at slant range and EC range as potential parameters for sorting out data points, it is evident that both exhibit different strengths and weaknesses. However, an additional advantage of the EC range not mentioned previously is that the values of EC range over time should be fairly constant for low eccentricity orbits. However, when looking at slant range, the values vary greatly over a short period of time, and if multiple sites are incorporated they will each have very different slant range values. In fact, slant range changes so quickly as a TSS passes near a sight, the value can change by several

thousand kilometers in minutes. Therefore, a plot of TS slant ranges for the upper and lower mass of a TSS can be very difficult to tell apart, by hand or by computer, because the slant range will have to cover such a large range of values. This final piece of information helped us decide to use the EC range parameter to sort the observational data.

IV. Estimating the TSS Orbit and Sorting the Data

Once a parameter was selected to use in the data sorting algorithm, the next step was to develop a method of estimating the orbit of the TSS CM based off of the tracking data. The estimation method used for this research is a four-fold process. The first process involves taking the *observed* azimuth, elevation, and slant range data from a tracking site and determining an *initial estimated* set of COEs. The second process uses a simulator to propagate the *estimated* COEs of the CM over the same time frame as all of the tracking site observations. These *estimated* COEs are then converted into *estimated* EC pqw coordinate frame position vectors. The third process involves taking the *observation* data again and converting that data into *observed* EC pqw position vectors. The fourth process involves iterating on steps 2 and 3 in order to minimize the difference between the *observed* and *estimated* pqw position vectors. This entire process allows us to take a poor initial guess of the CM COEs and optimize the initial guess until a much better estimate of the CM COEs is found.

The pqw coordinate system referred to above is also known as the perifocal coordinate system. The perifocal coordinate system is a coordinate system which is based on the orbit of the satellite. The origin of the pqw coordinate system is the center of the Earth. The $\hat{p}$ and $\hat{q}$ directions lie in the plane of the orbit, with the $\hat{p}$ direction aligned with the perigee point, while the $\hat{q}$ direction is perpendicular to $\hat{p}$. The $\hat{w}$ direction is perpendicular to the plane of the satellite orbit, since it must be perpendicular

to both $\hat{p}$ and $\hat{q}$. Thus for a single satellite there should not ever be any value in the $\hat{w}$ direction since the satellite should always lie in the plane of its orbit. For the tethered satellite problem, we will also be making an assumption that the TSS maintains a nadir orientation, so each of the end masses and the CM of a TSS also should have no $\hat{w}$ value.

One might wonder why position vectors are calculated in the pqw coordinate frame instead of the IJK frame. One benefit of the IJK frame is that we can calculate the *actual* values of the *observations* in IJK coordinates. However, pqw coordinates are based off of the orbit, and in our case we only have a poor initial COE *estimate* and we do not know the *actual* orbit. The reason for using pqw coordinates in the estimation process is simply that the algorithm is more robust. The estimator was implemented with IJK components but the algorithm did not converge unless the first guess COEs were essentially the same as the actual COEs. One potential reason why pqw coordinates work better centers on the $\hat{w}$ component of the pqw coordinate frame. The $\hat{w}$ component for any satellite must be near-zero. Even for a TSS, only the libration that occurs out of the plane of the orbit will affect the $\hat{w}$ values for the end masses. So, this essentially decouples this component from the in-plane variables.

Assumptions

There are four major assumptions used in the estimation process which simplify the problem. These four assumptions include nadir orientation, rigid tether, CM and

center of gravity equal, and J2 perturbations being the only significant perturbations which affect the orbital motion of the CM.

As talked about in the introduction, nadir orientation is a commonly accepted starting point to analyze tethered satellites. This is particularly important in our application since we are dealing with different TSS's, and where we have little or no knowledge about the key parameters of the TSS. Trying to handle libration in addition to estimating an orbit and data sorting, while only using a limited amount and type of tracking data, would cause the complexity of this issue to increase greatly. In the future, as more TSS's are flown and there is more understanding of the real-world dynamics of tethers, handling libration might become more reasonable. In addition, nadir orientation is a reasonable assumption if the TSS maintains a fairly low libration angle. Looking at a real-world case, the libration values for TiPS went from approximately 30-40 degrees down to approximately 5-7 degrees in less than a year (Purdy et al., 1997:2).

The second assumption in determining a CM orbit and data sorting is the tether will remain fully rigid. This means the tether is treated as an inflexible bar. This assumption eliminates the need to attempt to calculate the tension on the tether at any particular time. Assuming a rigid tether is one important way to help simplify the differential equations and is often used in studying tether motion (Beletsky and Levin, 1993:48).

The third important assumption for this research is the CM of a TSS is always equal to the center of gravity of the system, and, therefore, the CM follows a near-Keplerian orbit. In reality, tether motion does affect orbital motion because it causes the center of gravity to constantly change. This changing force on orbital motion caused by

tether motion is related by the term $\left(\frac{P}{r}\right)^2$ (Cochran et al., 1999:1826). Since the tether length, P, is much, much smaller than r and this term is squared, the effects of tether motion are very small. This is a well-accepted assumption used by many tether analysis techniques that deal with trying to determine the orbital motion of the CM of the TSS. In addition, since we may not know much about the TSS, trying to calculate the perturbations on orbital motion caused by the changes in the attitude becomes almost impossible. So, since these perturbations on orbital motion due to attitude motion are small and they are almost impossible to calculate with the information given, we will assume the CM and the center of gravity are always equal.

Finally, the last assumption used in this research deals with perturbations to two-body motion. The J_2 perturbations caused by the oblateness of the Earth are generally one of the most common perturbations to include in the study of orbital motion. While there are many other perturbations which affect all satellites such as solar pressure, J_2 will generally dominate for Low-Earth Orbit (LEO) satellites.

The one additional perturbation which might cause significant problems for TSS analysis is air drag. Since a TSS most likely has a much larger surface area to mass ratio than normal satellites, very low flying tethers could have significant perturbations due to air drag. However, since we are assuming to know very little about the TSS it also becomes hard to determine an accurate drag estimate. In addition, as long as a TSS is flying high enough where drag becomes much less of a concern, then this perturbation becomes small enough to ignore. Therefore, J_2 perturbations are the only perturbations which are considered in this research.

Obtaining an Initial Estimate of the COEs of the CM

Having scoped the problem with all of the assumptions and having developed an idea for what parameter works best for data sorting, the next step is to determine a first-guess for the Classical Orbital Elements (COEs) of the CM at a specific epoch time. This problem can be solved using any recognized orbit determination technique. This paper uses the Herrick-Gibbs technique, but other methods such as Gauss or f and g may also be used.

The Herrick-Gibbs formula used here was obtained from Pedro Escobal's *Methods of Orbit Determination*, 1965. The Herrick-Gibbs technique requires 3 EC IJK position vectors to come up with an initial estimate for the COEs. However, before showing the Herrick-Gibbs formula it is important to show how IJK position vectors are obtained in the first place from the tracking site data. The technique for obtaining IJK position vectors from radar data was obtained from Bate, Mueller, and White's *Fundamentals of Astrodynamics*.

The process of taking *observations* from radar data and converting it into accurate inertial IJK position vector data entails several coordinate transformations, an understanding of the eccentricity of the Earth, and being able to calculate local sidereal times (Bate et al., 1971:83). The first step of converting radar data requires transforming the data into the Topocentric-Horizon Coordinate System. This coordinate system has the radar's location on the surface of the Earth as its origin. The axes are defined as South, East, and Up or more commonly S, E, Z.

The relationship which relates *observed* slant range (r_o), elevation (e_o), and azimuth (a_o) to an *observed* position vector in the SEZ frame, ${}^{SEZ}\vec{r}_o$, can be arrived at through simple geometry:

$$\begin{pmatrix} {}^{S}r_o \\ {}^{E}r_o \\ {}^{Z}r_o \end{pmatrix} = \begin{pmatrix} -r_o * \cos(e_o) * \cos(a_o) \\ r_o * \cos(e_o) * \sin(a_o) \\ r_o * \sin(e_o) \end{pmatrix} \tag{10}$$

where ${}^{S}r$, ${}^{E}r$, ${}^{Z}r$, are the S, E, and Z components of the SEZ slant range vector, ${}^{SEZ}\vec{r}_o$, respectively. This SEZ frame is not an inertial frame since the origin is a tracking site located on the surface of a rotating Earth. So, to get to an inertial frame with the center of the Earth as its origin several more items must be discussed.

In order to potentially use tracking data from multiple sites it is important to be as accurate as possible when discussing where a tracking site is located. The easiest way to calculate the position of a tracking site is simply to use the site's latitude and longitude and convert those directly using a spherical model of the Earth. However, since the Earth is not truly spherical, using this method can potentially lead to errors in the TS position on the order of kilometers. Since data sorting relies on having the most accurate EC range possible, this potential error needs to be eliminated if possible. One of the most common ways to handle the true shape of the Earth is to treat it as an ellipsoid instead of a sphere (Bate et al., 1971:93). Bate, Mueller, and White go through an excellent discussion of calculating the position vector of a site on the Earth based on its latitude, longitude, and altitude above mean sea level. We will summarize this set of calculations here since it is so vital for data sorting.

When doing calculations using an ellipsoid model of the Earth there are a couple of constants to keep in mind. These constants include Equatorial radius ($a_e \approx 6378.165km$), and Earth Eccentricity ($e_e \approx 0.08181$) (Bate et al., 1971:94). The other quantities needed to calculate the location of a site on an ellipsoid Earth include the site's geodetic latitude (f), geographic longitude (b), height above mean sea level (*H*), and the time of the observation.

The first step in this process is to calculate Greenwich sidereal time, t_g, at the time of the observation. This is accomplished by looking up in a table the Greenwich sidereal time for a particular date, which we will term t_{go}. For example, Bate, Mueller, and White give a value of $99.990704°$ for t_{go} for 1 Jan 1971 at 0 hours Universal Time (Bate et al., 1971:104). If it is known how many days (*D*) have passed since that time (to include fractions of a day), then t_g is calculated by:

$$t_g = t_{go} + 1.0027379093 * 360° * D \tag{11}$$

Once t_g is known for the specified time, the next step is to calculate the local sidereal time for the site, (t_{site}). There is a simple relationship between Greenwich sidereal time and the site's geographic longitude:

$$t_{site} = t_g + b \tag{12}$$

Now that local sidereal time is known the next step is to calculate the inertial IJK position vector of the site at that time. This is accomplished by initially calculating the following two quantities:

$$x = \left| \frac{a_e}{\sqrt{1 - e_e^2 * \sin^2(f)}} + H \right| * \cos(f) \tag{13}$$

$$z = \left| \frac{a_e * (1 - e^2)}{\sqrt{1 - e_e^2 * \sin^2(f)}} + H \right| * \sin(f) \tag{14}$$

Next, the inertial IJK position vector of the tracking site, ${}^{IJK}\vec{R}$, is obtained from:

$${}^{IJK}\vec{R} = x * \cos(t_{site})\hat{I} + x * \sin(t_{site})\hat{J} + z\hat{K} \tag{15}$$

Knowing the inertial IJK position of the site is only the first half of determining the IJK position vector of the *observed* object. Now, the *observed* SEZ slant range vector, ${}^{SEZ}\vec{r}_o$, found in Equation (10) needs to be converted into an *observed* inertial IJK slant range vector, ${}^{IJK}\vec{r}_o$. This is accomplished by using a transformation matrix:

$$\begin{bmatrix} {}^{I}r_o \\ {}^{J}r_o \\ {}^{K}r_o \end{bmatrix} = \begin{bmatrix} \sin(f) * \cos(t_{site}) & -\sin(t_{site}) & \cos(f) * \cos(t_{site}) \\ \sin(f) * \sin(t_{site}) & \cos(t_{site}) & \cos(f) * \sin(t_{site}) \\ -\cos(f) & 0 & \sin(f) \end{bmatrix} * \begin{bmatrix} {}^{S}r_o \\ {}^{E}r_o \\ {}^{Z}r_o \end{bmatrix} \tag{16}$$

where ${}^{I}r_o$, ${}^{J}r_o$, and ${}^{K}r_o$, are simply the I, J, and K components of ${}^{IJK}\vec{r}_o$, respectively. Finally, the *observed* inertial IJK position vector for the tracked object, ${}^{IJK}\vec{r}_o$, is obtained by:

$${}^{IJK}\vec{r}_o = {}^{IJK}\vec{R} + {}^{IJK}\vec{r}_o \tag{17}$$

Once the tracking data is finally converted into *observed* IJK position vectors, the Herrick-Gibbs formula for calculating the *observed* velocity vector can be used (Escobal, 1965:305). The Herrick-Gibbs method essentially takes three position vectors to calculate a velocity vector for the second position vector. Using this method, the second position vector time is set to zero, so the time for position vector one will be negative.

This technique is started by calculating what Escobal terms "modified times" (Escobal, 1965). If each of the three *observed* position vectors, ${}^{IJK}\vec{r}_{oi}$, (*i*=1,2,3) has an associated time (t_j), where $j = 1,2,3$, then the modified times (y) are calculated by:

$$y_{ij} = k * \left(t_j - t_i\right) \tag{18}$$

where $k = m_e = 3.986 * 10^5 km^3 / s^2$, which is the Earth gravitational parameter, and $i = 1,2,3$. Next, the following quantities are calculated:

$$\begin{aligned} T_{13} &= t_3 - t_1 \\ \overline{G}_1 &\equiv \frac{y_{23}}{y_{12} * T_{13}} \\ \overline{G}_3 &\equiv \frac{y_{12}}{y_{23} * T_{13}} \\ \overline{G}_2 &\equiv \overline{G}_1 - \overline{G}_3 \end{aligned} \tag{19}$$

In addition to the previous quantities, the following quantities must also be calculated:

$$\begin{aligned} \overline{H}_1 &\equiv \frac{y_{23}}{12} \\ \overline{H}_3 &\equiv \frac{y_{12}}{12} \\ \overline{H}_2 &\equiv \overline{H}_1 - \overline{H}_3 \end{aligned} \tag{20}$$

Using both the $\overline{G}$ and $\overline{H}$ quantities, the coefficients of the velocity vector are calculated as follows:

$$d_i = \overline{G}_i + \frac{\overline{H}_i}{r_{oi}^3} \tag{21}$$

Which then leads to the *observed* velocity vector:

$${}^{IJK}\vec{v}_{o2} = {}^{IJK}\dot{\vec{r}}_{o2} = -d_1 * {}^{IJK}\vec{r}_{o1} + d_2 * {}^{IJK}\vec{r}_{o2} + d_3 * {}^{IJK}\vec{r}_{o3} \tag{22}$$

It is important to remember where all of these calculations are leading, and that is to a first-guess of the COEs at an epoch time of the TSS CM. This estimate is not going to be nearly accurate enough because of the fact that all of the methods for calculating velocity vectors from position vectors and then obtaining COEs from position and velocity vectors are designed for single-body satellites. If a TSS is very short, or if all of the readings are taken from the same end mass and that end mass contains most of the mass of the system, then this estimate may be close, but in most cases for TSS's this will only give a rough estimate. But, we do need some starting point to get the estimation process started even if it is very rough. So, the next step in obtaining an estimate of the COEs of the CM is to convert the position and velocity vectors into COEs. We once again refer back to Bate, Mueller, and White for the method of calculating COEs from a position and velocity vector.

Position and velocity is all that is needed to uniquely determine the orbit of a satellite. The method for accomplishing this first starts by calculating the angular momentum vector, $\vec{h}$ by the following equation:

$$\vec{h} = {}^{IJK}\vec{r}_o \times {}^{IJK}\vec{v}_o \tag{23}$$

Next, the node vector is calculated by crossing the angular momentum vector with the $\hat{K}$ direction:

$$\vec{n} \equiv \hat{K} \times \vec{h} \tag{24}$$

The eccentricity vector is the next important item which will help us determine the COEs, and this vector is calculated as follows:

$$\vec{e} = \frac{1}{m} * \left[\left(v_o^2 - \frac{m_e}{r_o} \right) * \vec{r}_o - \left(\vec{r}_o \cdot \vec{v}_o \right) * \vec{v}_o \right] \tag{25}$$

These three vectors enable us to obtain the *initial estimate* of the TSS CM COEs for the epoch time using the following set of equations:

$$\begin{aligned}
a_0 &= \frac{h^2}{m_e * \left(1 - e^2\right)} \\
e_0 &= \left|\vec{e}\right| \\
i_0 &= \cos^{-1}\left(\frac{\vec{h} \cdot \hat{K}}{h} \right) \\
w_0 &= \cos^{-1}\left(\frac{\vec{n} \cdot \vec{e}}{n * e} \right) \\
\Omega_0 &= \cos^{-1}\left(\frac{\vec{n} \cdot \hat{I}}{n} \right) \\
n_0 &= \cos^{-1}\left(\frac{\vec{e} \cdot \vec{r}_o}{e * r_o} \right)
\end{aligned} \tag{26}$$

For each of the angles calculated previously the following quadrant checks apply:

$$\begin{gathered}
i \text{ is always less than } 180° \\
\text{If } \vec{n} \cdot \hat{J} > 0 \text{ then } \Omega < 180° \\
\text{If } \vec{e} \cdot \hat{K} > 0 \text{ then } w < 180° \\
\text{If } \vec{r} \cdot \vec{v} > 0 \text{ then } n_0 < 180°
\end{gathered} \tag{27}$$

With all of these calculations we finally have an estimate for the CM COEs at our epoch time. This initial guess is then used in the rest of the estimation process as a starting point.

Calculating Estimated EC pqw Position Vectors

Propagating Estimated CM COEs for all Observation Times.

Once the *first-guess* COEs are finally calculated for the epoch time, the *estimated* COEs for the CM at *all* times at which our observations occur need to be calculated. Since we have chosen to ignore all perturbations except J_2 the following set of equations apply for determining the *estimated* COEs of the CM at some time, *t*, when the *initial estimated* COEs are given for our epoch time, t_0:

$$\begin{aligned}
a(t) &= a_0 \\
i(t) &= i_0 \\
e(t) &= e_0 \\
w(t) &= w_0 + \left(\frac{3}{2} * \frac{J_2}{p^2} * \left[2 - \frac{5}{2} * \sin^2(i)\right]\right) * \bar{n} * (t - t_0) \\
\Omega(t) &= \Omega_0 - \left(\frac{3}{2} * \frac{J_2}{p^2} * \cos(i)\right) * \bar{n} * (t - t_0) \\
M(t) &= M_0 + \bar{n} * (t - t_0)
\end{aligned} \tag{28}$$

where J_2 has a dimensionless value of 0.00108263 (Wertz and Larson, 1999:143). The anomalistic mean motion, $\bar{n}$, (Escobal, 1965:369) is given by:

$$\bar{n} = n_0 * \left[1 + \frac{3}{2} * J_2 * \frac{\sqrt{1 - e^2}}{p^2} * \left(1 - \frac{3}{2} * \sin^2(i)\right)\right] \tag{29}$$

where n_0 is the unperturbed mean motion and is equal to $\sqrt{m_e / a^3}$. The parameter p is called the semi-parameter of the orbit and is equal to $a * (1 - e^2)$. The terms $M(t)$ and M_0 are the mean anomaly of the object's position at time *t* and t_0, respectively. The variable we are interested in determining at time *t* is the true anomaly, n, not the mean

anomaly. Calculating true anomaly for an elliptical orbit requires a series of calculations. The first step for calculating true anomaly at time t is to calculate the eccentric anomaly at time t_0, E_0, and then calculate M_0. This is accomplished with the following set of equations:

$$E_0 = 2 * \tan^{-1}\left(\sqrt{\frac{1-e_0}{1+e_0}} * \tan\left(\frac{n_0}{2}\right)\right) \tag{30}$$

$$M_0 = E_0 - e_0 * \sin(E_0) \tag{31}$$

Calculating M at time t is calculated as follows:

$$M(t) = M_0 + \overline{n} * (t - t_0) \tag{32}$$

There is no direct relationship for calculating true anomaly from mean anomaly, so an iterative procedure must be employed. First, eccentric anomaly at time t is calculated from:

$$E(t) = M(t) + e(t) * \sin(E(t)) \tag{33}$$

This equation can not be solved directly for eccentric anomaly, so an iteration technique must be applied. One popular method of solving this problem is to use a Newton-Rhapson Iteration Method. We solve this problem by first guessing the value of $E(t)$. A good first guess for orbits which do not have a high eccentricity value is to set $E(t)$ equal to $M(t)$. Then, substitute this guess for $E(t)$ into equation (31). If the new value for mean anomaly, $M_2(t)$ is equal to $M(t)$ within a certain tolerance level, say $1 * 10^{-12}$, then this guess for $E(t)$ is the final answer for $E(t)$. However, if it is not less than the designated tolerance level, then a better estimate for $E(t)$ is obtained from:

$$E_2(t) = E(t) + \frac{M(t) - M_2(t)}{1 - e(t) * \cos(E(t))} \tag{34}$$

Once this new estimate for eccentric anomaly, $E_2(t)$ is calculated, then this new estimate becomes the new value to substitute into equation (31). This process is repeated until the desired tolerance level is achieved. This method of determining the eccentric anomaly converges quickly for small eccentricity. Now that eccentric anomaly is calculated at time *t* we can finally calculate the true anomaly at *t*, $n(t)$. The equation used to calculate this value is the same as equation (30), but now it is reversed to solve for true anomaly:

$$n(t) = 2 * \tan^{-1}\left(\sqrt{\frac{1+e(t)}{1-e(t)}} * \tan\left(\frac{E(t)}{2}\right)\right) \tag{35}$$

The equations and techniques described in this section are all that is needed to obtain *estimated* COEs for the CM of a TSS at any time as long as an *initial estimate* at some epoch time, t_0 is obtained first. Unfortunately, knowing the COEs of the CM at each time does not allow us to calculate the quantities needed for the estimation process. So, there are several additional steps which need to be accomplished First, we need to convert the newly found *estimated* CM COEs at each particular time back into *estimated* IJK position vectors. Second, we need to determine the *estimated* IJK position vectors of both of the end masses. Finally, the *estimated* IJK position vectors of the end masses need to be converted into *estimated* pqw position vectors. All of these processes are covered in the next two sections.

Converting Estimated CM COEs into Estimated CM IJK Position Vectors.

Converting the *estimated* COEs back into *estimated* IJK position vectors is a two-step process. The first step is to calculate the *estimated* position vector in the pqw, or perifocal, coordinate system. The second step is to accomplish a coordinate transformation on the *estimated* position vector in the pqw coordinate system into the IJK coordinate system. Bate, Mueller, and White have an excellent discussion of this task (Bate et al., 1971:72). We will briefly summarize what they cover next.

There are two equations which are needed to calculate the *estimated* EC position vector in the pqw frame. The first equation is used to calculate the magnitude of the *estimated* position vector in the pqw frame and is related to the *estimated* COEs by:

$$r_e(t) = \frac{a(t) * \left(1 - e(t)^2\right)}{1 + e(t) * \cos(n(t))} \tag{36}$$

Then calculate the *estimated* position vector in the pqw frame:

$${}^{pqw}\vec{r}_e(t) = r_e(t) * \cos(n(t))\hat{p} + r_e(t) * \sin(n(t))\hat{q} \tag{37}$$

Now that the *estimated* position vector has been defined in the pqw coordinate system all that is needed to calculate the *estimated* position vector in the IJK frame is a coordinate transformation. The transformation matrix required to go from the pqw frame to the IJK frame is shown in the next equation and is equal to $\underset{pqw}{\overset{IJK}{R}}$. In regards to this transformation matrix, $c(\ )$ represents cosine and $s(\ )$ represents sine:

$$\underset{pqw}{\overset{IJK}{R}} = \begin{bmatrix} c(\Omega)*c(w) - s(\Omega)*c(i)*s(w) & -c(\Omega)*s(w) - s(\Omega)*c(i)*c(w) & s(\Omega)*s(i) \\ s(\Omega)*c(w) + c(\Omega)*c(i)*s(w) & -s(\Omega)*s(w) + c(\Omega)*c(i)*c(w) & -c(\Omega)*s(i) \\ s(i)*s(w) & s(i)*c(w) & c(i) \end{bmatrix} \tag{38}$$

Using this transformation matrix the *estimated* IJK position vector of the CM, ${}^{IJK}\vec{r}_e$, is obtained by the next equation.

$$\begin{bmatrix} {}^{I}r_e \\ {}^{J}r_e \\ {}^{K}r_e \end{bmatrix} = \underset{pqw}{\overset{IJK}{R}} * \begin{bmatrix} {}^{p}r_e \\ {}^{q}r_e \\ {}^{w}r_e \end{bmatrix} \tag{39}$$

This completes the method for determining the *estimated* inertial position vector of the CM in the EC IJK frame at time *t*. For a TSS, however, the CM is most likely a point in space somewhere between the end masses, and a radar site will probably track one of the end masses and not the CM. The next section discusses how to determine the *estimated* pqw position vectors of the end masses when given the *estimated* IJK position vector of the CM.

Calculating Estimated TSS End Mass pqw Position Vectors.

Determining the *estimated* IJK position vectors of the end masses of a TSS requires knowledge of the length of the tether, the mass of the end objects, and the mass of the tether. If it is known that the system is a TSS and the tether parameters are known, such as TiPS, then those values can be used directly. If, however, the length is unknown, or if it is an unknown TSS, then these parameters will be estimated along with the TSS orbit.

In order to determine the *estimated* end mass IJK position vectors from the *estimated* CM IJK position vector we refer back to the $\hat{u}_1$ coordinate component. Due to the nadir-oriented assumption the *estimated* position vectors for both end masses will only have values in the $\hat{u}_1$ direction. The equation for determining the distance m_1 is from the CM, d_{m1}^{cm}, is shown by:

$$d_{m1}^{cm} = \frac{m_2 * P + m_t * \frac{P}{2}}{m_1 + m_2 + m_t} \tag{40}$$

The value, m_t, is the mass of the tether. We will assume the value is zero in the case of an unknown TSS. By knowing the distance of m_1 from the CM, it is easy to calculate the distance of m_2 from the CM, d_{m2}^{cm}, by:

$$d_{m2}^{cm} = P - d_{m1}^{cm} \tag{41}$$

Knowing the *estimated* distances of the end masses from the CM, the *estimated* $\hat{u}_1$ position vector values for end mass 1, ${}^{u}\vec{r}_{e1}$, and mass 2, ${}^{u}\vec{r}_{e2}$, are related to the *estimated* CM magnitude, r_e, by the next set of equations:

$$\begin{aligned} {}^{u}\vec{r}_{e1} &= \left(r_e - d_{m1}^{cm}\right)\hat{u}_1 \\ {}^{u}\vec{r}_{e2} &= \left(r_e + d_{m2}^{cm}\right)\hat{u}_1 \end{aligned} \tag{42}$$

The *estimated* IJK position vectors for end mass 1, ${}^{IJK}\vec{r}_{e1}$, and mass 2, ${}^{IJK}\vec{r}_{e2}$ can be obtained through simple geometry. Figure 5 shows the relationship between the $\hat{u}_1$ direction and the IJK coordinate frame.

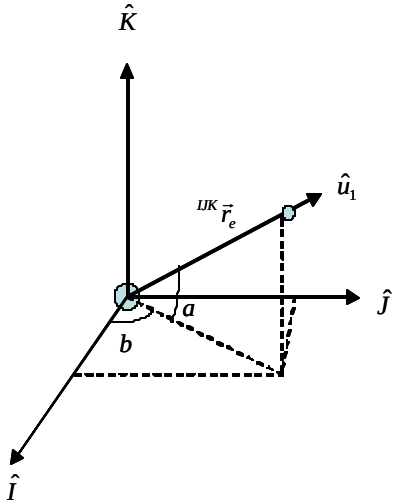

Figure 5- Relationship between $\hat{u}_1$ and IJK Coordinate Frame

The $\hat{u}_1$ direction is given by two angles and the *estimated* IJK position vector of the CM, ${}^{IJK}\vec{r}_e$. Remember, the origin for both of these systems is the center of the Earth. The first angle, b, is the angle measured from the $\hat{I}$ direction to the dashed line which is the projection of ${}^{IJK}\vec{r}_e$ in the I-J plane. The second angle, a, is the angle measured from the I-J plane to ${}^{IJK}\vec{r}_e$. Both angles are calculated by:

$$\begin{aligned} b &= \tan^{-1}\left(\frac{{}^{IJK}\vec{r}_e \cdot \hat{J}}{{}^{IJK}\vec{r}_e \cdot \hat{I}}\right) \\ a &= \tan^{-1}\left(\frac{{}^{IJK}\vec{r}_e \cdot \hat{K}}{r_e}\right) \end{aligned} \tag{43}$$

Once these two angles are known the *estimated* IJK position vectors for end mass 1, ${}^{IJK}\vec{r}_{e1}$, and end mass 2, ${}^{IJK}\vec{r}_{e2}$ are calculated by:

$$\begin{aligned} {}^{IJK}\vec{r}_{e1} &= \left|{}^{u}\vec{r}_{e1}\right| * \cos(a) * \cos(b)\hat{I} + \left|{}^{u}\vec{r}_{e1}\right| * \cos(a) * \sin(b)\hat{J} + \left|{}^{u}\vec{r}_{e1}\right| * \sin(a)\hat{K} \\ {}^{IJK}\vec{r}_{e2} &= \left|{}^{u}\vec{r}_{e2}\right| * \cos(a) * \cos(b)\hat{I} + \left|{}^{u}\vec{r}_{e2}\right| * \cos(a) * \sin(b)\hat{J} + \left|{}^{u}\vec{r}_{e2}\right| * \sin(a)\hat{K} \end{aligned} \tag{44}$$

This is the final step in determining the *estimated* IJK position vectors for each of the TSS end masses. Now, we can determine the *estimated* pqw position vectors for each of the TSS end masses. The transformation of all of the end mass positions obtained from Equation (44) uses the inverse of the conversion from pqw coordinates to IJK coordinates given in Equation (38) and is given by:

$$\begin{bmatrix} {}^{p}r_e \\ {}^{q}r_e \\ {}^{w}r_e \end{bmatrix}_i = \left(\underset{pqw}{R}^{IJK}\right)^T * \begin{bmatrix} {}^{I}r_e \\ {}^{J}r_e \\ {}^{K}r_e \end{bmatrix}_i \tag{45}$$

where *i* is equal to 1 or 2 depending on the end mass we are calculating. This is all that is needed to obtain *estimated* pqw position vectors for end mass 1, ${}^{pqw}\vec{r}_{e1}$, and end mass 2, ${}^{pqw}\vec{r}_{e2}$ when the *estimated* IJK position vectors are known. Having the *estimated* pqw position vectors now, the next step we need to do is convert all of the *observation* data into *observed* pqw position vectors, ${}^{pqw}\vec{r}_{o}$.

Converting Observation Data into pqw Position Vectors

At first glance calculating the *observed* pqw position vectors, ${}^{pqw}\vec{r}_{o}$, may not seem logical or possible. The *estimated* orbit is known since we determined this earlier. So, converting the *estimated* COEs into *estimated* end mass pqw position vectors is possible. But, we do not know the *actual* orbit which the satellite is following, so we cannot calculate the *observed* EC position vectors in the *actual* pqw frame. Instead, we calculate the *observed* EC position vectors in the *estimated* pqw frame.

One of the important items to remember about the pqw frame is that any tracked object in space should have a zero or near-zero component in the $\hat{w}$ direction. So, as position vectors are calculated in the *estimated* pqw frame, *all* of the *estimated* pqw position vectors will already be near-zero in the $\hat{w}$ direction, while the *observed* pqw position vectors may or may not have a near-zero component.

Up to this point, we have not discussed how to calculate *observed* position vectors in the pqw frame for all of the data points, but the general process of calculating pqw

position vectors has been shown. In Equations (10)-(17), we calculated IJK position vectors for several of the *observations* in order to obtain an initial estimate. So, the first step of calculating pqw position vectors for the *observations* is to take *all* of the observations and calculate *all* of their corresponding IJK position vectors using these equations. The next step of calculating pqw position vectors for the *observations* requires the use of the *estimated* set of COEs in Equation (45) to convert the *observed* IJK position vectors into *observed* pqw position vectors.

Now that we have *estimated* and *observed* pqw position vectors we can begin the process of improving the initial guess of the COEs so the difference between the *estimated* and *observed* pqw position vectors is minimized.

Minimizing Observed and Estimated Differences

Calculating TSS Position Residuals.

A common method for comparing an estimate of an orbit with what is observed is computing residuals. A residual is the difference between an *observed* and *estimated* quantity (Bate et al., 1971:123). For this estimation problem, we computed residuals for *both* EC range and pqw position vector components.

Initial estimation attempts with just using the EC range residuals showed that this approach was very effective in sorting the data and determining the orbit semi-major axis, eccentricity, and true anomaly. This information completely specified the distance to the center of the Earth for an end-body, so the other orbital elements are unobservable when

using only the EC range residuals. Consequently, we added the pqw position vectors to the residuals in order to estimate the orbit plane (longitude of the ascending node, inclination, and argument of perigee). As mentioned above, the pqw frame position vectors had better convergence properties than the IJK frame. A combination of these two quantities for the residuals gave the most robust and accurate performance for both the data sorting and the orbit estimation.

It is common practice to calculate a residual by subtracting *estimates* from *observations* so this is the convention we use here (Wiesel, 2003:25). Calculating the residuals for a TSS is different than calculating residuals for a single-body satellite. Since we have two end masses, and we do not know ahead of time which one is represented by the data, we have to calculate a set of residuals for each end mass.

Since there are two end mass *estimates*, but only one *observation* there will be two sets of residuals which are calculated. Calculating the EC range residuals for mass 1, X_{r1}, and mass 2, X_{r2}, is done by subtracting the *estimates*, $r_{e1}(t)$ and $r_{e2}(t)$, from the observation, $r_o(t)$, as shown by:

$$\begin{aligned} X_{r1}(t) &= r_o(t) - r_{e1}(t) \\ X_{r2}(t) &= r_o(t) - r_{e2}(t) \end{aligned} \tag{46}$$

Since there are three pqw components and a different pqw *estimate* for both end masses there are a total of six pqw component residuals. These six residuals will be represented as follows: m_1 p component residual, $\mathrm{X}_{\bar{r}1p}$, m_1 q component residual, $\mathrm{X}_{\bar{r}1q}$, m_1 w component residual, $\mathrm{X}_{\bar{r}1w}$, m_2 p component residual, $\mathrm{X}_{\bar{r}2p}$, m_2 q component residual, $\mathrm{X}_{\bar{r}2q}$, and m_2 w component residual, $\mathrm{X}_{\bar{r}2w}$. Using the *observed* pqw position

vectors, ${}^{pqw}\vec{r}_o(t)$, and the *estimated* pqw position vectors, ${}^{pqw}\vec{r}_{e1}(t)$ and ${}^{pqw}\vec{r}_{e2}(t)$, calculated previously from Equation (45) the pqw component residuals are shown as follows:

$$\begin{aligned}
X_{\vec{r}1p}(t) &= \left({}^{pqw}\vec{r}_o(t)\cdot\hat{p}\right)-\left({}^{pqw}\vec{r}_{e1}(t)\cdot\hat{p}\right)\\
X_{\vec{r}2p}(t) &= \left({}^{pqw}\vec{r}_o(t)\cdot\hat{p}\right)-\left({}^{pqw}\vec{r}_{e2}(t)\cdot\hat{p}\right)\\
X_{\vec{r}1q}(t) &= \left({}^{pqw}\vec{r}_o(t)\cdot\hat{q}\right)-\left({}^{pqw}\vec{r}_{e1}(t)\cdot\hat{q}\right)\\
X_{\vec{r}2q}(t) &= \left({}^{pqw}\vec{r}_o(t)\cdot\hat{q}\right)-\left({}^{pqw}\vec{r}_{e2}(t)\cdot\hat{q}\right)\\
X_{\vec{r}1w}(t) &= \left({}^{pqw}\vec{r}_o(t)\cdot\hat{w}\right)-\left({}^{pqw}\vec{r}_{e1}(t)\cdot\hat{w}\right)\\
X_{\vec{r}2w}(t) &= \left({}^{pqw}\vec{r}_o(t)\cdot\hat{w}\right)-\left({}^{pqw}\vec{r}_{e2}(t)\cdot\hat{w}\right)
\end{aligned} \tag{47}$$

For our purposes, since there is no libration in our estimated TSS, subtracting the *estimated* terms out of $X_{\vec{r}1w}$ and $X_{\vec{r}2w}$ is not really necessary because those values should be zero. But, we have included the terms here for completeness.

We now have all of the residuals necessary to do the optimization and data sorting. The next section describes how the optimization and data sorting was accomplished using MATLAB.

Optimizing the COEs and Data Sorting.

Knowing all of the residuals enables us to quantify how far the COE *estimate* is from the *actual* orbit. It also allows us to try and find a new COE *estimate* which is better than the previous one. The process we are using to determine a new estimate is optimization. It is beyond the scope of this paper to discuss how the optimization process works, but we will explain how the optimization process was implemented in MATLAB.

The MATLAB optimization toolbox includes a function called fmincon. This function takes a constrained state vector, and attempts to minimize some desired value.

In the case of a TSS there are two different state vectors depending on what is known about the system. If the TSS parameters are known, then the state vector only includes the COEs of the CM. With an unknown TSS, the state vector must include the COEs of the CM and three additional parameters. The additional parameters required include the mass of m_1, mass of m_2, and the length of the tether. We have ignored the mass of the tether itself in this case which is generally much smaller than the end masses. The third parameter, tether length, may or may not be able to be accurately achieved using this optimization method, depending on the data. The reasons for this will be shown and discussed in the Results chapter.

The objective function to be minimized by fmincon is the Root Sum Square of the residuals (RSS). In the previous section all of the residuals were calculated between the *estimates* of both end masses and the *observed* object. Since the *observed* object obviously cannot be both end masses at the same time we have to choose which object we think the observation really represents. In order to make our choice at this point we have to compare the absolute values of the computed residuals of both end masses. The smaller of the two absolute residuals is the choice for which one will be fed into the final RSS. This applies separately to each paired set of residuals. For example, if the absolute values of a particular set of residuals at time *t* for $\mathrm{X}_{\bar{r}1p}$ and $\mathrm{X}_{\bar{r}2p}$ are $|-20|$ and $|25|$, respectively, then $|\mathrm{X}_{\bar{r}1p}|$ is the term which will be added into the final RSS, and $\mathrm{X}_{\bar{r}2p}$ is thrown out. At the same time, if the absolute values of X_{r1} and X_{r2} at time *t* are $|-20|$ and $|15|$, respectively, then $|\mathrm{X}_{r2}|$ will be added into the final RSS, and X_{r1} is thrown out. This may not make sense at this point because the same observation is assigned to both

end masses for two different residual calculations, but, ultimately, when the optimization is through, the best-fit COEs should ideally cause all of the residuals to be assigned to the same end mass.

It is vital to note that this is the point where data sorting also occurs.

As discussed in the preliminary analysis chapter, data sorting relies on comparing the *estimated* and *observed* EC range. So, the data sorting process is really a comparison of the X_{r1} and X_{r2} residuals. The final sort of the data points is based on the final iteration of the optimization process. During the optimization process the assignment of the data points will be switching back and forth between m_1 and m_2 to try and determine what minimizes the RSS best, but when the optimization process is complete the data point assignments are finalized.

We now have a way of picking which residuals will be represented in the final RSS. Since it is not known ahead of time which residual will be selected, several new terms must be defined. These new terms, which are the absolute values of the smaller residuals found in Equations (46) and (47), are termed *final* residuals for each observation *i*, and are represented as follows: final p component residual, $X_{i\ \bar{r}fp}$, final q component residual, $X_{i\ \bar{r}fq}$, final w component residual, $X_{i\ \bar{r}fw}$, and, final EC range residual, $X_{i\ rf}$.

Through a trial-and-error process, it was found that adding a weighting factor to the EC range residual term, $X_{i\ rf}$, improved performance. Since the data sorting process was such a crucial part of this research, and since there are three terms for calculating residuals in the pqw frame, there needed to be a weighting added onto the $X_{i\ rf}$ term to

ensure it retained its importance. The weighting factor chosen for this research was 10, so the calculation for the final RSS is given by:

$$RSS = \sqrt{\left[\sum_{i=1}^{n}\left(\left(X_{i\ \bar{r}fp}\right)^2 + \left(X_{i\ \bar{r}fq}\right)^2 + \left(X_{i\ \bar{r}fw}\right)^2 + 10*\left(X_{i\ rf}\right)^2\right)\right]} \tag{48}$$

where *n* is equal to the total number of *observations*.

This entire process of optimizing and sorting data enables us to determine a *final estimate* on the CM orbit, and which end mass is being tracked for each individual *observation*. But, before showing any results, we need to discuss how we obtain all of the *observation* data. The next section explains the three different sources of observation data we used for this research and how we handled each type of data.

Sources of TSS Observation Data

Up to now it has been assumed observation data is readily available and each individual observation includes the following information: tracking location, time of observation, azimuth, elevation, and slant range. This assumption is true when dealing with real-world data, but when dealing with simulated data this is not the case. Since real-world data is already obtained in the format described above there does not need to be any explanation as to how to handle this data. However, we also used two different sources of simulated data for our research. We will cover how each type of data is converted into the format described previously. Once the data is in this format it is handled just like real-world data.

The first source of simulated data comes from a simulator we developed specifically for this research. This simulator is fairly simple because it uses all of the same assumptions we have already discussed. So, a TSS simulated using this simulator always maintains nadir orientation, and follows the previously described motion in its CM COEs.

We first start this simulator with a known set of *true* COEs for the CM at a specific epoch time, t_0. Next, an end time and a timestep are selected for the simulation. We then use equation set (28) to obtain a set of COEs for each timestep after t_0 until the end time. For example, assume t_0 starts at time 0, the timestep is 20 seconds, and the end time is 400 seconds. This means there will be 21 total sets of COEs.

All of these CM COE sets calculated are then converted into the inertial IJK frame. The procedure is the same as with the estimated data; i.e. Equations (36)-(39). Once the CM IJK position vectors are known, we can calculate the end mass IJK position vectors. In order to calculate these position vectors we have to specify *actual* tether length, tether mass, and end-body masses in the simulator. Using this information and equations (40)-(44) we now have IJK position vectors for each of the end masses.

The final step of generating an individual position vector *observation* is to randomly select one of the end masses to be the observed mass depending on what we are trying to test. For example, if we want to run a simulation with approximately 50% lower mass observations and 50% upper mass observations, then we set the lower mass random percentage value to 50. Then we use a random number generator function in MATLAB to generate a number between 1 and 100. The MATLAB equation for doing this is shown next.

$$datapt = rand * 100 \tag{49}$$

If the randomly generated number equals 50 or below, then the IJK position vector coordinates for the lower mass are used for that observation; otherwise the upper mass coordinates are used. This process is then repeated for every observation. This method does not permit correlated probability from one observation to the next which would be present in actual operations.

In addition to the previous time information we also have to specify an actual start time date. This time date specifies the year, day, hour, minute, and second at which the simulation starts. The time date allows us to calculate the inertial position of any number of tracking sites in the IJK frame, as long as the latitude, longitude, and height above mean sea level are specified. Calculating a tracking site's inertial position is done as described previously using equations (11)-(15).

Next we take our tracking site IJK position vector, ${}^{IJK}\vec{R}$, and our *observed* IJK position vector, ${}^{IJK}\vec{r}_o$, and determine the tracking site's *observed* IJK slant range vector, ${}^{IJK}\vec{r}_o$:

$${}^{IJK}\vec{r}_o = {}^{IJK}\vec{r}_o - {}^{IJK}\vec{R} \tag{50}$$

The next step is to convert the IJK slant range vector, ${}^{IJK}\vec{r}_o$, into the SEZ slant range vector, ${}^{SEZ}\vec{r}_o$, using the transpose of the coordinate transformation matrix shown in equation (16). This transformation is shown next:

$$\begin{bmatrix} {}^{S}r_o \\ {}^{E}r_o \\ {}^{Z}r_o \end{bmatrix} = \begin{bmatrix} \sin(f)*\cos(t_{site}) & \sin(f)*\sin(t_{site}) & -\cos(f) \\ -\sin(t_{site}) & \cos(t_{site}) & 0 \\ \cos(f)*\cos(t_{site}) & \cos(f)*\sin(t_{site}) & \sin(f) \end{bmatrix} * \begin{bmatrix} {}^{I}r_o \\ {}^{J}r_o \\ {}^{K}r_o \end{bmatrix} \tag{51}$$

The last part of obtaining true *observations* of the data is to convert ${}^{SEZ}\vec{r}_o$ into azimuth, elevation, and slant range values for the specific tracking site. The *observed* slant range value, r_o is the easiest to obtain because it is just the magnitude of ${}^{SEZ}\vec{r}_o$. This is shown in the next equation:

$$r = \left| {}^{SEZ}\vec{r}_o \right| \tag{52}$$

The elevation angle, e, then is calculated by:

$$e = \sin^{-1}\left(\frac{{}^{SEZ}\vec{r}_o \cdot \hat{Z}}{r} \right) \tag{53}$$

Obviously if this calculated elevation angle is less than zero degrees then the object is below the horizon and the site cannot see the object, so it is not a valid observation for that site. Last, calculating azimuth angle requires calculating two azimuth values and then doing a quadrant check. The two azimuth values are calculated in the next set of equations.

$$\begin{aligned} a_1 &= \sin^{-1}\left(\frac{{}^{SEZ}\vec{r}_o \cdot \hat{E}}{r * \cos(e)} \right) \\ a_2 &= \sin^{-1}\left(\frac{{}^{SEZ}\vec{r}_o \cdot \hat{S}}{-r * \cos(e)} \right) \end{aligned} \tag{54}$$

The quadrant check is done as follows:

If $a_2 > 0$ and $a_1 > 0$ then $a = a_1$; If $a_2 < 0$ and $a_1 > 0$ then $a = 180° - a_1$

If $a_2 < 0$ and $a_1 < 0$ then $a = 180° - a_1$; If $a_2 > 0$ and $a_1 < 0$ then $a = 360° + a_1$

Noise must be added to this *perfect* data to get a realistic test of the data sort and tether OD algorithm. In order to add *realistic* errors we need some information about the

tracking site. As talked about in the preliminary analysis chapter every tracking site has some error in its measurements. If it is known what the statistical error is for each of the three measurement values then these errors can be used in a simulation to add realistic errors to our *perfect* azimuth, elevation, and slant range data.

MATLAB has another random number generator that works for adding realistic errors to our readings if the tracking site statistics are known. This random number generator generates a normally distributed random number with a zero mean, and the standard deviation and variance both equal to 1. The random number generated by this function can be directly multiplied by the $1s$ errors for a site to add realistic error to the data. The equation used to generate this realistic random error is shown next.

$$e_o = e_o + randn * s_e \tag{55}$$

The term *randn* is the random number generator function in MATLAB and s_e is the statistical $1s$ error for the site's elevation readings. The same type of equation can also be used to generate realistic random errors in the *observed* slant range and azimuth.

This entire procedure takes simulated COEs and converts it into observation data with realistic errors. From this point the same procedures as before are used to estimate the epoch time COEs and sort the simulated data.

The second source of simulated data used for this research is a high-fidelity tether simulator program called TETHERSIM^{TM} developed by Tethers Unlimited, Inc. TETHERSIM^{TM} uses a 4th order Runge-Kutta algorithm for the orbital dynamics propagation of the satellite, end masses, and tether elements. It uses a more comlex gravitation model by using an 8th order spherical harmonic model of the geopotential. In

addition, it includes a 1st order lunar gravity model. TETHERSIM™ also uses the International Geomagnetic Reference Field for simulating a geomagnetic field model. Finally, this simulator accounts for air drag on a TSS by using the MSISE90 Neutral Atmospheric Empirical Model, but this drag model is only activated if a TSS is below 400 kilometers in altitude. Overall, the main reason for using TETHERSIM™ is to test the estimation method with a more realistic TSS model. Using this model we can obtain an idea of how well the COE estimation and data sorting method works when other affects such as libration, higher-order perturbations and tether tension are added.

For our purposes, TETHERSIM™ was used to provide an output of IJK position vectors for the end masses of a TSS. We then randomly selected which end mass would represent a particular observation, and then the randomly selected IJK position vectors were converted into *observed* azimuth, elevation, and slant range readings for a specified tracking site as described previously. We then added error into those readings as discussed before by using the tracking site's statistical $1s$ errors. Finally, the new *observed* azimuth, elevation, and slant range readings were used to estimate the epoch time COEs and the data sorting was accomplished as previously discussed.

Estimation and Data Sorting Conclusion

This entire chapter outlines the tools and methods we have used to solve the data sorting and OD problems for a TSS. The next chapter, Results, describes the specific

cases we have attempted to solve using this method. We also analyze the results of these cases and show the strengths and weaknesses of this methodology.

V. Results

The results from this research show great promise in helping to sort data and determine estimates of the orbit for the CM of a TSS. In addition, it also shows some promise for helping to identify an unknown TSS. In order to validate these statements this chapter shows the results of several specific cases using data from all three sources of TSS data discussed in the methodology section. The first case analyzed is a baseline case developed in great detail to ensure the case is well understood. The rest of the chapter analyzes changes in the key parameters of this baseline case. The variations in these parameters help to show the strengths and weaknesses of this TSS data sorting and OD method.

Baseline Case

One of the key aspects of the baseline case is the TSS parameters are known. This means the optimization process assumes the masses of the various objects and the length of the tether are known and are not variables that need to be estimated. There are several key parameters that make up the baseline case. These parameters, along with the analysis tools developed for this research, help provide a great baseline for understanding how well this estimation method works. The first item we set up for this baseline case is

the key parameters. The second item includes using the analysis tools to give us an estimate of how well we expect the methodology to work.

Baseline Case Parameters.

The parameters used for the baseline case include specific tether information, tracking site details, and the orbit.

The specific tether information used to set up the baseline case include the length of the tether, the masses of each of the end-bodies in addition to the tether itself, and the orientation of the tether in space. First, the length of the tether for the baseline case is set at $4.023 km$ (Purdy et al., 1997:2). The length was set at this value because it is the only value which can be used to provide a comparison between simulated data and a large supply of real-world data since TiPS is the only TSS which provides a large source of real-world data. The mass properties of the TSS are also set to similar values as TiPS; therefore, the lower mass is $95.3 lbs$ ($43.32 kg$), the upper mass is $22.4 lbs$ ($10.18 kg$), and the tether mass is $12 lbs$ ($5.45 kg$) (Purdy et al., 1997:2). The final piece of information is the orientation of the satellite in space. While looking at the baseline case we used the simulator we developed to analyze the results of the methodology, so the orientation of the TSS is perfectly nadir-oriented. Later, we will analyze the effects of libration by using TETHERSIM^{TM} data.

The tracking site details needed to do a complete simulation of the baseline case include the location of the site, and the tracking site errors associated with the azimuth, elevation, and slant range readings. The simulated tracking site used for this baseline case has the following values for latitude, longitude, and altitude above mean sea level:

$30.57242°N$, $86.21485°W$, and $0.03640km$. The tracking site errors simulated for this tracking site are: $s_r = 0.021km$, $s_e = 0.023°$, $s_a = 0.019°$.

The final piece of information needed for this baseline case is the *true* orbit of the TSS CM at the epoch time, and the source of the observations. The epoch time and the COEs for the *true* orbit at the epoch time are listed next:

Epoch time:

Year = 1997 Day = 210 hour = 11 Zulu minute = 30 second = 30.000

COEs:

$$a_{cm}(t_0) = 7400km$$
$$e_{cm}(t_0) = 0.004$$
$$i_{cm}(t_0) = 65.3°$$
$$w_{cm}(t_0) = 70.0°$$
$$\Omega_{cm}(t_0) = 220.45°$$
$$n_{cm}(t_0) = 70.0°$$

The source of the data is a random mix of 50% upper mass data and 50% lower mass data. In addition, the total time and the time step used for the baseline case is 200 seconds with data points every 10 seconds.

Baseline Case Estimation.

The first item which needs to be shown is the first guess of the CM COEs. The Herrick-Gibbs *first-guess estimate* determined by Equations (10)-(27) at the epoch time is shown next.

CM COEs- 1st Estimate at Epoch time determined by Herrick-Gibbs

$$a_{cm}(t_0) = 7366.8km$$
$$e_{cm}(t_0) = 0.023539$$
$$i_{cm}(t_0) = 65.626°$$
$$w_{cm}(t_0) = 41.84°$$
$$\Omega_{cm}(t_0) = 220.16°$$
$$n_{cm}(t_0) = 98.855°$$

However, to demonstrate the robustness of this algorithm, we will assume we have an even worse *first-estimate*, so outlined next is the first guess of the CM COEs at the epoch time.

CM COEs- 1st Estimate at Epoch time

$$a_{cm}(t_0) = 7000km$$
$$e_{cm}(t_0) = 0.0001$$
$$i_{cm}(t_0) = 45.3°$$
$$w_{cm}(t_0) = 25.0°$$
$$\Omega_{cm}(t_0) = 190.2°$$
$$n_{cm}(t_0) = 20.0°$$

The next step propagates the *actual* COEs using equations (28)-(35) for the specified time period and then determines the *actual* CM IJK position vectors for those COEs using equations (36)-(39). After determining the CM IJK position vectors, the *actual* IJK position vectors for each of the end masses are determined using equations (40)-(44). Now that the *actual* end mass position vectors are known, we randomly selected one end mass for each observation using our 50% criteria and equation (49). With each observation being allocated to a particular end mass we then calculated perfect *observed* azimuth, elevation, and slant range readings for the specified tracking site using equations

(50)-(54). These perfect readings then had error added into them randomly using the defined statistical tracking site errors and equation (55).

The process of OD and data sorting begins at this point as defined in the methodology. First, the *observed* azimuth, elevation, and slant range readings are converted back into IJK values using equations (10)-(17). The optimization routine is then called which does all of the calculations for the propagation of the *first-guess* COEs. The first-guess COEs of the CM are propagated over the time period for each particular observation using equations (28)-(35). Once the *estimated* COEs of the CM are calculated for each observation those COEs are converted back into IJK position vectors and the *estimated* end mass pqw position vectors are also determined using equations (36)-(45). The *observed* IJK position vectors are also converted into pqw position vectors using equation (45). Once the *observed* and *estimated* pqw position vectors are determined, the residuals are calculated as described in equations (46)-(47).

While MATLAB runs the fmincon function on the RSS function calculated in equation (48), the CM COEs at the epoch time are being refined and the data sorting is occurring. Once the minimization function is complete, the final estimated COEs are determined. The final estimated COEs for this particular run of the baseline case are shown next.

CM COEs- Final COE Estimate w/errors in Data at Epoch time

$$a_{cm}(t_0) = 7394.7km$$
$$e_{cm}(t_0) = 0.0037208$$
$$i_{cm}(t_0) = 65.309°$$
$$w_{cm}(t_0) = 59.468°$$
$$\Omega_{cm}(t_0) = 220.45°$$
$$n_{cm}(t_0) = 80.538°$$

Considering the very poor initial guess on the initial CM COEs, this is a good estimate. In fact, the final *estimated* COEs determined by this process, in general, yield a smaller RSS than a *perfect guess* solution when errors are added into the data. That is, the *estimated* orbit is a better fit for the corrupted observations than the *true* orbit. In the case of dealing with *perfect* data the *estimated* COEs were essentially equal to the *actual* COEs. The real key to this estimate though, is how well it does for properly identifying the observations. Figure 6 shows how well a plot of the EC ranges matches up. The 'x's correspond to the EC range *observations* with error added in them. The lower set of dots represents the *estimated* EC range of the lower mass over time. The upper set of asterisks represents the *estimated* EC range of the upper mass over time. The '+'s represents the *perfect observation* with no error added. The *perfect observation* is the *observed* object's true state at time *t*. The 'O's represent the lower mass *estimates* determined using *perfect* data. The squares represent the upper mass *estimates* determined using *perfect* data. The same convention will be used throughout all of the EC range plots except where noted.

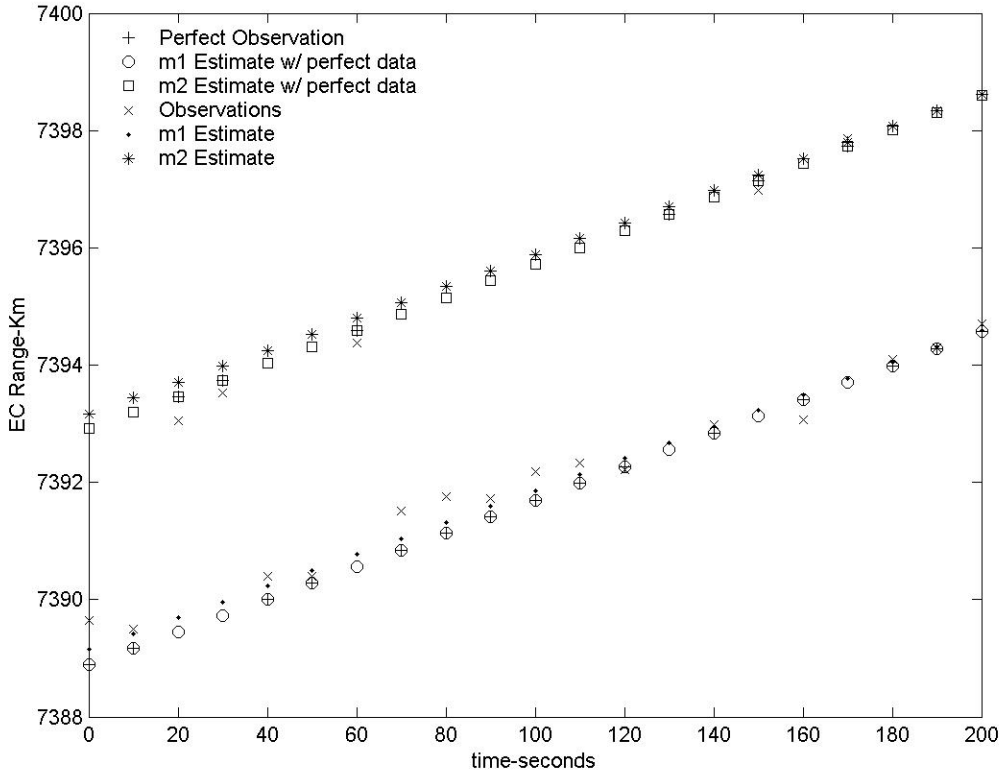

Figure 6- Baseline Case Plot of Observed versus Estimated EC Position Vector Magnitudes

This plot matches up very well, and the validation of this plot occurs by comparing the end mass assignments of the *observed* data points and the *estimated* data points. When this comparison is done, it can be shown that **all** of the *estimated* data points have been assigned to the appropriate end mass.

Comparing Baseline Case Results and Tracking Site Error Analysis.

The key for understanding the excellent data sorting results is the analytic tracking site error analysis. Figure 7 shows the EC range uncertainties for this case. The uncertainty envelope corresponds to $3.62s_r$. This value was selected because the

envelope for the upper and lower masses intersect at exactly 28°, which is the lowest elevation for the observations of this case.

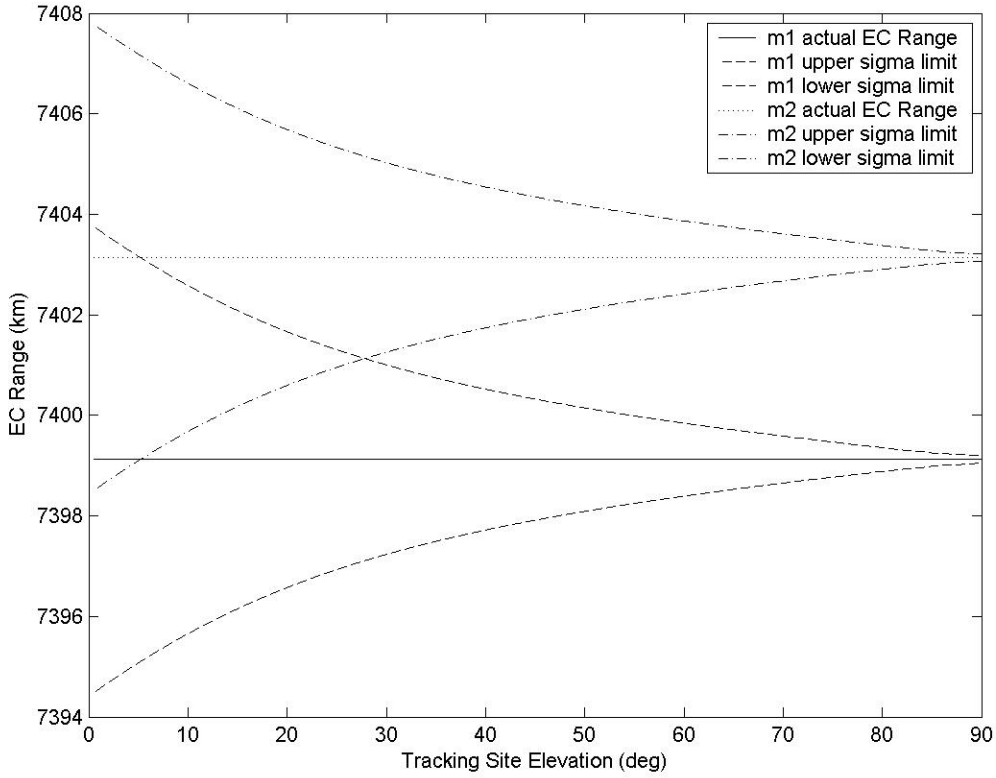

Figure 7- Error Analysis for Lowest Elevation Tracking Data for Baseline Case

For this value of uncertainty, we expect 0.01% (Beyer, 1991:503) of all observations to lie outside the envelope (which would result in the data point being tagged to the wrong end body). This means we should ideally achieve 99.99% data sorting accuracy for this baseline case even at our lowest elevation reading. The higher elevation observations should have an even higher percentage, but since we are looking at approximately 100% accuracy we can see why the baseline case did not misidentify any of the observations.

The baseline case had a total of 21 observations to correctly identify. To test whether any sorting errors would occur with more data, we ran the baseline case ten times in a row with different random errors. Running the baseline case ten times gives us a total of 210 observations, so this is a much larger sample of data points where we can see if any incorrect identifications occur. The results of doing this large run still gave us a total of zero misidentifications for all of the runs which helps to validate the usefulness of the tracking site error analysis tool.

The results of the baseline case show great promise in data sorting and determining an estimated set of COEs for a TSS. But, the results shown previously only apply to one specific case. The next section examines the affects of varying certain TSS and tracking site parameters. The strengths and weaknesses of the chosen methodology for solving this problem are highlighted.

Parametric Studies of Different TSS and Tracking Site Parameters

There are several key parameters that help provide an understanding of the strengths and weaknesses of this methodology. To show the importance of these parameters, they will each be discussed and the results of some studies done on each will be shown. The parameters we are interested in studying include time/number of observations, and tether length/elevation angle. We will also address separately several other items which may not be considered parameters but still provide insight into this

problem and they include the affects of mixed versus single end mass data, TETHERSIM^{TM} results, and the unknown TSS.

Time /Number of Observations Variations.

One of the weaknesses of this optimization process is that a sufficient number of data points are required over a certain time period. When analyzing the baseline case by only changing the amount of time and the number of observations, the problem of not having enough observations became apparent. Changing the total observation time did not cause any data sorting problems for the baseline case. However, it is apparent that the less time an object is tracked the worse the estimated COEs will be. Of more importance however, is the number of observations. Running the baseline case over a time period of 100 seconds and obtaining 11 total observations 10 seconds apart, the results became very poor. The COE estimate was not very accurate, and some of the observations were inappropriately tagged.

Due to the limitless combinations of different TSS and time/number of observations, it becomes difficult to define a timeframe and exact number of observations required for good results, but after doing many simulation runs, a minimum of 15 observations almost always was needed, and 20 or more observations is recommended. As stated previously, the timeframe of the observations can vary to some extent but, in general, the shorter the timeframe the worse the COE results, so longer timeframes are recommended.

Tether Length/Elevation Angle Variations.

The main conclusion of this section is that, in general, the longer a tether, the less likely misidentification will occur with mixed data. But, the longer a tether is, the less likely it is that a tracking site will receive mixed data.

The tracking site error analysis tool illustrates the effect of tether length. Compare Figure 7 for a 4*km* tether with Figure 3 for a 2*km* tether. Both of these figures have tracking sites with the same errors. The only difference in the two is the tether length. This shows that a 2*km* tether has a much greater chance of misidentifying an observation.

The tracking site error analysis was verified by feeding in a *perfect* guess for the COEs and comparing the number of observations correctly identified with the number of observations predicted to be identified correctly by the tool. Using over a thousand data points for this validation technique the final number of correctly identified observations was always within $\pm 1\%$ of the predicted value. For example, figure 3 predicts for a 2*km* TSS under the conditions shown at approximately 8.5 degrees elevation we should receive 84% correctly identified objects. Performing a data sort for observations near 8.5 degrees, the percentage obtained for correctly identified objects using 1050 data points was 84.1%.

As discussed before, this analysis tool provides a best-case confidence level. So, it should generally be assumed that results obtained using less-than-perfect COEs will yield lower results. As the estimated COEs become farther away from the truth, then the confidence level drops even more so obtaining estimated COEs which are as close to the truth as possible is important to yield results close to the maximum confidence level.

Single Object Only Data

While other TSS filter methods rely on having data from only one object, this estimation method assumes in advance that we do not know if we have data all from one end body. Because of this assumption, when we do have data all from one end mass of a TSS, this sometimes causes a weakness to show up in the method. The weakness can show up in one of two ways. First, the optimization method may 'lock' onto the completely wrong object and assign all of the data points to the wrong end mass. Another common problem is that the optimizer will try and run the estimated positions of both end masses through the observations. This problem is illustrated in Figure 8.

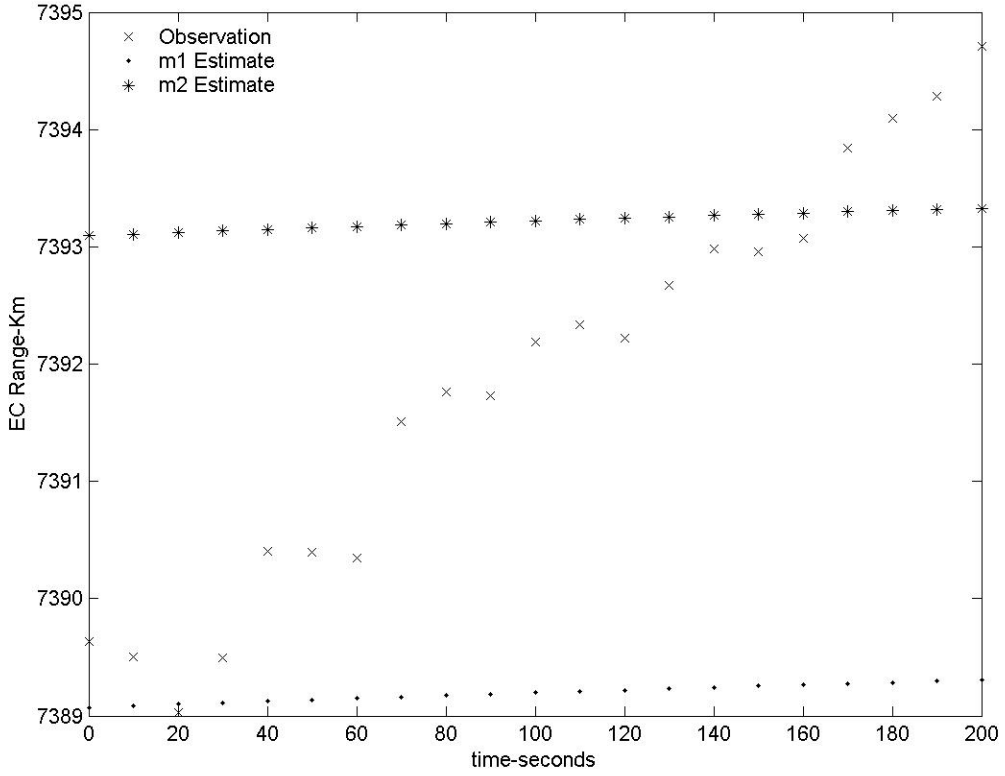

Figure 8- Bottom Mass Only Data for Baseline Case

Even though all of the data points are observations of the lower mass, the general optimization has tried to assign the observations to a mix of lower and upper mass observations. The best way to handle this problem is to do two additional optimization runs. The first additional run calculates all residuals by always assuming the observations are from the upper mass, and the second additional run assumes all of the observations are from the lower mass. What this allows us to do is compare the three RSS values from all three runs. We then assign the final solution to the lowest of the three RSS values. Figure 9 shows the best of the three solutions for the previous case of all lower end mass observations on our baseline case.

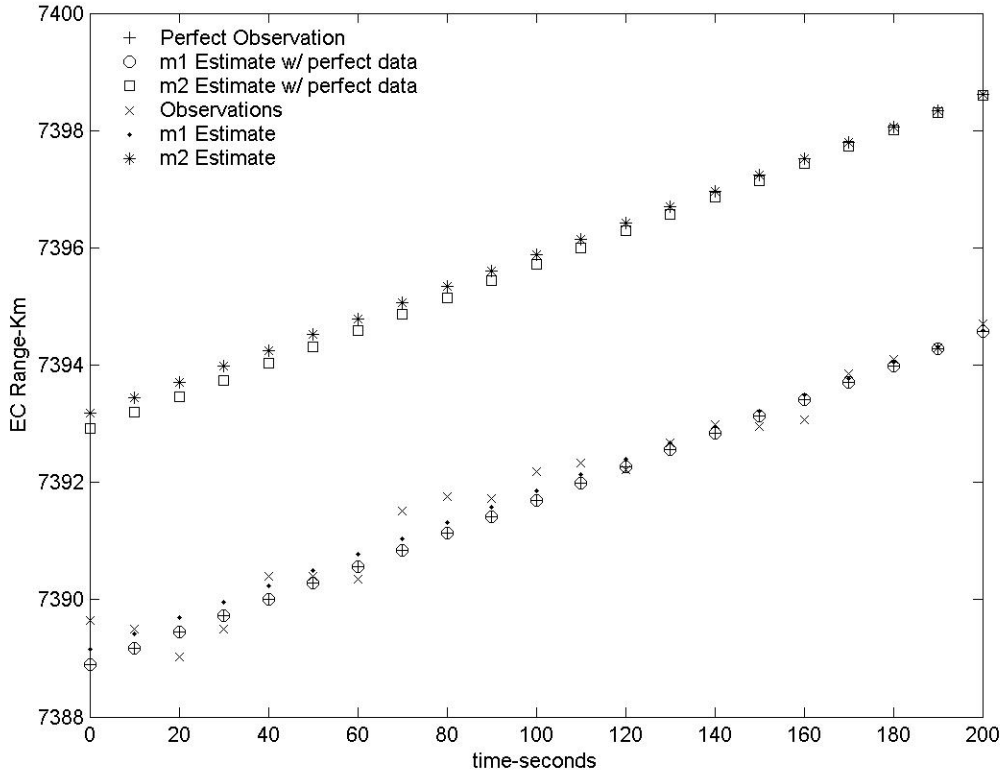

Figure 9- Bottom Mass Only Data for Baseline Case Showing Best of 3 Optimizations

This figure shows that the perfect and imperfect data observations have all been assigned to the lower mass, which is the correct solution. One problem with doing these additional optimization runs is the computing time required increases because now three optimization runs are done instead of one, but there is no good way to get around this problem. Also, the tracking site errors can cause problems with this because it still may assign observations incorrectly if the error is large enough. Therefore, using tracking sites' with smaller errors and higher elevation viewing angles is important for accurate results.

One benefit of single-object only data for the known TSS case is the number of observations required decreases in some cases. This variation of the known TSS allowed the optimization to obtain fairly good results even with as few as 5 observations. However, this only applies to the *known* TSS case. As will be shown later the unknown TSS case still requires many observations to yield accurate results.

TETHERSIM™ Results

As discussed previously in the sources of data, TETHERSIM™ is a tether simulation program that generates more realistic tether motion. The purpose for testing this data is to see how additional factors such as libration and higher-order perturbations affect the optimization process. The TETHERSIM™ data analyzed is similar to the baseline case in that it is a 50% mix of data from both end masses and the tether properties are the same as the baseline case. As shown in Figure 10 the data sorting

results for TETHERSIM™ are still very good as long as the tracking site errors remain reasonable.

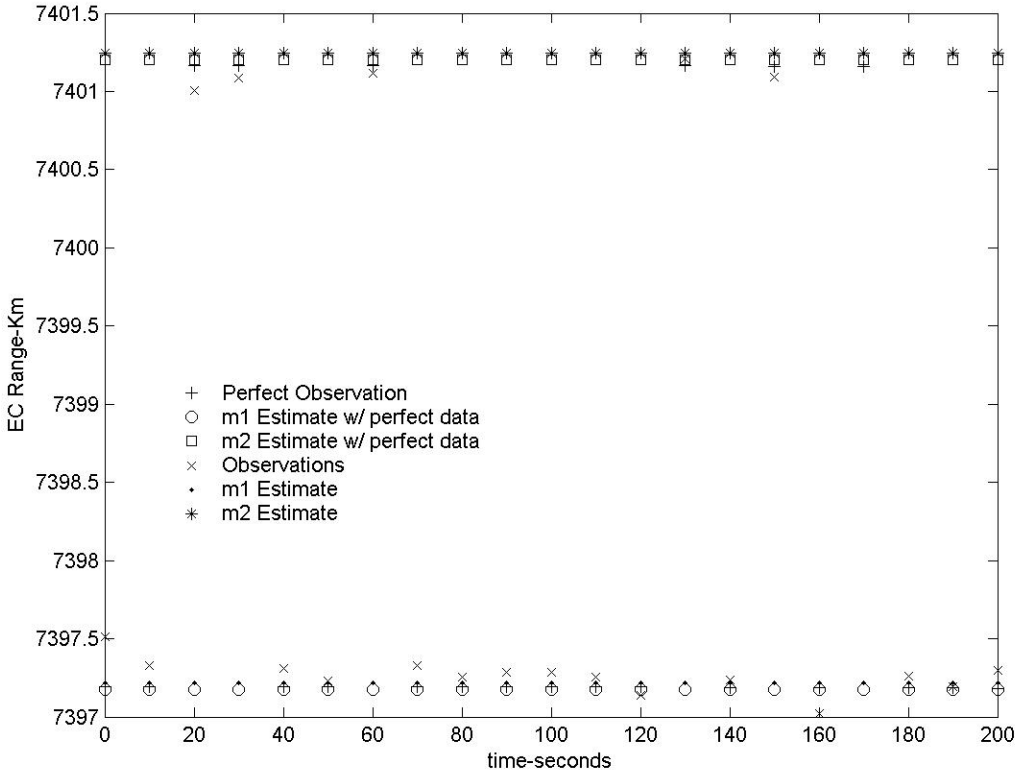

Figure 10- TETHERSIM™ Known TSS Mixed Data Results

The RSS results for TETHERSIM™ data are higher than the RSS results from the simulator developed in-house. The main reason for this is because the libration angles now essentially make the tether look 'shorter' than it really is. As can be seen in the figure, even the perfect observations are not completely aligned with the estimated solution of that data. Although there are still significant residuals (because the model in the estimator does not match the more sophisticated model used to generate the data), all all of the data points have been assigned properly.

The libration angles for this particular case of TETHERSIM^{TM} data were approximately 5-7 degrees for both in-plane and out-of-plane libration angles. The results from this case help to show that as long as libration angles are reasonable then this estimation method is effective.

Unknown TSS

After analyzing numerous known TSS cases, this estimation method shows great promise for doing data sorting and OD for known TSS's. However, the other important part of this research is to see how well this methodology applies to the unknown TSS case. We will look at an unknown TSS similar to the baseline case, and then we will also look at the results of a longer unknown TSS case where the data comes only from one end-mass. But, first, we investigate what happens when this optimization method is applied to a single-body satellite.

Single-body Satellite System.

The reason for applying this methodology to a single-body satellite system is we want to ensure that a satellite which is *not* a TSS is not identified as a TSS. This was accomplished by using the in-house simulator to generate observations for a satellite that has a tether length of zero. The simulator used the same COEs and the same tracking site errors as the baseline case. The estimation process gave back very good estimates of the COEs for the CM for this case, and, more importantly, it identified that the CM was approximately zero kilometers away from the observed object. This distance to the CM

from the observed object is the key to identifying whether a satellite is part of a TSS or not. If the distance is close to zero it can be assumed that the satellite is a single-body satellite. However, there is one problem with this assumption. If a TSS has a large portion of the mass all in one end-body such that the CM is very near that mass and if that mass happens to be the observed end mass then it will look like this is a single-body satellite. There is nothing which can be done in this case because the observed end mass is essentially traveling on a normal Keplerian orbit, and unless the other tiny mass is observed there is no way to tell that the larger mass is part of a TSS.

One other problem which might occur with a single-body satellite depends on the tracking site errors. If the errors are large enough the optimization method may try and assign the observations to multiple end masses and it may say the single-body satellite is a short TSS and that both end masses have been observed. This is why reducing tracking site errors is important for properly identifying TSS.

Unknown Baseline Case TSS.

When analyzing an unknown TSS using this methodology it is much better to obtain observations from both end masses if at all possible. By observing both end masses it becomes much easier to get a good estimate of the length of the tether. This is demonstrated in Figure 11 by taking the baseline case and saying it is an unknown TSS.

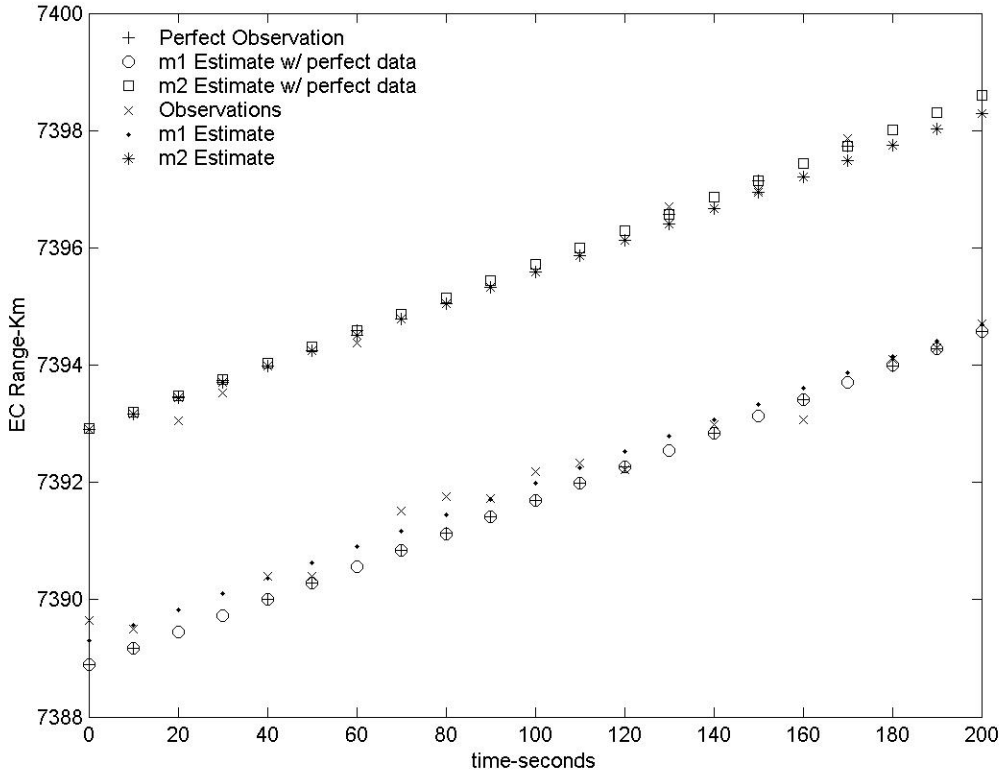

Figure 11- Baseline Case Unknown TSS Results

The determined tether lengths of the perfect observations and imperfect observations are 4.0231*km* and 3.6117*km*, respectively. Considering the actual length of the tether is 4.023*km* these results are extremely good. In addition, all of the observations were assigned to the correct end mass, and the COE results even for the imperfect data were still fairly good as shown next.

CM COEs- Final COE Estimate for Data with Errors for Unknown Baseline Case TSS

$$a_{cm}(t_0) = 7394.2km$$
$$e_{cm}(t_0) = 0.0036867$$
$$i_{cm}(t_0) = 65.309°$$
$$w_{cm}(t_0) = 60.06°$$
$$\Omega_{cm}(t_0) = 220.45°$$
$$n_{cm}(t_0) = 79.944°$$

One thing to note about the COE results are that the argument of perigee and true anomaly terms are not very close to the actual values which are both $70°$. The orbit is nearly circular, so perigee is hard to observe. However, the sum of the two values for the *estimate* and the *actual* values is equal to $140°$. The more total time and data used for the unknown case the closer the COEs will be to the truth. Of course, this may also make it necessary to use multiple tracking sites to obtain enough good observations over enough time to get as accurate an estimate as possible.

Single End Mass Observations for an Unknown TSS.

The last scenario we want to look at to analyze the unknown TSS is to look at what happens when all of the observed data of an unknown TSS is from only one of the end masses. Since this is more likely to occur with longer tethers than shorter tethers we chose to analyze a TSS that has different tether parameters, but the orbit is the same as the baseline case. A longer time frame of data is needed to obtain accurate results with a single-mass only observed unknown tether, so the time parameters were set to 400 total seconds with data points taken every 20 seconds. The tether parameters for this new case include the following values: m_1 and m_2 are of equal mass and the tether is $20km$ long. This means the CM of the TSS is $10km$ away from each end mass. All of the observed

data was taken from the bottom mass for this case. The results of analyzing this case are shown next in Figure 12.

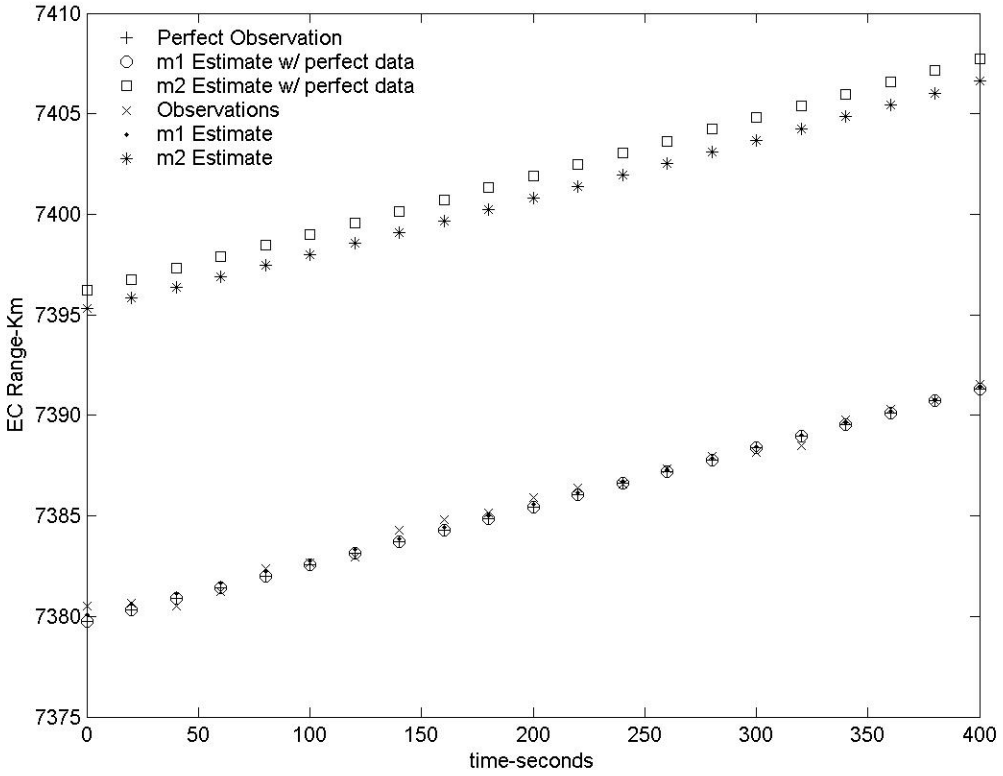

Figure 12- 20 km Unknown TSS Results

The most important item to note about this figure is that with perfect and imperfect data all of the observations have been correctly assigned to the lower end mass. The next analysis of this case comes from looking at the estimated COE and tether parameter results. The final estimated COEs are still fairly accurate considering how little is known about the TSS.

CM COEs- Final COE Estimate for Data with Errors for 20km Unknown TSS

$$a_{cm}(t_0) = 7406km$$
$$e_{cm}(t_0) = 0.00406$$
$$i_{cm}(t_0) = 65.305°$$
$$w_{cm}(t_0) = 78.484°$$
$$\Omega_{cm}(t_0) = 220.45°$$
$$n_{cm}(t_0) = 61.517°$$
$$m_1 = 3.4218kg$$
$$m_2 = 10.646kg$$
$$P = 15.227km$$

The *estimated* argument of perigee and true anomaly add up to the same value as the *actual* argument of perigee and true anomaly of $140°$. For this case, the masses of the two bodies and the tether length are unobservable individually. The only observable quantity is the distance of the *observed* body to the CM. There are an unlimited number of combinations of tether lengths and masses which can provide the correct distance to the CM in this situation, so these numbers in and of themselves do not mean anything, but it is the combination of the data which provides the important information. Using equation (40) to determine the distance from the CM to m_1 we find this distance is $11.523km$. In analyzing the perfect data, the distance is found to be even closer to the truth at $10.001km$.

The unknown TSS case with only data from one end mass is by far the most difficult case to analyze. The results for the case shown are very good, but there are also times when the results have not been nearly as spectacular using this method. In order to come up with accurate results for this situation it is extremely important the data be as

accurate as possible. In addition it definitely takes more observations and time to do good data sorting and OD.

Real-World TiPS Results

Analyzing real-world data presents additional problems which do not appear when dealing with simulated data. The most obvious difference with real-world data is that we have no truth to compare the results. Specifically, there is no way of knowing if the data sorting has occurred properly. As far as the OD process goes, we can compare the results with long-term estimates of the orbit of TiPS, but this only gets us an approximate orbit.

One other important factor when dealing with real-world TiPS data deals with the CM of TiPS. There is a dipole antenna sewn into the tether itself for TiPS which is less than 100 meters away from the CM. This causes problems because the tracking sites occasionally obtain tracking data inadvertently from this dipole antenna. This means we now have 3 separate objects which must be taken into account. In addition, due to the actual masses of each end body of TiPS the CM is less than 1 kilometer away from the lower end mass. Using our tracking site error analysis tool to estimate the confidence level of a particular set of observations, we observe that the dipole antenna may cause significant problems for data sorting. Figure 13 shows a plot of sigma values relating to TiPS and a tracking site with the same errors as given before. The CM *actual* EC range and sigma limits have been included in this figure as solid dots.

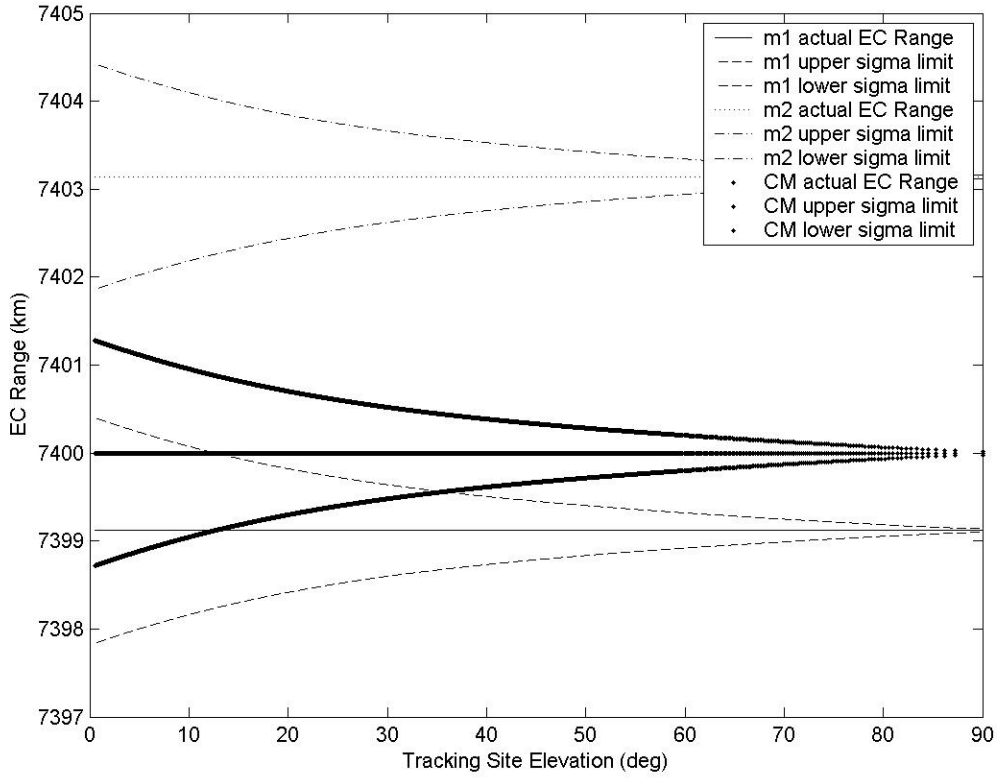

Figure 13- TiPS Tracking Site Error Analysis

This figure shows that even with observations at approximately 40 degrees elevation there is only an 84% chance of correctly identifying the lower mass and CM observations and this is with a perfect COE estimate. This means the chances of telling the dipole antenna from the lower mass are even lower since we do not know the actual COEs.

We analyzed real-world tracking data for TiPS taken in July 1997. We chose this specific set of tracking data because of the favorable tracking site viewing geometry. Even with the favorable geometry for this tracking site pass, the highest elevation angles obtained from the site were approximately $67°$, while the lowest elevation angle was

approximately $3°$. Many other tracking site passes looked at had much worse viewing geometries than this pass. The EC range plot for this data has a couple of differences from previous EC range plots. The 'x' locations for the observations now include $3\,s_r$ bars to show the area where the *actual* value is most likely located. These sigma bars are not a true representation of the *real* tracking site errors, but are estimates instead. The *estimated* CM of TiPS is shown in this figure as the 'O'. The *estimated* lower and upper mass locations are indicated by their appropriate symbols as shown in the legend.

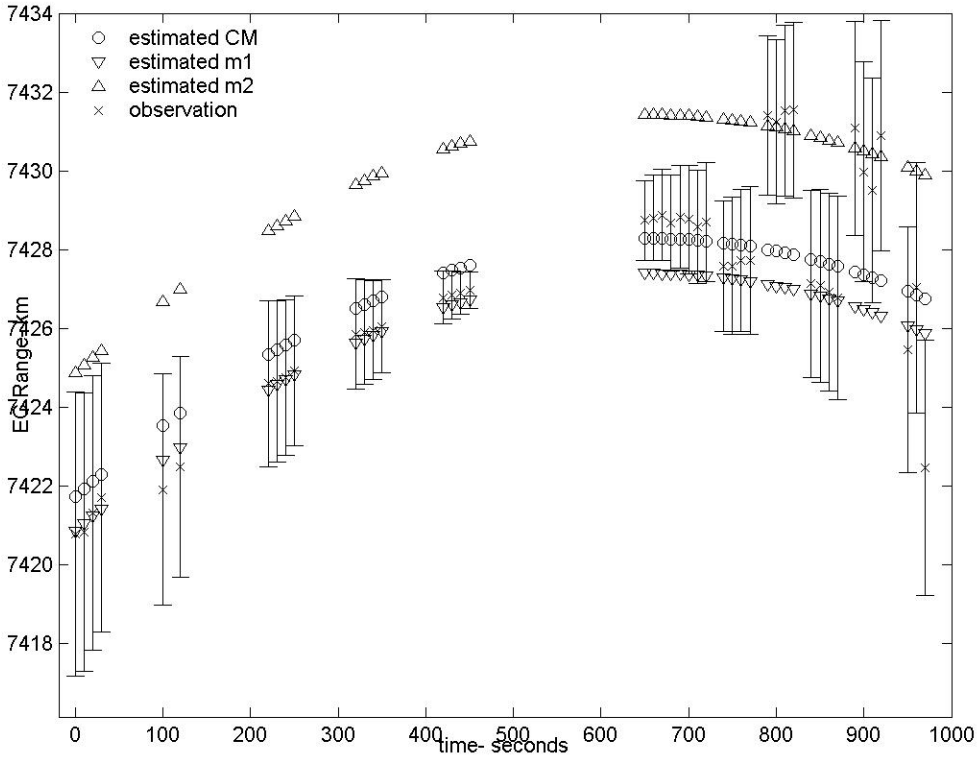

Figure 14- TiPS EC Position Vector Magnitude Plot

This figure shows how well the estimation process has done even with the real-world difficulties mentioned previously. The locations of the 'x's compared to the

estimated locations of the end masses and the CM match well, especially for the higher elevation angle observations indicated by the shorter covariance bars. The chart seems to indicate all of the observations taken before 500 seconds are observations of the lower mass. After the large time gap in the middle is when it appears the tracking site started to observe the CM and the upper mass. The results of this figure show how important data sorting is when dealing with a TSS because it is obvious that not all of the observations are of the same end mass, even though *all* of the data was supposed to be from the *lower* end mass.

Also, looking at the figure, the very last 'x' looks like it is nowhere near the estimate or the rest of the data points, but looking at the upper sigma limit for that particular observation it may just be a very bad reading on the lower end mass because the upper limit is near the estimated location of the lower end mass. This is a great example of how low elevation data can potentially cause problems for sorting observations. The next part of analyzing the solution obtained by this optimization is to look at the COEs for the estimate.

CM COEs - Final Estimate at Epoch time for Real-World TiPS Data

$$a_{cm}(t_0) = 7396.2km$$
$$e_{cm}(t_0) = 0.0043427$$
$$i_{cm}(t_0) = 63.423°$$
$$w_{cm}(t_0) = 357.67°$$
$$\Omega_{cm}(t_0) = 144.48°$$
$$n_{cm}(t_0) = 142.93°$$

These COE results are consistent with long-term plots of TiPS data, especially when looking at how eccentricity changed slowly over time for TiPS and how the argument of perigee did not seem to change over time at all.

One more item which can be inferred from the plot of the TiPS data is it appears as long as a tracking site continues to take observations it usually tracks the same end mass. The times when it switches to one of the other objects usually occur after a time lapse. This might either occur because the site temporarily loses the object or the time interval is built in for some other reason. The other case where the data switching seems to occur is the very low elevation data case that corresponds to the last set of three observations. For example, the last three observations look like they are mass 1, CM, and then mass 1 again. Whether this is due to errors or tracking the different object is unclear, but this also helps to illustrate how low elevation data causes problems for tracking a TSS.

Overall, the results of analyzing real-world TiPS tracking data shows that this method definitely has promise for helping to sort out observations and determine a decent estimate of the COEs for the CM of a TSS.

VI. Conclusion

Data sorting and orbit determination of tethered satellite systems is a difficult problem. There are many complications which can arise when dealing with a TSS, but this research has helped to show how optimization can potentially be used to help solve the data sorting and OD problem. There are several operational implications to be taken away from this research. In addition, there is also room for future research on this topic. Both of these items are discussed next to help show the way ahead for further understanding of TSS's.

Operational Implications

There are three important operational implications to be taken away from this research. First, and foremost, it is fairly apparent that accurate data sorting and OD for TSS requires more total observations and more accurate observations than normal single-body satellites. In fact, where a decent COE estimate for a single-body satellite can be obtained using only 2 or 3 observations using techniques such as Herrick-Gibbs, this is probably not possible for tethered satellites. The method used in this research tells a TSS apart from a single-body satellite by analyzing the differences in the motion of the CM to the motion of the observed object(s). In order to obtain an accurate estimate there needs to be a significant amount of data in order to tell the two motions apart.

The second important implication is that real-world tracking site errors have to be accounted for when discussing the possibility of identifying TSS's. If tracking site errors are very large, it might not be possible to sort the data or obtain accurate COE estimates. It does not do any good to only look at perfect or near-perfect simulated data if the real-world data is so bad that no accurate information can be obtained from it. The more accurate a tracking site is, especially in elevation angle readings, the better chance there is of observing the difference in the motion of a single-body satellite versus a TSS. The tracking site error analysis tool is an excellent tool to help identify the limits of a tracking site's data sorting capabilities. In addition, tracking site errors also play a role in determining how well the COEs can be estimated.

Third, any methods of trying to obtain accurate COE estimates of the CM for a TSS need to account for possible mixed data. It does not do any good to just assume all tracking site data comes from one end mass when in reality it has definitely been shown that this is not the case. Any further research into optimization methods or filter methods concerning TSS's should take note of this important fact because the results yielded will most likely be poor if the data sorting problem is not taken into account.

Future Research

A future in which TSS's provide unique capabilities in space is approaching rapidly. TiPS is just the beginning as far as tethered systems are concerned, and therefore further research should be undertaken to help understand the unique nature of these

satellite systems. Specifically, it is important to understand how to obtain accurate COEs for TSS's because as more tethered systems are deployed in space the harder it will be to keep track of everything unless accurate COEs are determined.

This research shows good promise for using optimization as a method to help identify accurate TSS COEs. One area of potential future research deals with the optimization process. Looking at other optimization methods or different optimizer tools, such as a FORTRAN optimizer, may help provide an understanding of what methods work best. The MATLAB optimization process has been shown to yield fairly good results for the cases analyzed, but since there are so many different cases to potentially analyze there is no way to say this optimization tool is the best.

Another area of further research for data sorting and OD of TSS's is to try and take into account tether libration. This research analyzed the data by always assuming the nadir-oriented case. The TETHERSIM™ and real-world results show this is a good starting point to analyze tethered systems which have fairly low libration angles. However, it may become necessary to account for larger libration angles by adding in libration angles as a part of the optimization routine. Of course, this will increase the optimization solution space and this may cause problems, but it is worth researching further. In addition, if a future method can account for libration and determine decent estimates on the libration this may help provide insight into the attitude dynamics of the TSS as well as the orbital dynamics.

Finally, since there is so little real-world TSS data doing further research with the higher-fidelity tether simulator programs, such as TETHERSIM™ becomes an important way to simulate and obtain more realistic data. Further research was not done using more

TETHERSIM™ data because it takes a large amount of time to fully understand how to use a simulator as complex as TETHERSIM™. That is why only a couple of baseline cases were analyzed using this type of data. If more cases can be developed in this simulator, or another tether simulator program, it may help provide more insight into the best way to handle data sorting and OD for TSS's.

Understanding how to sort observations and obtain accurate COEs for a TSS has real-world operational impact. This is why further research concerning these problems should be continued. Without a more in-depth understanding of how to accomplish these tasks, as more tethered systems are deployed in space, this could potentially cause real-world problems. This research has helped provide a step in the right direction for understanding data sorting and OD for TSS's, but further research is definitely warranted.

Appendix: Primary MATLAB Programs

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
% Capt Mark Faulstich- AFIT/ENY
% TSS data sorting & OD Final Optimization Program
% Final Version- 6 February 2004
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
clear all % Clears all variables in memory
format short g %Sets screen output format
warning off MATLAB:divideByZero % Turns off MATLAB divide by zero warning
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Global Variable Declarations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global rpqw_est rm2pqw_est rm1pqw_est rm1ijkest rm2ijkest rijk_est estdatapt calccmRSS
ConCM compareperfectRSS UNKSS dtr rtd J2 mu RE tetherparameters rijkdatapt
rijkdataptperfect IJKtimes IJKtimesperfect calcbtmRSS calctopRSS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Source of Observation Data (Datasource)
% 1 = in-house data generator
% 2 = TETHERSIM data
% 3 = Real-World TiPS data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Datasource = 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variable which decides if you actually want
% to optimize the imperfect data or not
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
evalimperfectdata = 1;
% 1 = optimize imperfect data
% 0 = do not optimize imperfect data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
% Set certain parameters
% If the source of the data is the in-house generator
% or TETHERSIM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
if ((Datasource == 1) | (Datasource == 2))
   lowerpct = 100; % percentage of data points that are lower
   upperpct = 100; % lowerpct - upperpct is the percentage of upper mass observations
   setseed = 1; %sets whether a seed is used for the rand function
   % 0 = no random seed set
   % 1 = random seed set
```

```
    askforsite = 1;
    % 1 = ask for user input to decide which site data to evaluate
    % 0 = hardwire in site data to evaluate
    evalperfectdata = 0; % sets whether to evaluate perfect data
    % 1 = optimize perfect data
    % 0 = do not optimize perfect data
    if setseed == 1
        randn('seed',25); % seed used for randn
        rand('seed',2500); % seed used for rand
    end
    if askforsite == 0
        sitetoeval = 399;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  On off values which set certain parameters
%  needed in the program
%  1 = value is active
%  0 = value is inactive
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
UNKSS = 1; % Sets whether the parameters of the Satellite System are known in advance
(global variable)
% 0 = Satellite System Parameters known in advance
% 1 = Satellite System Parameters are unknown

EllEarth = 1; % Sets whether calculations are done with an elliptical Earth model instead of
spherical
% 0 = Calculations done using spherical Earth model
% 1 = Calculations done using elliptical Earth model

CalcEstOrbit = 0; % Sets whether the initial CM COE estimate is calculated from the observations
or an estimate is given
% 0 = An estimate is given in the program
% 1 = Calculate an initial estimated orbit using Herrick-Gibbs

ConCM = 0; % Sets whether to check if any of the data points are from the CM
%  This is useful for real-world TiPS data because some observations come from the dipole
antenna
% 0 = does not check if any observations are the CM
% 1 = does check if any observations are the CM

CalcGrndtrack = 1; % Sets whether to calculate and plot the ground track of the data
% 0 = do not calculate the ground track of the data
% 1 = calculate the ground track of the data

Plotsigmabars = 1; % Sets whether to plot 3 sigma bars for imperfect data observations for the
EC range plot
% 0 = do not plot 3 sigma bars
% 1 = plot 3 sigma bars for imperfect data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
%  This next parameter sets whether to do an optimization
```

```
% over the CM first, and then proceed to do optimizations
% from there.  It also still does the normal optimizations
% so there are a total of 7 optimizations done if
% this variable is turned on.
% This sometimes helps identify
% single-body satellites or helps with known TSS's to get
% even better estimates.
% WARNING!!!!!- This setting sometimes hurts unknown TSS
% identification, especially for shorter unknown tethers
% with single-mass only data, and shorter timeframes.
% Sometimes it is useful to try this optimization
% with this setting off and on and then
% compare the results by hand.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
CMoptimizationfirst = 0;
% 0 = do not do a set of CM optimizations
% 1 = do all 7 different types of optimizations

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% Global Conversion Factors and Constants
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
dtr = pi/180; % converts degrees to radians
rtd = 180/pi; % converts radians to degrees
J2 = 0.00108263; % Dimensionless J2 geopotential coefficient
mu = 3.986032e5; % Earth Gravitational Parameter (km^3/sec^2)
RE = 6378.165; %  Equatorial radius of the Earth (km)
Ae = RE;
if EllEarth == 1
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Earth elliptical model constant
   % obtained from
   % Fundamentals of Astrodynamics
   % by Bate et al.- p. 94
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   Ee = 0.08181; % Earth Eccentricity needed if using elliptical Earth model
   f = 1/298.30; % flattening of the earth quantity used to do ground track obtained from Escobal
else
   Ee = 0; % Earth eccentricity if using spherical Earth model
   f = 0; % flattening of the Earth if spherical Earth
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
% Actual Tether parameters-
% Always needed for UNKSS = 0
% Always needed for Datasource = 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
m1 = 1; %43.32; % lower body (mass 1) mass (kg)
m2 = 1; %10.18; % upper body (mass 2) mass (kg)
tethermass = 5.45; % mass of the tether (kg)
```

```
ro = 20; %4.023; % length of the tether (km)
% Calculate distance of both end masses from the CM
distancem1 = (m2*ro + tethermass*ro/2)/(m1 + m2 + tethermass); % m1 distance to CM
distancem2 = ro - distancem1; % m2 distance to CM
'tetherparameters = [m1, m2, tethermass, ro, distancem1, distancem2]'
tetherparameters = [m1, m2, tethermass, ro, distancem1, distancem2]
tetherparameterstemp = [tetherparameters];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%  Tether Parameters Guess needed for UNKSS = 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
m1guess = 1; %kg
m2guess = 1; %kg
tethermassguess = 0; %kg
roguess = 0; %km

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
% Set up the first estimate if not calculating the estimate
% from the observations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
if CalcEstOrbit == 0
   aguess = 7000; %km
   eguess = 0.0001;
   iguess = 45.3; %deg
   wguess = 25; %deg
   Capwguess = 190.2; %deg
   taguess = 20; %deg
   if UNKSS == 1
      'COEguess = [a, e, i, w, Capw, v, m1, m2, ro]'
      COEguess = [aguess, eguess, iguess, wguess, Capwguess, taguess, m1guess, m2guess,
roguess]
      COEguess2 = [aguess, eguess, iguess, wguess, Capwguess, taguess];
   else
      'COEguess = [a, e, i, w, Capw, v]'
      COEguess = [aguess, eguess, iguess, wguess, Capwguess, taguess]
      COEguess2 = COEguess;
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
% Set the options for the optimization and
% Set the lower and upper bounds for the optimization routine
% depending on whether the satellite system parameters are known
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
if UNKSS == 1
   % bounds are [a, e, i, w, Capw, ta, m1, m2, ro]
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
    % w, Capw, and ta bounds go below 0 deg and above 360 deg
    % because if they do not sometimes the program can get 'stuck'
    % at 0 or 360 degrees because it can't go any farther.
    % These values are adjusted after the optimization is complete
    % so they are between 0 and 360 degrees

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
    % if the m1 and m2 lower bounds are set to 0 this sometimes
    % causes divide by zero type issues

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
    lb = [6400; 0; 0; -360; -360; -360; 1e-010; 1e-010; 0];
    ub = [57440; 0.9; 90; 720; 720; 720; 1000; 1000; 1000];
    lb2 = [6400; 0; 0; -360; -360; -360];
    ub2 = [57440; 0.9; 90; 720; 720; 720];
else
    % bounds are [a, e, i, w, Capw, ta]
    lb = [6400; 0; 0; 0; 0; 0];
    lb2 = lb;
    ub = [57440; 0.9; 90; 720; 360; 720];
    ub2 = ub;
end
% options just sets the main fmincon options needed for the optimization
options=optimset('LargeScale', 'off', 'MaxFunEvals', 10000, 'MaxIter', 10000, 'display', 'off');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Obtain information about all possible tracking sites to include
%  Latitude, Longitude, altitude, and statistical tracking site errors
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[numbersensors, sensorlats, sensorlongs, sensoraltitudes, sensorlatsrad, sensorlongsrad,
sigmasrall, sigmaeall, sigmaeallrad, sigmaaall, sigmaaallrad, sensorid] = sensorinfo;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Obtain observation data based on what the data source is
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Datasource 1 is the in-house
%  data generator
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Datasource == 1
```

```
   % Generate IJK position vector data using in-house simluator

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
   %  Actual Orbit parameters

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
   asim = 7400; % km
   esim = 0.004;
   isim = 65.3; % deg
   wsim = 70.0; % deg
   Capwsim = 220.45; % deg
   tasim = 70.0; % deg
   'COEsim = [a, e, i, w, Capw, v]'
   COEsim = [asim, esim, isim, wsim, Capwsim, tasim]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
   %  Time parameters needed for Datasource = 1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
   timestep = 20; % time step (sec) used to generate additional COEs
   totalnumseconds = 420; % total seconds to generate additional COEs
   starttime = 97210113030.000; % actual start time at time t0
   % in the format YYDDDHHMMSS.SSS
   %  Generate an array of timesteps to generate COEs
   counter0 = 0; % place holder counter
   for counter1 = 0:timestep:totalnumseconds
      counter0 = counter0 + 1;
      COEtimes(counter0) = counter1; % array of COE times
   end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
   %  Propagate the initial COEs

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
   [AllCOEs, AllCOEsrad, endcount] = COEpropagator(COEsim, COEtimes);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
   %  Convert COEs into Earth-centered pqw position vectors

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
   [rpqw_out] = coe2rpqw(endcount, AllCOEsrad);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
   %  Convert from r in the pqw frame to the Earth-centered ijk frame
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
   [r_sat_ijk] = pqw2ijk(rpqw_out, AllCOEsrad, endcount);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
   %  Determine IJK position vectors of end masses relative to CM assuming
   %  nadir orientation of the TSS

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
   [rm1ijk, rm2ijk] = calc_r_endmasses(r_sat_ijk, tetherparameters, endcount);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
   %  Pick which end mass is the observed mass

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
   [rijkdataptperfect, obs] = pickrandom(r_sat_ijk, rm1ijk, rm2ijk, endcount, lowerpct, upperpct);
   IJKtimes = COEtimes;
end % end of acquiring IJK observation data from in-house simulator


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Datasource 2 is the TETHERSIM
%  data generator
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Datasource == 2
   filename = 'TiPS copy.out'; % name of TETHERSIM data file
   tethersimfiletype = 1; %specifies the style of TETHERSIM data file
   starttime = 97212230000.000; % actual start time at time t0
   % in the format YYDDDHHMMSS.SSS

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
   %  Obtaining TETHERSIM data and picking random data pts.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
   tethersimdata = importdata(filename);
   totalcount = size(tethersimdata,1);
   for datacounter = 1:totalcount

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
       % TETHERSIM data can be outputted in various formats
       % the two formats here are the two formats used to generate data
       % more formats are possible, but the important thing is to know
       % where the IJK vectors are located in the data

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

```
        if tethersimfiletype == 0
           tsimtime(datacounter) = tethersimdata(datacounter,1);
           r2(datacounter,:) = [tethersimdata(datacounter,2)/1000,
tethersimdata(datacounter,3)/1000, tethersimdata(datacounter,4)/1000];
           r1(datacounter,:) = [tethersimdata(datacounter,5)/1000,
tethersimdata(datacounter,6)/1000, tethersimdata(datacounter,7)/1000];
        else
           tsimtime(datacounter) = tethersimdata(datacounter,1);
           r2(datacounter,:) = [tethersimdata(datacounter,18)/1000,
tethersimdata(datacounter,19)/1000, tethersimdata(datacounter,20)/1000];
           r1(datacounter,:) = [tethersimdata(datacounter,21)/1000,
tethersimdata(datacounter,22)/1000, tethersimdata(datacounter,23)/1000];
        end
        if datacounter == 1
           time(datacounter) = 0;
        else
           time(datacounter) = tsimtime(datacounter) - tsimtime(1);
        end
        r1mag(datacounter) = sqrt(r1(datacounter,1)^2 + r1(datacounter,2)^2 + r1(datacounter,3)^2);
        r2mag(datacounter) = sqrt(r2(datacounter,1)^2 + r2(datacounter,2)^2 + r2(datacounter,3)^2);
    end
    if time(1) == time(2)
        % TETHERSIM files sometimes have two of the same readings at time 0
        for counter = 2:totalcount
           rm1ijk(counter-1,:) = [r1(counter,:), r1mag(counter)];
           rm2ijk(counter-1,:) = [r2(counter,:), r2mag(counter)];
           IJKtimes(counter-1) = time(counter);
        end
        totalcount = totalcount - 1;
    else
        for counter = 1:totalcount
           rm1ijk(counter,:) = [r1(counter,:), r1mag(counter)];
           rm2ijk(counter,:) = [r2(counter,:), r2mag(counter)];
           IJKtimes(counter) = time(counter);
        end
    end
    endcount = totalcount;
    [rijkdataptperfect, obs] = pickrandom2(rm1ijk, rm2ijk, endcount, lowerpct);
end % end of acquiring IJK observation data from TETHERSIM


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Datasource 3 is Real-world data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Datasource == 3

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
    %  Read in the datafile

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
    filename = 'sensor334.xls';
    filetype = 1; % filetype = 1 if file is .xls, or 0 if a text file
```

```
    [totalcount, sensor, actualtime, IJKtimes, azimuth, elevation, slantrange] =
acquiredata(filename, filetype);
    starttime = actualtime(1);
    endcount = totalcount;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
    %  Convert azimuth, elevation, slantrange to
    %  ro in the sez frame

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
    [rosezobs] = azelslant2sez(sensor, endcount, azimuth, elevation, slantrange);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%  Calculate Actual times using starttime and IJKtime for
%  Datasource = 1 or 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
if ((Datasource == 1) | (Datasource == 2))
    [actualtime] = Calculateactualtime(starttime, IJKtimes, endcount);
    IJKtimesperfect = IJKtimes;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%  Calculate Greenwich Sidereal Times for all of the
%  corresponding actual times
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
[GSTtimes] = CalculateGSTtime(actualtime, endcount);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%  Compute Earth-Centered Inertial IJK coordinates
%  for all of the trackings sites
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
[Rsensors, LSTall] = computesensorR(sensorid, Ee, GSTtimes, endcount, numbersensors,
sensorlatsrad, sensorlongsrad, sensoraltitudes);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Convert from ro SEZ coordinates to ro IJK coordinates for Datasource = 3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Datasource == 3
    [roijkobs] = sez2ijk(sensorid, rosezobs, endcount, numbersensors, sensorlatsrad, LSTall);
    [rijkdatapt] = roijk2rijk(roijkobs, endcount, numbersensors, Rsensors);
    endcount2 = endcount;
    GSTtimes2 = GSTtimes;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
```

```
% Calculate Azimuth, elevation, and slant range data
% for Datasource = 1 or 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
if ((Datasource == 2) | (Datasource == 1))
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Convert from rijkdataptperfect to
   % roijkdataptperfect
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   [roijkdataptperfect] = rijk2roijk(rijkdataptperfect, endcount, numbersensors, Rsensors);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Convert from roijkdataptperfect
   % to rosezdataptperfect
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   [rosezdataptperfect] = roijk2rosez(roijkdataptperfect, endcount, numbersensors, sensorlatsrad,
LSTall);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Convert from rosezdataptperfect
   % to az, el, slant readings for all sites
   % which can actually see the object
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   [allsensordataperfect, trackingsite, trackercount, totalcount] = rosez2azelslant(IJKtimes,
actualtime, rosezdataptperfect, endcount, numbersensors);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
   % ask for site to optimize data for if askforsite = 1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
   if askforsite == 1
      [sitetoeval] = picksite(trackingsite, trackercount);
   end
   [GSTtimes2, R1sensor, LSTsensor, sensor, endcount2, azimuth, elevation, slantrange,
actualtime, IJKtimes, tracknumber] = getsitedata(GSTtimes, Rsensors, LSTall, sitetoeval,
totalcount, allsensordataperfect);
   for counter = 1:numbersensors
      if sensor == sensorid(counter)
         sensoridnum = counter;
      end
   end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
   % add realistic tracking site errors to the data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
   [badaz, badel, badslant] = messupdata(endcount2, azimuth, elevation, slantrange,
sensoridnum, sigmasrall, sigmaaall, sigmaeall);
   for counter = 1:endcount
      observed = obs(tracknumber);
   end
   [rosezobs] = azelslant2sez(sensor, endcount2, badaz, badel, badslant);
```

```
    [roijkobs] = sez2ijk2(sensorid, sensoridnum, rosezobs, endcount2, numbersensors,
sensorlatsrad, LSTsensor);
    [rijkdatapt] = roijk2rijk2(roijkobs, endcount2, R1sensor);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Obtain sigma information to plot on the position magnitude
%  plot if desired
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if ((Datasource == 2) | (Datasource == 1))
    [totalsigma] = findtotalsigma(sensoraltitudes, endcount2, azimuth, elevation, slantrange,
sensoridnum, sigmasrall, sigmaaall, sigmaeallrad);
else
    [totalsigma] = findrealworldsigma(sensoraltitudes, sigmasrall, sigmaeallrad, sensor, endcount,
azimuth, elevation, slantrange, sensorid, numbersensors);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Calculate the first estimated orbit from the observations
%  if desired
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if CalcEstOrbit == 1
    if ((Datasource == 2) | (Datasource == 1))
        [COEguessperfect] = herrickgibbs(rijkdataptperfect, IJKtimesperfect);
        if UNKSS == 1
            COEguessperfect2 = COEguessperfect;
            COEguessperfect = [COEguessperfect, m1guess, m2guess, roguess]
        else
            COEguessperfect2 = COEguessperfect
        end
    end
    [COEguessnotperfect] = herrickgibbs(rijkdatapt, IJKtimes);
    if UNKSS == 1
        COEguessnotperfect2 = COEguessnotperfect;
        COEguessnotperfect = [COEguessnotperfect, m1guess, m2guess, roguess]
    else
        COEguessnotperfect2 = COEguessnotperfect
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% set some initial parameters needed for the optimization
% these values are needed as is, so they should not be altered
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
calcbtmRSS = 0;
calctopRSS = 0;
calccmRSS = 0;
compareperfectRSS = 0;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Optimize perfect data if desired
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if ((Datasource == 2) | (Datasource == 1))
   if evalperfectdata == 1
      if CalcEstOrbit == 1
         COEguess = COEguessperfect;
         COEguess2 = COEguessperfect2;
      end
      compareperfectRSS = 1;
      [RSS1stperfectdata] = optimizedata(COEguess)
      if CMoptimizationfirst == 1
         if UNKSS == 1
            tetherparameters = [1, 0, 0, 0, 0, 0];
            tempUNKSS = 1;
         else
            tempUNKSS = 0;
         end
         UNKSS = 0;
         calccmRSS = 1;
         [COEfinperfectcm, RSSfinperfectcm, Exitflag, output] = fmincon(@optimizedata,
COEguess2, [],[],[],[],lb2,ub2, [], options);
         [RSSfinperfectcm] = optimizedata(COEfinperfectcm);
         cmobsperfect = estdatapt;
         cmm1ijkestperfect = rm1ijkest;
         cmm2ijkestperfect = rm2ijkest;
         cmcmijkestperfect = rijk_est;
         cmm1pqwestperfect = rm1pqw_est;
         cmm2pqwestperfect = rm2pqw_est;
         cmcmpqwestperfect = rpqw_est;
         if tempUNKSS == 1
            tetherparameters = tetherparameterstemp;
            UNKSS = 1;
            COEfinperfectcm2 = [COEfinperfectcm, m1guess, m2guess, roguess];
         else
            COEfinperfectcm2 = COEfinperfectcm;
         end
         calccmRSS = 0;
         [COEfinperfectcmf, RSSfinperfectcm2, Exitflag, output] = fmincon(@optimizedata,
COEfinperfectcm2, [],[],[],[],lb,ub, [], options);
         [RSSfinperfectcm2] = optimizedata(COEfinperfectcmf);
         cmobsperfect2 = estdatapt;
         cmm1ijkestperfect2 = rm1ijkest;
         cmm2ijkestperfect2 = rm2ijkest;
         cmcmijkestperfect2 = rijk_est;
         cmm1pqwestperfect2 = rm1pqw_est;
         cmm2pqwestperfect2 = rm2pqw_est;
         cmcmpqwestperfect2 = rpqw_est;
         calcbtmRSS = 1;
         [COEfinperfectcmm1, RSSfinperfectcmm1, Exitflag, output] = fmincon(@optimizedata,
COEfinperfectcm2, [],[],[],[],lb,ub, [], options);
         [RSSfinperfectcmm1] = optimizedata(COEfinperfectcmm1);
         cmobsperfectm1 = estdatapt;
         cmm1ijkestperfectm1 = rm1ijkest;
```

```
        cmm2ijkestperfectm1 = rm2ijkest;
        cmcmijkestperfectm1 = rijk_est;
        cmm1pqwestperfectm1 = rm1pqw_est;
        cmm2pqwestperfectm1 = rm2pqw_est;
        cmcmpqwestperfectm1 = rpqw_est;
        calcbtmRSS = 0;
        calctopRSS = 1;
        [COEfinperfectcmm2, RSSfinperfectcmm2, Exitflag, output] = fmincon(@optimizedata,
COEfinperfectcm2, [],[],[],[],lb,ub, [], options);
        [RSSfinperfectcmm2] = optimizedata(COEfinperfectcmm2);
        cmobsperfectm2 = estdatapt;
        cmm1ijkestperfectm2 = rm1ijkest;
        cmm2ijkestperfectm2 = rm2ijkest;
        cmcmijkestperfectm2 = rijk_est;
        cmm1pqwestperfectm2 = rm1pqw_est;
        cmm2pqwestperfectm2 = rm2pqw_est;
        cmcmpqwestperfectm2 = rpqw_est;
        calctopRSS = 0;
        if ((RSSfinperfectcm < RSSfinperfectcm2) & (RSSfinperfectcm < RSSfinperfectcmm1) &
(RSSfinperfectcm < RSSfinperfectcmm2))
            bestcmfirstRSSfinperfect = RSSfinperfectcm;
            bestcmfirstCOEfinperfect = COEfinperfectcm;
            bestcmfirstobsperfect = cmobsperfect;
            bestcmfirstm1ijkperfect = cmm1ijkestperfect;
            bestcmfirstm2ijkperfect = cmm2ijkestperfect;
            bestcmfirstcmijkperfect = cmcmijkestperfect;
            bestcmfirstm1pqwperfect = cmm1pqwestperfect;
            bestcmfirstm2pqwperfect = cmm2pqwestperfect;
            bestcmfirstcmpqwperfect = cmcmpqwestperfect;
            cmbestoptimization = 1;
        else
            if ((RSSfinperfectcm2 < RSSfinperfectcmm1) & (RSSfinperfectcm2 <
RSSfinperfectcmm2))
                bestcmfirstRSSfinperfect = RSSfinperfectcm2;
                bestcmfirstCOEfinperfect = COEfinperfectcmf;
                bestcmfirstobsperfect = cmobsperfect2;
                bestcmfirstm1ijkperfect = cmm1ijkestperfect2;
                bestcmfirstm2ijkperfect = cmm2ijkestperfect2;
                bestcmfirstcmijkperfect = cmcmijkestperfect2;
                bestcmfirstm1pqwperfect = cmm1pqwestperfect2;
                bestcmfirstm2pqwperfect = cmm2pqwestperfect2;
                bestcmfirstcmpqwperfect = cmcmpqwestperfect2;
                cmbestoptimization = 2;
            else
                if (RSSfinperfectcmm1 < RSSfinperfectcmm2)
                    bestcmfirstRSSfinperfect = RSSfinperfectcmm1;
                    bestcmfirstCOEfinperfect = COEfinperfectcmm1;
                    bestcmfirstobsperfect = cmobsperfectm1;
                    bestcmfirstm1ijkperfect = cmm1ijkestperfectm1;
                    bestcmfirstm2ijkperfect = cmm2ijkestperfectm1;
                    bestcmfirstcmijkperfect = cmcmijkestperfectm1;
                    bestcmfirstm1pqwperfect = cmm1pqwestperfectm1;
                    bestcmfirstm2pqwperfect = cmm2pqwestperfectm1;
                    bestcmfirstcmpqwperfect = cmcmpqwestperfectm1;
```

```
                cmbestoptimization = 3;
            else
                bestcmfirstRSSfinperfect = RSSfinperfectcmm2;
                bestcmfirstCOEfinperfect = COEfinperfectcmm2;
                bestcmfirstobsperfect = cmobsperfectm2;
                bestcmfirstm1ijkperfect = cmm1ijkestperfectm2;
                bestcmfirstm2ijkperfect = cmm2ijkestperfectm2;
                bestcmfirstcmijkperfect = cmcmijkestperfectm2;
                bestcmfirstm1pqwperfect = cmm1pqwestperfectm2;
                bestcmfirstm2pqwperfect = cmm2pqwestperfectm2;
                bestcmfirstcmpqwperfect = cmcmpqwestperfectm2;
                cmbestoptimization = 4;
            end
        end
    end
end
calcbtmRSS = 1;
[COEfinperfectbtm, RSSfinperfectbtm, Exitflag, output] = fmincon(@optimizedata,
COEguess, [],[],[],[],lb,ub, [], options);
[RSSfinperfectbtm] = optimizedata(COEfinperfectbtm);
btmobsperfect = estdatapt;
btmm1ijkestperfect = rm1ijkest;
btmm2ijkestperfect = rm2ijkest;
btmcmijkestperfect = rijk_est;
btmm1pqwestperfect = rm1pqw_est;
btmm2pqwestperfect = rm2pqw_est;
btmcmpqwestperfect = rpqw_est;
calcbtmRSS = 0;
calctopRSS = 1;
[COEfinperfecttop, RSSfinperfecttop, Exitflag, output] = fmincon(@optimizedata, COEguess,
[],[],[],[],lb,ub, [], options);
[RSSfinperfecttop] = optimizedata(COEfinperfecttop);
topobsperfect = estdatapt;
topm1ijkestperfect = rm1ijkest;
topm2ijkestperfect = rm2ijkest;
topcmijkestperfect = rijk_est;
topm1pqwestperfect = rm1pqw_est;
topm2pqwestperfect = rm2pqw_est;
topcmpqwestperfect = rpqw_est;
calctopRSS = 0;
[COEfinperfect, RSSfinperfect, Exitflag, output] = fmincon(@optimizedata, COEguess,
[],[],[],[],lb,ub, [], options);
[RSSfinperfect] = optimizedata(COEfinperfect);
genobsperfect = estdatapt;
genm1ijkestperfect = rm1ijkest;
genm2ijkestperfect = rm2ijkest;
gencmijkestperfect = rijk_est;
genm1pqwestperfect = rm1pqw_est;
genm2pqwestperfect = rm2pqw_est;
gencmpqwestperfect = rpqw_est;
if UNKSS == 1
    'bestCOEperfect = [a, e, i, w, Capw, v, m1, m2, ro]'
else
    'bestCOEperfect = [a, e, i, w, Capw, v]'
```

```
end
if ((RSSfinperfect < RSSfinperfectbtm) & (RSSfinperfect < RSSfinperfecttop))
   bestoptimization = 3;
   bestRSSfinperfect = RSSfinperfect;
   bestCOEfinperfect = COEfinperfect;
   bestobsperfect = genobsperfect;
   bestm1ijkperfect = genm1ijkestperfect;
   bestm2ijkperfect = genm2ijkestperfect;
   bestcmijkperfect = gencmijkestperfect;
   bestm1pqwperfect = genm1pqwestperfect;
   bestm2pqwperfect = genm2pqwestperfect;
   bestcmpqwperfect = gencmpqwestperfect;
else
   if (RSSfinperfecttop < RSSfinperfectbtm)
      bestoptimization = 2;
      bestRSSfinperfect = RSSfinperfecttop;
      bestCOEfinperfect = COEfinperfecttop;
      bestobsperfect = topobsperfect;
      bestm1ijkperfect = topm1ijkestperfect;
      bestm2ijkperfect = topm2ijkestperfect;
      bestcmijkperfect = topcmijkestperfect;
      bestm1pqwperfect = topm1pqwestperfect;
      bestm2pqwperfect = topm2pqwestperfect;
      bestcmpqwperfect = topcmpqwestperfect;
   else
      bestoptimization = 1;
      bestRSSfinperfect = RSSfinperfectbtm;
      bestCOEfinperfect = COEfinperfectbtm;
      bestobsperfect = btmobsperfect;
      bestm1ijkperfect = btmm1ijkestperfect;
      bestm2ijkperfect = btmm2ijkestperfect;
      bestcmijkperfect = btmcmijkestperfect;
      bestm1pqwperfect = btmm1pqwestperfect;
      bestm2pqwperfect = btmm2pqwestperfect;
      bestcmpqwperfect = btmcmpqwestperfect;
   end
end
if CMoptimizationfirst == 1
   if (bestcmfirstRSSfinperfect < bestRSSfinperfect)
      if cmbestoptimization == 1
         'best perfect solution is cm optimization only'
      else
         if cmbestoptimization == 2
            'best perfect solution is cm optimization then general optimization'
         else
            if cmbestoptimization == 3
               'best perfect solution is cm optimization then bottom mass optimization'
            else
               'best perfect solution is cm optimization then top mass optimization'
            end
         end
      end
      bestRSSfinperfect = bestcmfirstRSSfinperfect;
      bestCOEfinperfect = bestcmfirstCOEfinperfect;
```

```
            bestobsperfect = bestcmfirstobsperfect;
            bestm1ijkperfect = bestcmfirstm1ijkperfect;
            bestm2ijkperfect = bestcmfirstm2ijkperfect;
            bestcmijkperfect = bestcmfirstcmijkperfect;
            bestm1pqwperfect = bestcmfirstm1pqwperfect;
            bestm2pqwperfect = bestcmfirstm2pqwperfect;
            bestcmpqwperfect = bestcmfirstcmpqwperfect;
         else
            if bestoptimization == 1
               'best perfect solution is bottom case optimization'
            else
               if bestoptimization == 2
                  'best perfect solution is top case optimization'
               else
                  'best perfect solution is general case optimization'
               end
            end
         end
      else
         if bestoptimization == 1
            'best perfect solution is bottom case optimization'
         else
            if bestoptimization == 2
               'best perfect solution is top case optimization'
            else
               'best perfect solution is general case optimization'
            end
         end
      end
      if UNKSS == 1
         m1estimateperfect =
bestCOEfinperfect(8)*bestCOEfinperfect(9)/(bestCOEfinperfect(7)+bestCOEfinperfect(8))
         m2estimateperfect = bestCOEfinperfect(9)-m1estimateperfect
      end
      [bestCOEfinperfect(4)] = adjustvalue(bestCOEfinperfect(4));
      [bestCOEfinperfect(5)] = adjustvalue(bestCOEfinperfect(5));
      [bestCOEfinperfect(6)] = adjustvalue(bestCOEfinperfect(6));
      bestCOEfinperfect
      bestRSSfinperfect
      bestobsperfect
   end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Optimize imperfect data if desired
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if evalimperfectdata == 1
   if CalcEstOrbit == 1
      COEguess = COEguessnotperfect;
      COEguess2 = COEguessnotperfect2;
   end
   compareperfectRSS = 0;
   [RSS1stdata] = optimizedata(COEguess)
```

```
    if CMoptimizationfirst == 1
        if UNKSS == 1
            tetherparameters = [1, 0, 0, 0, 0, 0];
            tempUNKSS = 1;
        else
            tempUNKSS = 0;
        end
        UNKSS = 0;
        calccmRSS = 1;
        [COEfincm, RSSfincm, Exitflag, output] = fmincon(@optimizedata, COEguess2,
[],[],[],[],lb2,ub2, [], options);
        [RSSfincm] = optimizedata(COEfincm);
        cmobs = estdatapt;
        cmm1ijkest = rm1ijkest;
        cmm2ijkest = rm2ijkest;
        cmcmijkest = rijk_est;
        cmm1pqwest = rm1pqw_est;
        cmm2pqwest = rm2pqw_est;
        cmcmpqwest = rpqw_est;
        if tempUNKSS == 1
            tetherparameters = tetherparameterstemp;
            UNKSS = 1;
            COEfincm2 = [COEfincm, m1guess, m2guess, roguess];
        else
            COEfincm2 = COEfincm;
        end
        calccmRSS = 0;
        [COEfincmf, RSSfincm2, Exitflag, output] = fmincon(@optimizedata, COEfincm2,
[],[],[],[],lb,ub, [], options);
        [RSSfincm2] = optimizedata(COEfincmf);
        cmobs2 = estdatapt;
        cmm1ijkest2 = rm1ijkest;
        cmm2ijkest2 = rm2ijkest;
        cmcmijkest2 = rijk_est;
        cmm1pqwest2 = rm1pqw_est;
        cmm2pqwest2 = rm2pqw_est;
        cmcmpqwest2 = rpqw_est;
        calcbtmRSS = 1;
        [COEfincmm1, RSSfincmm1, Exitflag, output] = fmincon(@optimizedata, COEfincm2,
[],[],[],[],lb,ub, [], options);
        [RSSfincmm1] = optimizedata(COEfincmm1);
        cmobsm1 = estdatapt;
        cmm1ijkestm1 = rm1ijkest;
        cmm2ijkestm1 = rm2ijkest;
        cmcmijkestm1 = rijk_est;
        cmm1pqwestm1 = rm1pqw_est;
        cmm2pqwestm1 = rm2pqw_est;
        cmcmpqwestm1 = rpqw_est;
        calcbtmRSS = 0;
        calctopRSS = 1;
        [COEfincmm2, RSSfincmm2, Exitflag, output] = fmincon(@optimizedata, COEfincm2,
[],[],[],[],lb,ub, [], options);
        [RSSfincmm2] = optimizedata(COEfincmm2);
        cmobsm2 = estdatapt;
```

```
cmm1ijkestm2 = rm1ijkest;
cmm2ijkestm2 = rm2ijkest;
cmcmijkestm2 = rijk_est;
cmm1pqwestm2 = rm1pqw_est;
cmm2pqwestm2 = rm2pqw_est;
cmcmpqwestm2 = rpqw_est;
calctopRSS = 0;
if ((RSSfincm < RSSfincm2) & (RSSfincm < RSSfincmm1) & (RSSfincm < RSSfincmm2))
   bestcmfirstRSSfin = RSSfincm;
   bestcmfirstCOEfin = COEfincm;
   bestcmfirstobs = cmobs;
   bestcmfirstm1ijk = cmm1ijkest;
   bestcmfirstm2ijk = cmm2ijkest;
   bestcmfirstcmijk = cmcmijkest;
   bestcmfirstm1pqw = cmm1pqwest;
   bestcmfirstm2pqw = cmm2pqwest;
   bestcmfirstcmpqw = cmcmpqwest;
   cmbestoptimization = 1;
else
   if ((RSSfincm2 < RSSfincmm1) & (RSSfincm2 < RSSfincmm2))
      bestcmfirstRSSfin = RSSfincm2;
      bestcmfirstCOEfin = COEfincmf;
      bestcmfirstobs = cmobs2;
      bestcmfirstm1ijk = cmm1ijkest2;
      bestcmfirstm2ijk = cmm2ijkest2;
      bestcmfirstcmijk = cmcmijkest2;
      bestcmfirstm1pqw = cmm1pqwest2;
      bestcmfirstm2pqw = cmm2pqwest2;
      bestcmfirstcmpqw = cmcmpqwest2;
      cmbestoptimization = 2;
   else
      if (RSSfincmm1 < RSSfincmm2)
         bestcmfirstRSSfin = RSSfincmm1;
         bestcmfirstCOEfin = COEfincmm1;
         bestcmfirstobs = cmobsm1;
         bestcmfirstm1ijk = cmm1ijkestm1;
         bestcmfirstm2ijk = cmm2ijkestm1;
         bestcmfirstcmijk = cmcmijkestm1;
         bestcmfirstm1pqw = cmm1pqwestm1;
         bestcmfirstm2pqw = cmm2pqwestm1;
         bestcmfirstcmpqw = cmcmpqwestm1;
         cmbestoptimization = 3;
      else
         bestcmfirstRSSfin = RSSfincmm2;
         bestcmfirstCOEfin = COEfincmm2;
         bestcmfirstobs = cmobsm2;
         bestcmfirstm1ijk = cmm1ijkestm2;
         bestcmfirstm2ijk = cmm2ijkestm2;
         bestcmfirstcmijk = cmcmijkestm2;
         bestcmfirstm1pqw = cmm1pqwestm2;
         bestcmfirstm2pqw = cmm2pqwestm2;
         bestcmfirstcmpqw = cmcmpqwestm2;
         cmbestoptimization = 4;
      end
```

```
            end
        end
    end
    calcbtmRSS = 1;
    [COEfinbtm, RSSfinbtm, Exitflag, output] = fmincon(@optimizedata, COEguess, [],[],[],[],lb,ub,
[], options);
    [RSSfinbtm] = optimizedata(COEfinbtm);
    btmobs = estdatapt;
    btmm1ijkest = rm1ijkest;
    btmm2ijkest = rm2ijkest;
    btmcmijkest = rijk_est;
    btmm1pqwest = rm1pqw_est;
    btmm2pqwest = rm2pqw_est;
    btmcmpqwest = rpqw_est;
    calcbtmRSS = 0;
    calctopRSS = 1;
    [COEfintop, RSSfintop, Exitflag, output] = fmincon(@optimizedata, COEguess, [],[],[],[],lb,ub, [],
options);
    [RSSfintop] = optimizedata(COEfintop);
    topobs = estdatapt;
    topm1ijkest = rm1ijkest;
    topm2ijkest = rm2ijkest;
    topcmijkest = rijk_est;
    topm1pqwest = rm1pqw_est;
    topm2pqwest = rm2pqw_est;
    topcmpqwest = rpqw_est;
    calctopRSS = 0;
    [COEfin, RSSfin, Exitflag, output] = fmincon(@optimizedata, COEguess, [],[],[],[],lb,ub, [],
options);
    [RSSfin] = optimizedata(COEfin);
    genobs = estdatapt;
    genm1ijkest = rm1ijkest;
    genm2ijkest = rm2ijkest;
    gencmijkest = rijk_est;
    genm1pqwest = rm1pqw_est;
    genm2pqwest = rm2pqw_est;
    gencmpqwest = rpqw_est;
    if UNKSS == 1
        'bestCOE = [a, e, i, w, Capw, v, m1, m2, ro]'
    else
        'bestCOE = [a, e, i, w, Capw, v]'
    end
    if ((RSSfin < RSSfinbtm) & (RSSfin < RSSfintop))
        bestoptimization = 3;
        bestRSSfin = RSSfin;
        bestCOEfin = COEfin;
        bestobs = genobs;
        bestm1ijk = genm1ijkest;
        bestm2ijk = genm2ijkest;
        bestcmijk = gencmijkest;
        bestm1pqw = genm1pqwest;
        bestm2pqw = genm2pqwest;
        bestcmpqw = gencmpqwest;
    else
```

```
    if (RSSfintop < RSSfinbtm)
        bestoptimization = 2;
        bestRSSfin = RSSfintop;
        bestCOEfin = COEfintop;
        bestobs = topobs;
        bestm1ijk = topm1ijkest;
        bestm2ijk = topm2ijkest;
        bestcmijk = topcmijkest;
        bestm1pqw = topm1pqwest;
        bestm2pqw = topm2pqwest;
        bestcmpqw = topcmpqwest;
    else
        bestoptimization = 1;
        bestRSSfin = RSSfinbtm;
        bestCOEfin = COEfinbtm;
        bestobs = btmobs;
        bestm1ijk = btmm1ijkest;
        bestm2ijk = btmm2ijkest;
        bestcmijk = btmcmijkest;
        bestm1pqw = btmm1pqwest;
        bestm2pqw = btmm2pqwest;
        bestcmpqw = btmcmpqwest;
    end
end
if CMoptimizationfirst == 1
    if (bestcmfirstRSSfin < bestRSSfin)
        if cmbestoptimization == 1
            'best  solution is cm optimization only'
        else
            if cmbestoptimization == 2
                'best  solution is cm optimization then general optimization'
            else
                if cmbestoptimization == 3
                    'best  solution is cm optimization then bottom mass optimization'
                else
                    'best  solution is cm optimization then top mass optimization'
                end
            end
        end
        bestRSSfin = bestcmfirstRSSfin;
        bestCOEfin = bestcmfirstCOEfin;
        bestobs = bestcmfirstobs;
        bestm1ijk = bestcmfirstm1ijk;
        bestm2ijk = bestcmfirstm2ijk;
        bestcmijk = bestcmfirstcmijk;
        bestm1pqw = bestcmfirstm1pqw;
        bestm2pqw = bestcmfirstm2pqw;
        bestcmpqw = bestcmfirstcmpqw;
    else
        if bestoptimization == 1
            'best  solution is bottom case optimization'
        else
            if bestoptimization == 2
                'best  solution is top case optimization'
```

```
            else
                'best  solution is general case optimization'
            end
         end
      end
   else
      if bestoptimization == 1
         'best  solution is bottom case optimization'
      else
         if bestoptimization == 2
            'best  solution is top case optimization'
         else
            'best  solution is general case optimization'
         end
      end
   end
   if UNKSS == 1
      m1estimate = bestCOEfin(8)*bestCOEfin(9)/(bestCOEfin(7)+bestCOEfin(8))
      m2estimate = bestCOEfin(9)-m1estimate
   end
   [bestCOEfin(4)] = adjustvalue(bestCOEfin(4));
         [bestCOEfin(5)] = adjustvalue(bestCOEfin(5));
         [bestCOEfin(6)] = adjustvalue(bestCOEfin(6));
         bestCOEfin
         bestRSSfin
   bestobs
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%  Calculate the ground track of the observed data if desired
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
if CalcGrndtrack == 1
   [datalat, datalong] = calculategroundtrack(endcount2, GSTtimes2, rijkdatapt, Ae, f);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %  Plot the ground track
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   [sensorlongs, sensorlats] = plotsensors(numbersensors, sensorlongs, sensorlats);
   plot(datalong, datalat, 'b.')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%  Plot Earth-centered position vector magnitude; EC Range
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
if (evalimperfectdata == 1)
   figure(2), clf
else
   if ((Datasource == 2) | (Datasource == 1))
      if (evalperfectdata == 1)
         figure(2),clf
      end
```

```
    end
end
if ((Datasource == 2) | (Datasource == 1))
    if ((evalperfectdata == 1) & (evalimperfectdata == 1))
        plot(IJKtimesperfect, rijkdataptperfect(:,4), 'k+', IJKtimesperfect, bestm1ijkperfect(:,4), 'ro',
IJKtimesperfect, bestm2ijkperfect(:,4), 'bs', IJKtimes, rijkdatapt(:,4), 'kx', IJKtimes, bestm1ijk(:,4),
'r.', IJKtimes, bestm2ijk(:,4), 'b*'), hold on
        xlabel('time-seconds')
                    ylabel('EC Range-Km')
                    Legend('Perfect Observation', 'm1 Estimate w/ perfect data', 'm2 Estimate w/
perfect data', 'Observations', 'm1 Estimate', 'm2 Estimate',0)
                    legend boxoff
        if Plotsigmabars == 1
            errorbar(IJKtimes, rijkdatapt(:,4), 3*totalsigma, 'kx')
        end
    else
        if evalperfectdata == 1
            plot(IJKtimesperfect, rijkdataptperfect(:,4), 'k+', IJKtimesperfect, bestm1ijkperfect(:,4), 'ro',
IJKtimesperfect, bestm2ijkperfect(:,4), 'bs')
            xlabel('time-seconds')
                            ylabel('EC Range-Km')
            Legend('Perfect Observation', 'm1 Estimate w/ perfect data', 'm2 Estimate w/ perfect
data',0)
            legend boxoff
        else
            if evalimperfectdata == 1
                plot(IJKtimes, rijkdatapt(:,4), 'kx', IJKtimes, bestm1ijk(:,4), 'r.', IJKtimes, bestm2ijk(:,4),
'b*'), hold on
                xlabel('time-seconds')
                                        ylabel('EC Range-Km')
                                        Legend('Observations', 'm1 Estimate', 'm2 Estimate',0)
                                        legend boxoff
                if Plotsigmabars == 1
                    errorbar(IJKtimes, rijkdatapt(:,4), 3*totalsigma, 'kx')
                end
            end
        end
    end
else
         if evalimperfectdata == 1
        plotcm = 1; % variable if you want to plot the estimated CM location for Real-World TiPS
data
        % 0 = do not plot estimated CM location
        % 1 = plot estimated CM location
        if plotcm == 1
            plot(IJKtimes, rijkdatapt(:,4), 'kx', IJKtimes, bestm1ijk(:,4), 'r.', IJKtimes, bestm2ijk(:,4), 'b*',
IJKtimes, bestcmijk(:,4), 'ko'), hold on
            Legend('Observations', 'm1 Estimate', 'm2 Estimate', 'CM estimate',0)
                            legend boxoff
        else
            plot(IJKtimes, rijkdatapt(:,4), 'kx', IJKtimes, bestm1ijk(:,4), 'r.', IJKtimes, bestm2ijk(:,4),
'b*'), hold on
            Legend('Observations', 'm1 Estimate', 'm2 Estimate',0)
                            legend boxoff
```

```
        end
        if Plotsigmabars == 1
           errorbar(IJKtimes, rijkdatapt(:,4), 3*totalsigma, 'kx')
        end
    end
          xlabel('time-seconds');
          ylabel('EC Range-Kilometers');
end

% End of Main Program
```

```
function [RSS] = optimizedata(COEguess)

global rpqw_est rm2pqw_est rm1pqw_est rm1ijkest rm2ijkest rijk_est estdatapt calccmRSS
ConCM compareperfectRSS endcount2 UNKSS dtr rtd J2 mu RE tetherparameters rijkdatapt
rijkdataptperfect IJKtimes IJKtimesperfect calcbtmRSS calctopRSS

RSS = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extract data from COEguess depending on
% if the tether parameters are known or not
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if UNKSS == 1
   COEestimate = [COEguess(1), COEguess(2), COEguess(3), COEguess(4), COEguess(5),
COEguess(6)];
   m1estimate = COEguess(7);
   m2estimate = COEguess(8);
   roestimate = COEguess(9);
   distancem1estimate = (m2estimate*roestimate/(m1estimate+m2estimate));
   distancem2estimate = roestimate-distancem1estimate;
else
   COEestimate = COEguess;
   m1estimate = tetherparameters(1);
   m2estimate = tetherparameters(2);
   roestimate = tetherparameters(4);
   distancem1estimate = tetherparameters(5);
   distancem2estimate = tetherparameters(6);
end

[COEestimate(4)] = adjustvalue(COEestimate(4));
[COEestimate(5)] = adjustvalue(COEestimate(5));
[COEestimate(6)] = adjustvalue(COEestimate(6));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Store the estimated tether parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tetherparametersest = [m1estimate, m2estimate, 0, roestimate, distancem1estimate,
distancem2estimate];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
% Determine if optimizing perfect data or imperfect data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
if compareperfectRSS == 1
   rijkobs = rijkdataptperfect;
   COEestimatetimes = IJKtimesperfect;
else
   rijkobs = rijkdatapt;
   COEestimatetimes = IJKtimes;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%  Propagate the estimated COEs
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
[AllCOEest,AllCOEestrad, estendcount] = COEpropagator(COEestimate, COEestimatetimes);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%  Convert the estimated COEs into Earth-Centered
%  pqw position vectors
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
[rpqw_est] = coe2rpqw(estendcount, AllCOEestrad);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
% Convert the estimated pqw data into IJK data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
[rijk_est] = pqw2ijk(rpqw_est, AllCOEestrad, estendcount);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%  Calculate the estimated IJK coordinates
%  of the upper and lower masses
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
[rm1ijkest, rm2ijkest] = calc_r_endmasses(rijk_est, tetherparametersest, estendcount);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%  Convert the end mass estimated IJK coordinates
%  back into pqw frame coordinates
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
[rm1pqw_est] = ijk2pqw(estendcount, rm1ijkest, AllCOEestrad);
[rm2pqw_est] = ijk2pqw(estendcount, rm2ijkest, AllCOEestrad);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%  Convert the observation data from Earth-Centered
%  IJK coordinates to pqw coordinates using the
%  ESTIMATED orbit as the reference
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
[rpqwobs] = ijk2pqw(estendcount, rijkobs, AllCOEestrad);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%  Calculate the Residuals depending on the parameters
%  for calcbtmRSS, calctopRSS, calccmRSS, and ConCM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
for counter = 1:estendcount
    if calcbtmRSS == 1
        pestimate = rm1pqw_est(counter,1);
        qestimate = rm1pqw_est(counter,2);
        westimate = rm1pqw_est(counter,3);
        magest = rm1pqw_est(counter,4);
        pobs = rpqwobs(counter,1);
        qobs = rpqwobs(counter,2);
        wobs = rpqwobs(counter,3);
```

```
    magobs = rpqwobs(counter,4);
    residualp = pobs - pestimate;
    residualq = qobs - qestimate;
    residualw = wobs - westimate;
    residualmag = magobs - magest;
    estdatapt(counter) = 1;
else
    if calctopRSS == 1
        pestimate = rm2pqw_est(counter,1);
        qestimate = rm2pqw_est(counter,2);
        westimate = rm2pqw_est(counter,3);
        magest = rm2pqw_est(counter,4);
        pobs = rpqwobs(counter,1);
        qobs = rpqwobs(counter,2);
        wobs = rpqwobs(counter,3);
        magobs = rpqwobs(counter,4);
        residualp = pobs - pestimate;
        residualq = qobs - qestimate;
        residualw = wobs - westimate;
        residualmag = magobs - magest;
        estdatapt(counter) = 2;
    else
        if calccmRSS == 1
            pestimate = rpqw_est(counter,1);
            qestimate = rpqw_est(counter,2);
            westimate = rpqw_est(counter,3);
            magest = rpqw_est(counter,4);
            pobs = rpqwobs(counter,1);
            qobs = rpqwobs(counter,2);
            wobs = rpqwobs(counter,3);
            magobs = rpqwobs(counter,4);
            residualp = pobs - pestimate;
            residualq = qobs - qestimate;
            residualw = wobs - westimate;
            residualmag = magobs - magest;
            estdatapt(counter) = 3;
        else
            p1estimate = rm1pqw_est(counter,1);
            q1estimate = rm1pqw_est(counter,2);
            w1estimate = rm1pqw_est(counter,3);
            mag1est = rm1pqw_est(counter,4);
            p2estimate = rm2pqw_est(counter,1);
            q2estimate = rm2pqw_est(counter,2);
            w2estimate = rm2pqw_est(counter,3);
            mag2est = rm2pqw_est(counter,4);
            pobs = rpqwobs(counter,1);
            qobs = rpqwobs(counter,2);
            wobs = rpqwobs(counter,3);
            magobs = rpqwobs(counter,4);
            residualp1 = pobs - p1estimate;
            residualq1 = qobs - q1estimate;
            residualw1 = wobs - w1estimate;
            residualmag1 = magobs - mag1est;
            residualp2 = pobs - p2estimate;
```

```
            residualq2 = qobs - q2estimate;
            residualw2 = wobs - w2estimate;
            residualmag2 = magobs - mag2est;
            if ConCM == 1
               pcmestimate = rpqw_est(counter,1);
               qcmestimate = rpqw_est(counter,2);
               wcmestimate = rpqw_est(counter,3);
               magcmest = rpqw_est(counter,4);
               residualpcm = pobs - pcmestimate;
               residualqcm = qobs - qcmestimate;
               residualwcm = wobs - wcmestimate;
               residualmagcm = magobs - magcmest;
               if ((abs(residualmagcm) < abs(residualmag1)) & (abs(residualmagcm) <
abs(residualmag2)))
                  residualmag = residualmagcm;
                  estdatapt(counter) = 3;
               else
                  if (abs(residualmag2) < abs(residualmag1))
                     residualmag = residualmag2;
                     estdatapt(counter) = 2;
                  else
                     residualmag = residualmag1;
                     estdatapt(counter) = 1;
                  end
               end
               if ((abs(residualpcm) < abs(residualp1)) & (abs(residualpcm) < abs(residualp2)))
                  residualp = residualpcm;
               else
                  if (abs(residualp2) < abs(residualp1))
                     residualp = residualp2;
                  else
                     residualp = residualp1;
                  end
               end
               if ((abs(residualqcm) < abs(residualq1)) & (abs(residualqcm) < abs(residualq2)))
                  residualq = residualqcm;
               else
                  if (abs(residualq2) < abs(residualq1))
                     residualq = residualq2;
                  else
                     residualq = residualq1;
                  end
               end
               if ((abs(residualwcm) < abs(residualw1)) & (abs(residualwcm) < abs(residualw2)))
                  residualw = residualwcm;
               else
                  if (abs(residualw2) < abs(residualw1))
                     residualw = residualw2;
                  else
                     residualw = residualw1;
                  end
               end
            else
               if (abs(residualmag2) < abs(residualmag1))
```

```
                residualmag = residualmag2;
                estdatapt(counter) = 2;
            else
                residualmag = residualmag1;
                estdatapt(counter) = 1;
            end
            if (abs(residualp2) < abs(residualp1))
                residualp = residualp2;
            else
                residualp = residualp1;
            end
            if (abs(residualq2) < abs(residualq1))
                residualq = residualq2;
            else
                residualq = residualq1;
            end
            if (abs(residualw2) < abs(residualw1))
                residualw = residualw2;
            else
                residualw = residualw1;
            end
        end
    end
end
totalresidual = residualp^2 + residualq^2 + residualw^2 + 10*residualmag^2;
RSS = RSS + totalresidual;
end
RSS = sqrt(RSS);

% End of optimizedata function
```

Bibliography

Barnds, William J. and Coffey, Shannon L. "Tracking of the TiPS Tethered Satellite System," *Advances in the Astronautical Sciences*, Volume 103, Astrodynamics Part II: 1843-1853 (August 1999).

Bate, Roger R. et al. *Fundamentals of Astrodynamics*. New York: Dover Publications, Inc., 1971.

Beletsky, Vladimir V. and Levin, Evgenii M. "Dynamics of Space Tether Systems," *Advances in the Astronautical Sciences*, Volume 83 (1993).

Beyer, William H. *CRC Standard Mathematical Tables and Formulae* (29th Edition). Boston: CRC Press, 1991.

Cicci, D.A. et al. "A Filtering Method for the Identification of a Tethered Satellite," *The Journal of the Astronautical Sciences*, Volume 49, Number 2: 309-326 (April-June 2001).

Cochran, J.E. et al. "Evaluation of the Information Contained in the Motion of One Satellite of a Two-Satellite Tethered System," *The Journal of the Astronautical Sciences*, Volume 48, Number 4: 477-493 (October-December 2000).

Cochran, J.E. et al. "Modeling Tethered Satellite Systems for Detection and Orbit Determination," *Advances in the Astronautical Sciences*, Volume 103, Astrodynamics Part II: 1821-1841 (August 1999).

Escobal, Pedro Ramon. *Methods of Orbit Determination*. New York: John Wiley & Sons, Inc., 1965.

Lovell, T.A. et al. "Use of Tethered Satellite Estimation Methods in Identifying Re-Entering Objects," *AAS/AIAA Space Flight Mechanics Meeting Clearwater, Florida*. San Diego: AAS Publications Office (23-26 January 2000).

Purdy, William et al. "TiPS: Results of a Tethered Satellite Experiment," *Advances in the Astronautical Sciences*, Volume 97, Astrodynamics Part I: 3-23 (August 1997).

Wertz, James R. and Larson, Wiley J. *Space Mission Analysis and Design* (3rd Edition). El Segundo, California: Microcosm Press, 1999.

Wiesel, William E. *Modern Orbit Determination*. Independently Published by William Wiesel, Copyright: William Wiesel, 2003.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
March 2004	Master's Thesis	June 2003 – March 2004

4. TITLE AND SUBTITLE

DATA SORTING AND ORBIT DETERMINATION OF TETHERED SATELLITE SYSTEMS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Faulstich, Mark J., Captain, USAF

5d. PROJECT NUMBER
ENR #

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GSS/ENY/04-M03

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Dr. Robert A. Racca
HQ AFSPC/XPY
1150 Vandenberg St. Suite 1105
(719) 556-3714

10. SPONSOR/MONITOR'S ACRONYM(S)

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

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Tethered satellite system end masses do not obey the normal laws of motion developed for determining their orbits. In addition, tethered satellite systems cause unique problems for satellite tracking because there are potentially two or more objects which may be tracked.

This thesis provides insight into these issues by developing a method of sorting out observation data of tethered satellite systems into their appropriate end mass and providing an estimate on the center of mass orbit of the tethered satellite system. The method used to accomplish both of these tasks is optimization of an estimated simulated orbit. This orbit estimate is optimized to provide the minimum difference between the end mass position estimates and the observations obtained from one or more tracking sites. This methodology also helps provide a baseline for tracking tethered satellite systems more accurately in the future.

15. SUBJECT TERMS

Orbits, Artificial Satellites, Orbiting Satellites, Satellites(Artificial)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Steven G. Tragesser, AFIT/ENY
U	U	U	UU	120	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, ext 4286 (steven.tragesser@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18