

1995

# Weighted Least-Squares orbit estimation using GPS SPS Navigation Solution data

Christopher George Bryan  
*San Jose State University*

Follow this and additional works at: [https://scholarworks.sjsu.edu/etd\\_theses](https://scholarworks.sjsu.edu/etd_theses)

---

## Recommended Citation

Bryan, Christopher George, "Weighted Least-Squares orbit estimation using GPS SPS Navigation Solution data" (1995). *Master's Theses*. 1125.

DOI: <https://doi.org/10.31979/etd.pbc3-y3u5>

[https://scholarworks.sjsu.edu/etd\\_theses/1125](https://scholarworks.sjsu.edu/etd_theses/1125)

This Thesis is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Theses by an authorized administrator of SJSU ScholarWorks. For more information, please contact [scholarworks@sjsu.edu](mailto:scholarworks@sjsu.edu).

## **INFORMATION TO USERS**

**This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.**

**The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.**

**In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.**

**Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.**

**Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.**

# **UMI**

A Bell & Howell Information Company  
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA  
313/761-4700 800/521-0600



**WEIGHTED LEAST-SQUARES ORBIT ESTIMATION  
USING GPS SPS NAVIGATION SOLUTION DATA**

A Thesis

Presented to

The Faculty of the Department of Physics

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

of Master of Science

by

Christopher George Bryan

December 1995

**UMI Number: 1377216**

---

**UMI Microform 1377216  
Copyright 1996, by UMI Company. All rights reserved.**

**This microform edition is protected against unauthorized  
copying under Title 17, United States Code.**

---

**UMI**

**300 North Zeeb Road  
Ann Arbor, MI 48103**

© 1995

Christopher George Bryan

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT  
OF PHYSICS

A handwritten signature in cursive script, appearing to read "Alejandro Garcia", written over a horizontal line.

Dr. Alejandro Garcia

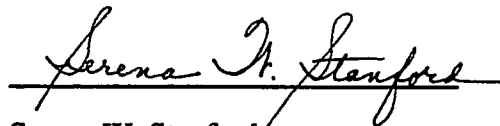
A handwritten signature in cursive script, appearing to read "Patrick Hamill", written over a horizontal line.

Dr. Patrick Hamill

A handwritten signature in cursive script, appearing to read "Allen Tucker", written over a horizontal line.

Dr. Allen Tucker

APPROVED FOR THE UNIVERSITY

A handwritten signature in cursive script, appearing to read "Serena W. Stanford", written over a horizontal line.

Serena W. Stanford

## ABSTRACT

### WEIGHTED LEAST-SQUARES ORBIT ESTIMATION USING GPS SPS NAVIGATION SOLUTION DATA

by Christopher George Bryan

We have implemented a Batch Weighted Least-Squares orbit estimation algorithm which processes Global Positioning System (GPS) Navigation Solution (position/velocity) measurement data. The purpose was to determine the viability of Navigation Solution data (as opposed to the raw GPS observables) for orbit estimation, as well as to determine the extent to which batch estimation techniques may be able to overcome the largest error source effecting most Standard Positioning Service (SPS) GPS receivers, Selective Availability (SA).

The algorithm was used to process simulated Navigation Solution data with SA errors. Other measurement errors were not considered. Results indicate that the SA contribution to the over-all orbit determination error budget is on the order of 6 meters RMS over a 12 hour estimation span.

The algorithm was also used to process actual Navigation Solution data from a NASA spacecraft (EUVE). Results indicate that the EUVE velocity data contains a significant time bias.



## Table of Contents:

List of Figures .....	vi
List of Tables .....	vii
List of Appendices .....	viii
<b>I. Introduction .....</b>	<b>1</b>
<b>II. Description of GPS and its Measurements.....</b>	<b>4</b>
A. GPS Overview .....	4
B. The GPS Navigation Solution.....	7
C. Characterization of Error Sources.....	10
<b>III. Theory of Weighted Least-Squares Orbit Determination....</b>	<b>14</b>
A. Optimal State Estimation Overview .....	14
B. Implementation of the Weighted Least-Squares Estimator.....	22
<b>IV. Numerical Integration of the Equations of Motion.....</b>	<b>27</b>
A. Cowell Formulation of the Equations of Motion .....	27
B. Fourth Order Runge-Kutta Numerical Integration .....	36
<b>V. Results and Discussion.....</b>	<b>39</b>
A. Simulated GPS Navigation Solution Data Processing .....	40
A.1 With Gaussian Errors.....	41
A.2 With Selective Availability Errors .....	45
B. Processing of EUVE GPS Navigation Solution Data.....	54
B.1 Description of the EUVE GPS Receiver .....	54
B.2 EUVE Orbit Fit Results .....	56
C. Conclusions.....	62
<b>VI. Future Work .....</b>	<b>64</b>
<b>VII. Acknowledgements .....</b>	<b>66</b>
<b>VIII. References.....</b>	<b>67</b>

## List of Figures

Figure 1:	The GPS Constellation.....	5
Figure 2:	Simultaneous Measurement of GPS Pseudo-ranges.....	7
Figure 3a:	Typical Horizontal/Vertical Position Errors Due To SA.....	12
Figure 3b:	Typical Velocity Errors Due To SA .....	12
Figure 4:	RIC Coordinates .....	26
Figure 5:	RSS of X, Y, Z Errors Due To Geopotential Truncation .....	29
Figure 6:	Ephemeris Differences ( $J_2$ vs. $J_4$ ).....	30
Figure 7:	Ephemeris Differences ( $J_4$ vs. $21 \times 21$ ) .....	30
Figure 8:	Atmospheric Density as a Function of Altitude.....	35
Figure 9:	Adaptive Stepsize for EUVE Orbit .....	38
Figure 10:	Final Iteration Gaussian Position Fit Data ( $\sigma = 35$ m) .....	42
Figure 11:	Final Propagated State vs. Truth (Gaussian Position Fit) .....	43
Figure 12:	Simulated SA Pseudo-range Errors (Zyla ARIMA $3 \times 2$ ).....	46
Figure 13:	Simulation Geometry.....	47
Figure 14:	Simulated Horizontal/Vertical SA Errors .....	48
Figure 15:	12 Hour Error Distribution of Simulated SA Data.....	49
Figure 16:	Final Iteration Data for Position Fit With SA.....	50
Figure 17:	Final Iteration Solution vs. Truth (Position Fit with SA) .....	51
Figure 18:	Comparison of Gaussian vs. SA Orbit Estimation Results.....	53
Figure 19:	True of Date ECI and ECEF Coordinates.....	55
Figure 20:	First Iteration EUVE RIC Position Residuals, 14 Sept. 1992 .....	56
Figure 21:	Final Iteration EUVE Position Residuals With / Without SA ..	57
Figure 22:	Final Iteration EUVE Position/Velocity Fit Data, No SA .....	59
Figure 23:	Effect of 0.14 Second Time Bias on Velocity Data .....	61

## List of Tables

Table 1:	Typical GPS Pseudo-Range Measurement Errors.....	10
Table 2:	EUVE Orbital Parameters .....	40
Table 3:	Summary Results of Gaussian Measurement Error Fits.....	44
Table 4:	SA Error Statistics For This Simulation.....	49
Table 5:	Summary Results of SA Measurement Error Fits.....	52

## List of Appendices:

<b>Appendix A: Program Listings</b> .....	<b>69</b>
A.1: EUVWLS_OD.....	70
A.2: J4gravPlusDrag.....	87
A.3: rk4.....	89
A.4: Zyla3x2.....	90
A.5: simGauss_hdr.....	91
A.6: simSA_hdr.....	93
A.7: Horiz_VertSA.....	97
A.8: VecSum_SA.....	98
<b>Appendix B: Graphical Results</b> .....	<b>100</b>

## I. Introduction

The estimation and prediction of a spacecraft's trajectory using sparse measurements which have been corrupted by various error sources, and imperfect physical models of the forces acting on the spacecraft, is sometimes called the "orbit determination problem". Although the fundamental mathematical and physical concepts required for modern spacecraft orbit determination have been well understood for many years, much progress has been made in recent years in the level of accuracy that can be achieved on a routine basis. Much of this improvement in accuracy is due to better models of the Earth's gravitational field, atmospheric density, and other perturbing influences. The accuracy of measurement data has also steadily increased (laser ranging data, for example); but the accuracy of the final orbit solution is often limited by the availability of sufficient quantities of the measurement data since it must generally be gathered in short arcs over fixed ground sites. Data sparseness becomes especially important in low Earth orbit where mismodeling of the geopotential and atmospheric drag forces is largest.

If, however, a spacecraft is equipped with an on-board Global Positioning System (GPS) receiver, tracking measurements are available on a nearly continuous basis. The data may either be used on-board in near-realtime or stored for later downlink for processing by ground software. The precision of GPS data can be excellent depending on the sophistication of the receiver and the subsequent processing of the data.

Most research into orbit estimation accuracy using GPS data has been focused on determining the ultimate accuracy limits that GPS can offer. The best example thus far is GPS data processing for the TOPEX/POSEIDON spacecraft (Yunck [18]). A principal mission of this spacecraft is the collection of radar altimetry data on the Earth's oceans, requiring very precise position determination of the spacecraft itself. NASA's Jet Propulsion Laboratory (JPL) has developed a number of novel orbit estimation techniques to achieve the level of orbit accuracy required by this mission. These techniques include reduced-dynamic sequential state estimation, processing of dual frequency pseudo-range and carrier phase data, and Differential GPS (DGPS) to remove the effects of Selective Availability (SA). DGPS and SA will be discussed in further detail in sections II and VI.

As more and more spacecraft begin to use on-board GPS receivers, the question arises: "What data processing technique is most appropriate for the majority of these missions?". It is unlikely that the same techniques used for TOPEX/POSEIDON are appropriate for most other spacecraft for a number of reasons including:

1. Most space missions do not require centimeter-level orbit accuracy.
2. Most spacecraft will have single frequency SPS (Standard Positioning Service) GPS receivers, not dual frequency.
3. DGPS orbit determination methods require access to a world-wide network of ground-based GPS receivers. Also, the amount of time required to collect and process the data may not be suitable for operational needs.

4. Some spacecraft store and downlink the derived position/velocity "Navigation Solutions" but not the raw measurements due to on-board storage limitations.

It is therefore appropriate to seek GPS data processing strategies tailored to the needs of the majority of spacecraft missions. This often means the use of single frequency Navigation Solution data with SA effects.

The research contained in this thesis suggests that Batch Weighted Least-Squares Estimation techniques are a viable alternative to sequential filtering, especially for standard ground-based orbit determination in the presence of Selective Availability effects. In addition, this thesis suggests that the use of Navigation Solution data can offer good results using less complex processing techniques than the above-mentioned methods.

We will present an overview of GPS and the fundamental concepts of orbit determination by classical Weighted Least-Squares (WLS) Differential Correction. For this research, an algorithm has been coded in MATLAB to optimally determine the orbit of a satellite from measurements of its position and velocity. The program can be used to generate its own simulated measurement data corrupted by either Gaussian or Selective Availability errors, which may then be input to the orbit determination algorithm. The algorithm is also used to process several spans of actual GPS Navigation Solution data from a NASA spacecraft, the Extreme Ultra-Violet Explorer (see Gold [1]). The results of both the simulated and actual orbit fits are discussed in section V.

## II. A Description of the Global Positioning System and its Measurements

### A. GPS Overview

Until the recent advent of the GPS era, traditional orbit determination techniques have principally employed tracking data obtained from line-of-sight observations of spacecraft, most commonly topocentric range, azimuth, elevation and range-rate data. More precise observations can be obtained by laser-ranging techniques (for spacecraft equipped with corner-cube reflectors). However, the main drawback of this data is its sparseness. That is, it can only be obtained over short arcs during line-of-sight tracking passes relative to fixed ground sites.

If a spacecraft is equipped with a GPS receiver, nearly continuous tracking observations are available. This data is in the form of pseudo-range and carrier phase measurements (see [5], [19] and Part B below). If these observations are taken simultaneously from four or more GPS satellites, a geometric determination of position and velocity can be obtained (the Navigation Solution) which can be considered a "derived measurement".

Figure 1 below shows the configuration of the 21-spacecraft GPS constellation.



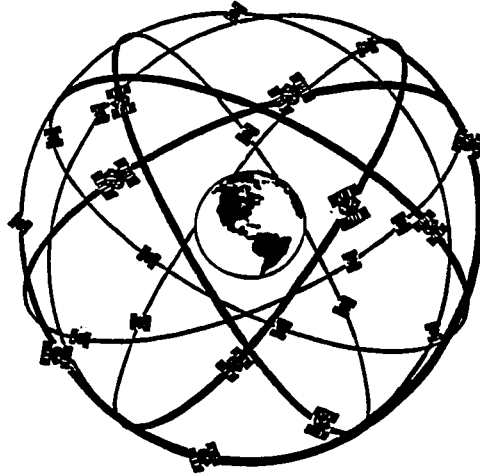


Figure 1: The GPS Constellation [19]

Although, in principle, the measurement of the position and velocity of a spacecraft is equivalent to determining its orbital elements, in practice this is not sufficient due to the following factors:

1. Individual position and velocity measurements may be corrupted by a number of error sources including the effects of "Selective Availability" (SA), the deliberate degradation by the Department of Defense (DOD) of the accuracy of the GPS signal.
2. Other elements of spacecraft state are often required for accurate propagation such as an estimate of the force due to atmospheric drag. An accurate estimate of this drag force is essential for the prediction of the future position of a spacecraft in low Earth orbit because atmospheric density is poorly modeled a priori. This is primarily due to the

unpredictable nature of solar activity which can often cause predictions of atmospheric density in the upper atmosphere to be in error by a factor of three or more.

For these reasons, a single measurement of GPS position and velocity will not in general be sufficient to predict the future position of a spacecraft.

Therefore, it is preferable to use the techniques of Optimal State Estimation to determine the orbital elements (see Smith [9]). In this technique, many GPS observations are processed simultaneously to obtain an "optimal" estimate of the spacecraft "state vector", which includes position, velocity, drag factor, and other parameters which may need to be corrected to improve orbit determination accuracy and ephemeris prediction.

In the case of GPS, the observations to be processed may be in either "raw" form (pseudo-range and carrier phase) or "derived" form (the position and velocity "Navigation Solution"), and the estimation of the state vector may take place in either "measurement space" or "solution space". This research will focus on the processing of the derived measurements of position and velocity, but the same concepts could apply to measurements of any type.

## B. The GPS Navigation Solution

Figure 2 below shows the concept of simultaneous measurements of four pseudo-ranges from four GPS spacecraft to geometrically determine the position of a spacecraft equipped with an on-board GPS receiver.

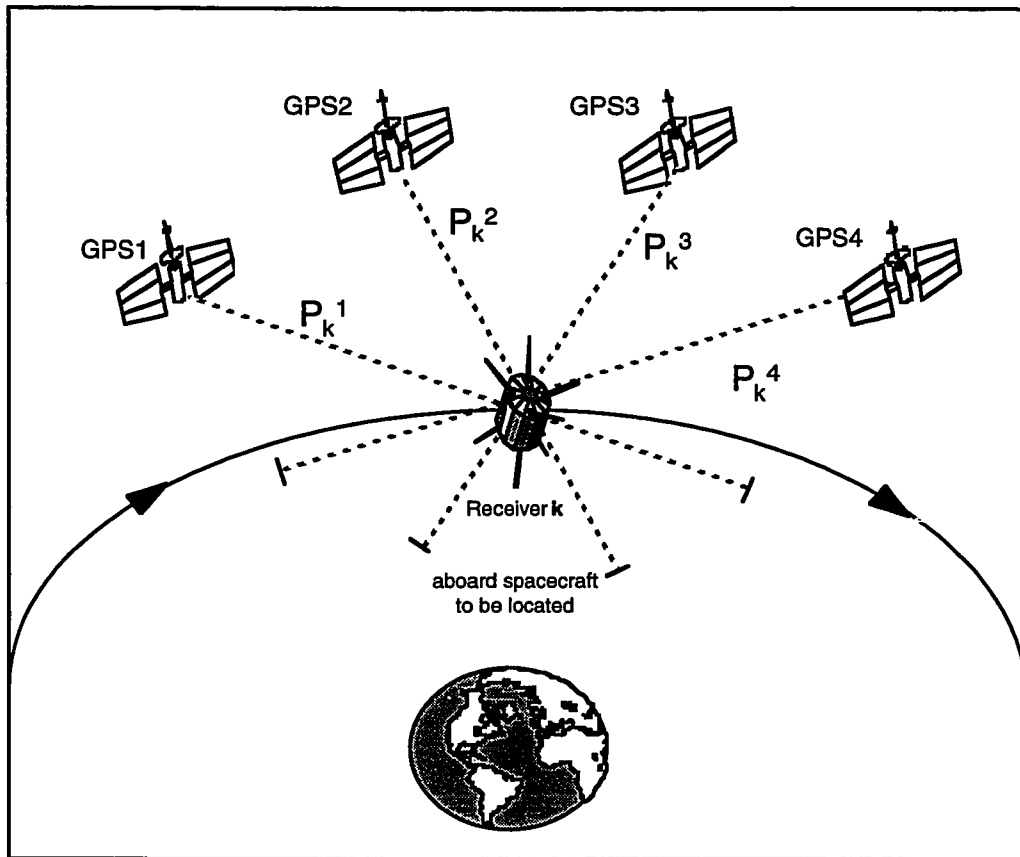


Figure 2. Simultaneous Measurement of GPS Pseudo-Ranges

The pseudo-range observable is defined as:

$$P_k^i = (t_k - t_i) c + \text{errors} \quad (1)$$

where the superscript refers to GPS satellite  $i$ , the subscript refers to GPS receiver  $k$ ,  $t_i$  is the time of transmission of the GPS signal as measured by the on-board atomic clock of GPS satellite  $i$ ,  $t_k$  is the time of reception of the GPS signal as measured by the clock of receiver  $k$ , and  $c$  is the speed of light. The "errors" generally include unmodeled measurement effects such as ionospheric delays, receiver noise, etc. Assuming that the GPS satellite clock time is known precisely relative to "GPS Time" (a continuous time scale whose fundamental unit is the SI second) and that receiver  $k$  has some initially unknown offset  $dt_k$  relative to GPS Time, equation (1) may be re-written as:

$$P_k^i = R_k^i + c dt_k + \text{errors}$$

where  $R_k^i$  is the true geometric range (distance) between GPS satellite  $i$  and the on-board receiver  $k$  (to be located). Clearly, pseudo-range is a biased range measurement. If  $dt_k$  is determined, the principal bias in the measurement may be removed and, in addition, the receiver may be synchronized to GPS Time.

If  $(x_k, y_k, z_k)$  are the unknown Cartesian components of receiver  $k$  position, and  $(x^i, y^i, z^i)$  are the known Cartesian components of GPS satellite  $i$  position, then given four measurements of pseudo-range  $P_k^i$  we have (neglecting the "errors"):

$$P_k^1 = \sqrt{(x^1 - x_k)^2 + (y^1 - y_k)^2 + (z^1 - z_k)^2} + c \cdot dt_k$$

$$P_k^2 = \sqrt{(x^2 - x_k)^2 + (y^2 - y_k)^2 + (z^2 - z_k)^2} + c \cdot dt_k$$

$$P_k^3 = \sqrt{(x^3 - x_k)^2 + (y^3 - y_k)^2 + (z^3 - z_k)^2} + c \cdot dt_k$$

$$P_k^4 = \sqrt{(x^4 - x_k)^2 + (y^4 - y_k)^2 + (z^4 - z_k)^2} + c \cdot dt_k$$

These are four simultaneous equations in the four unknowns  $x_k, y_k, z_k, dt_k$ . Thus (in theory), both the receiver position and the offset of the receiver clock from GPS Time may be determined. In this way, each position measurement may be associated with a very precise time tag since the time offset measurement will usually permit the synchronization of the receiver clock to within a few hundred nanoseconds of GPS Time.

In a similar fashion, four simultaneous measurements of GPS carrier phase will yield the Cartesian components of receiver velocity. That is, the receiver measures integrated carrier phase over some small time interval in order to determine the doppler shift, which is proportional to the line-of-sight relative velocity between the receiver and the GPS satellite (see May [11]). The geometric determination of the GPS receiver position and velocity from pseudo-range and carrier phase measurements is called the "Navigation Solution".

### C. Characterization of GPS Error Sources

The sources of error in the pseudo-range measurement have been estimated and are listed in Table 1 below:

Source	Range Error
GPS SV Clock Errors & Ephemeris Errors	6 m to 8 m
Ionospheric Delays (uncorrected)	10 m to ~100 m
Tropospheric Delays (modeled)	2 m
Multipath (dependent on receiver antenna configuration)	1 m to ~10 m
Receiver Noise / Resolution C/A Code	7 m
Receiver Noise / Resolution P Code	1 m
Selective Availability	~30 m ( $1\sigma$ )
Geometric Dilution of Precision (multiplier)	2 - 5

Table 1: Typical GPS Pseudo-Range Measurement Errors ([15], [16], [17], [20])

According to Conley [17], velocity errors (in the presence of SA) on the order of approximately 0.4 meters per second RMS are also to be expected, with occasional transients of up to 2 meters per second.

As can be seen from Table 1, errors due to poor modeling of ionospheric delays can be particularly significant (although less so at orbital altitudes). For this reason, the GPS signal is transmitted on two L-band frequencies simultaneously ("L1" and "L2"). Since ionospheric delays are proportional to

frequency, this delay may be estimated and removed if both L1 and L2 data are processed by the receiver. However, only specially equipped (and authorized) receivers have guaranteed access to the GPS signal on both frequencies [19].

There are two pseudo-random codes which are impressed on the GPS carrier to facilitate the measurement of pseudo-range: the C/A (Clear Acquisition) code and the P-code (Precise code). The L1 signal contains both the C/A and P-codes while L2 contains (currently) only the P-code. The P-code is transmitted at a higher bit-rate and so is inherently somewhat more precise, however the DOD may at any time encrypt the P-code, denying access to civilian users. Since the P-code is the only code on both frequencies, civilian users are also denied the capability to correct for ionospheric errors. In addition, the Clear Acquisition code is subject to the effects of Selective Availability (SA), as will now be discussed.

Unlike most other measurement errors, SA is neither the result of the measurement process itself nor the result of limitations of physical models (e.g. ionospheric models). Rather it is an additional error source which is deliberately placed on the GPS signal by the Department of Defense (DOD) to limit the accuracy of GPS data to unauthorized users. There can be two contributions to this (see [12], [16]):

1. Epsilon: A deliberate error in the GPS "Ephemeris Message" (which the receiver uses to calculate each GPS satellite position).
2. Clock Dither: A deliberate variation in the GPS carrier signal frequency.

The algorithm used by the DOD to create SA is classified, however the errors are bounded (by agreement) to approximately 150 meters spherical error, two sigma. Examples of typical vertical and horizontal position and velocity errors are shown in Figures 3a and 3b below from Conley [17].

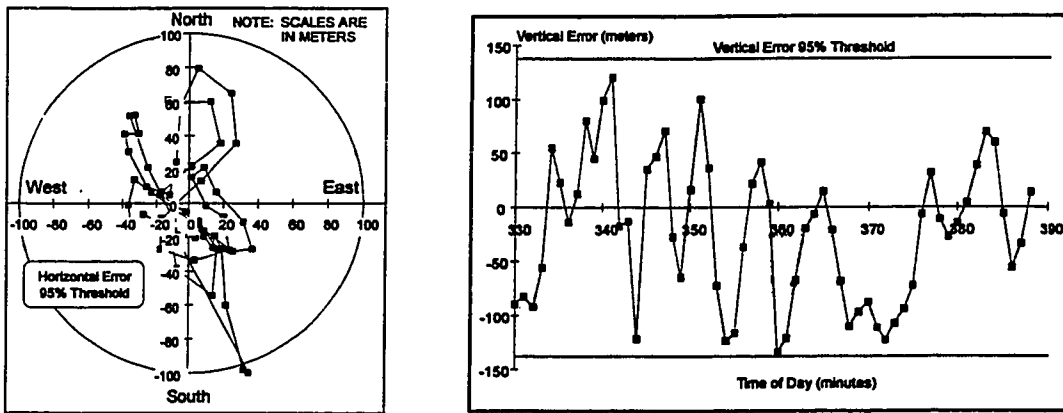


Figure 3a: Typical Horizontal / Vertical Position Errors Due To SA

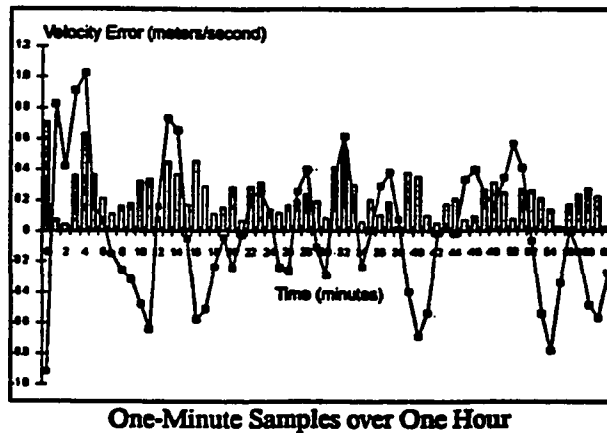


Figure 3b: Typical Velocity Errors Due To SA



SA effects are a significant problem for most terrestrial applications (although the techniques of Differential GPS can mitigate this in most cases - see section VI) but the effect on orbit determination accuracy is less severe. There are two basic reasons for this:

1. Although SA is effectively an unpredictable (though bounded) time-varying bias (colored noise) which is highly correlated over a time span of a few minutes, it may begin to take a more random appearance if sampled at a lower frequency than this "de-correlation time," and if sampled over a suitably long time span.
2. The true trajectory of a spacecraft is highly constrained by the laws of Physics and therefore not easily masked by SA effects.

Although details on the implementation of SA are somewhat limited, we know that when SA is turned on it can effect both the C/A and P-codes. Precise Positioning Service (PPS) GPS receivers will have unrestricted access to the P-code on both frequencies even when encrypted and will not be subject to the effects of SA. However, most GPS users (including spacecraft) will only have access to single frequency C/A code data with SA effects [16].

In section V we will explore in more detail the effect of SA on orbit determination accuracy.

### III. Theory of Weighted Least-Squares Orbit Determination

As we shall see, modern orbit determination techniques require the application of several different areas of Mathematics and Physics including Optimal State Estimation (which also includes Probability and Statistics), Astrodynamics (the application of Celestial Mechanics to spacecraft), and Numerical Analysis.

#### A. Optimal State Estimation Overview

The problem of orbit determination involves finding a set of observation and dynamical model parameters which "best" match a set of measurements. These parameters form the "state vector", which will be denoted  $\mathbf{x}$ . The parameters in the dynamical model include the initial conditions ( $x, y, z, v_x, v_y, v_z$  or the equivalent), as well as other model parameters to be estimated such as a drag factor (usually the Ballistic coefficient,  $B$ ). Observation model parameters to be estimated as part of the state vector are typically parameters such as the unknown biases in the measurements. The observation or measurement model can be expressed as (see Smith [9]):

$$\mathbf{m} = \mathbf{H}(\mathbf{x}) + \mathbf{v} \quad (2)$$

where  $\mathbf{x}$  is the spacecraft state vector,  $\mathbf{v}$  is measurement error (usually assumed to be random with Gaussian distribution),  $\mathbf{H}$  is some (generally non-linear) function which maps the state to the measurements, and  $\mathbf{m}$  is the

measurement itself. In other words, given a set of initial conditions  $\mathbf{x}$  at some epoch, a corresponding set of predicted tracking measurements  $\mathbf{m}$  at any time  $t$  may be generated by determining the position of the satellite at time  $t$ , geometrically (or otherwise) calculating the measurement and adding an appropriate amount of measurement error. These measurements may take many forms such as range, azimuth, elevation or range-rate relative to a particular ground site, or geocentric position and velocity as measured by an on-board GPS receiver.

The problem then is: given a set of measurements  $m_i$  at various times  $t_i$ , how do we determine the state vector  $\mathbf{x}$  which "best" matches the measurements? Unfortunately, due to the non-linearity of  $H$ , a unique closed-form solution to (2) cannot readily be obtained. We therefore seek to linearize (2) by assuming that an initial estimate or "guess" exists for the state vector (call it  $\mathbf{x}_0$ ) which is sufficiently close to the true state vector  $\mathbf{x}$  such that we may perform a Taylor Series expansion of (2) about  $\mathbf{x}_0$ . If  $\mathbf{x}$  consists of  $x$ ,  $y$ ,  $z$ ,  $v_x$ ,  $v_y$ ,  $v_z$ , and  $B$ , then the multi-variable Taylor expansion appears as:

$$\begin{aligned}
m(\vec{\mathbf{x}}) = & m(\vec{\mathbf{x}}_0) + \left. \frac{\partial H}{\partial x} \right|_{\vec{\mathbf{x}}_0} \cdot (x - x_0) + \left. \frac{\partial H}{\partial y} \right|_{\vec{\mathbf{x}}_0} \cdot (y - y_0) + \left. \frac{\partial H}{\partial z} \right|_{\vec{\mathbf{x}}_0} \cdot (z - z_0) \\
& + \left. \frac{\partial H}{\partial v_x} \right|_{\vec{\mathbf{x}}_0} \cdot (v_x - v_{x_0}) + \left. \frac{\partial H}{\partial v_y} \right|_{\vec{\mathbf{x}}_0} \cdot (v_y - v_{y_0}) + \left. \frac{\partial H}{\partial v_z} \right|_{\vec{\mathbf{x}}_0} \cdot (v_z - v_{z_0}) + \left. \frac{\partial H}{\partial B} \right|_{\vec{\mathbf{x}}_0} \cdot (B - B_0)
\end{aligned} \tag{3}$$

where all higher order terms beyond linear have been excluded. Each measurement  $m_i(\mathbf{x})$  at each  $t_i$  will result in an equation such as (3) above.

If the  $m_i(\mathbf{x})$  are assumed to be the actual measurements corresponding to the true trajectory, and  $\mathbf{m}(\mathbf{x})$  is the vector of these measurements, we may then

form the vector of measurement "residuals":

$$\Delta z = \mathbf{m}(\mathbf{x}) - \mathbf{m}(\mathbf{x}_0)$$

and we may re-write (3) in matrix form as:

$$\Delta z = \mathbf{A} \Delta \mathbf{x} \quad (4)$$

where  $\Delta \mathbf{x} = \mathbf{x} - \mathbf{x}_0$ , the vector of corrections to the initial state vector  $\mathbf{x}_0$ , and  $\mathbf{A}$  is the matrix of partial derivatives of the observations with respect to the state vector elements, sometimes referred to as the "normal matrix." In the case where the measurements are position and velocity,  $\mathbf{A}$  appears as:

$$\mathbf{A} = \begin{bmatrix} \frac{\partial}{\partial x} m_x |_{t_1} & \frac{\partial}{\partial y} m_x |_{t_1} & \frac{\partial}{\partial z} m_x |_{t_1} & \frac{\partial}{\partial v_x} m_x |_{t_1} & \frac{\partial}{\partial v_y} m_x |_{t_1} & \frac{\partial}{\partial v_z} m_x |_{t_1} & \frac{\partial}{\partial B} m_x |_{t_1} \\ \frac{\partial}{\partial x} m_x |_{t_2} & \frac{\partial}{\partial y} m_x |_{t_2} & \frac{\partial}{\partial z} m_x |_{t_2} & \frac{\partial}{\partial v_x} m_x |_{t_2} & \frac{\partial}{\partial v_y} m_x |_{t_2} & \frac{\partial}{\partial v_z} m_x |_{t_2} & \frac{\partial}{\partial B} m_x |_{t_2} \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \frac{\partial}{\partial x} m_y |_{t_1} & \frac{\partial}{\partial y} m_y |_{t_1} & \frac{\partial}{\partial z} m_y |_{t_1} & \frac{\partial}{\partial v_x} m_y |_{t_1} & \frac{\partial}{\partial v_y} m_y |_{t_1} & \frac{\partial}{\partial v_z} m_y |_{t_1} & \frac{\partial}{\partial B} m_y |_{t_1} \\ \frac{\partial}{\partial x} m_y |_{t_2} & \frac{\partial}{\partial y} m_y |_{t_2} & \frac{\partial}{\partial z} m_y |_{t_2} & \frac{\partial}{\partial v_x} m_y |_{t_2} & \frac{\partial}{\partial v_y} m_y |_{t_2} & \frac{\partial}{\partial v_z} m_y |_{t_2} & \frac{\partial}{\partial B} m_y |_{t_2} \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \frac{\partial}{\partial x} m_{v_z} |_{t_1} & \frac{\partial}{\partial y} m_{v_z} |_{t_1} & \frac{\partial}{\partial z} m_{v_z} |_{t_1} & \frac{\partial}{\partial v_x} m_{v_z} |_{t_1} & \frac{\partial}{\partial v_y} m_{v_z} |_{t_1} & \frac{\partial}{\partial v_z} m_{v_z} |_{t_1} & \frac{\partial}{\partial B} m_{v_z} |_{t_1} \\ \frac{\partial}{\partial x} m_{v_z} |_{t_2} & \frac{\partial}{\partial y} m_{v_z} |_{t_2} & \frac{\partial}{\partial z} m_{v_z} |_{t_2} & \frac{\partial}{\partial v_x} m_{v_z} |_{t_2} & \frac{\partial}{\partial v_y} m_{v_z} |_{t_2} & \frac{\partial}{\partial v_z} m_{v_z} |_{t_2} & \frac{\partial}{\partial B} m_{v_z} |_{t_2} \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ \end{bmatrix}$$

where for clarity we adopt the following notation:

$$\frac{\partial H}{\partial \mathbf{x}} \Big|_{\hat{\mathbf{x}}_0} = \frac{\partial m_j}{\partial \mathbf{x}} \Big|_{t_i}$$

That is, for each measurement type  $m_j$  (in this case,  $j = 1 \dots 6$ ), based on the initial estimate of the state vector ( $\mathbf{x}_0$ ) we form the partial derivatives of those measurements with respect to changes in each element of the state vector at each measurement time  $t_i$ .

The vector of corrections to the state is simply:

$$\Delta \mathbf{x} = \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \\ v_x - v_{x_0} \\ v_y - v_{y_0} \\ v_z - v_{z_0} \\ B - B_0 \end{bmatrix}$$

and the vector  $\Delta \mathbf{z}$  of measurement residuals (observed minus predicted measurements) is:

$$\Delta z = \begin{bmatrix} x_{\text{obs}1} - x_{\text{p}1} \\ x_{\text{obs}2} - x_{\text{p}2} \\ \circ \\ y_{\text{obs}1} - y_{\text{p}1} \\ y_{\text{obs}2} - y_{\text{p}2} \\ \circ \\ z_{\text{obs}1} - z_{\text{p}1} \\ z_{\text{obs}2} - z_{\text{p}2} \\ \circ \\ v_{x_{\text{obs}1}} - v_{x_{\text{p}1}} \\ v_{x_{\text{obs}2}} - v_{x_{\text{p}2}} \\ \circ \\ v_{y_{\text{obs}1}} - v_{y_{\text{p}1}} \\ v_{y_{\text{obs}2}} - v_{y_{\text{p}2}} \\ \circ \\ v_{z_{\text{obs}1}} - v_{z_{\text{p}1}} \\ v_{z_{\text{obs}2}} - v_{z_{\text{p}2}} \\ \circ \end{bmatrix}$$

where  $x_p, y_p, z_p, \dots$  are the predicted position and velocity measurements  $\mathbf{m}(\mathbf{x}_0)$  based on the initial estimate of the state,  $x_{\text{obs}}, y_{\text{obs}}, z_{\text{obs}}, \dots$  are the observed measurements  $\mathbf{m}(\mathbf{x})$ .

The partial derivatives contained in matrix  $\mathbf{A}$  may be calculated either analytically or numerically, although analytical partial derivatives can be quite complex (if they can be found at all). The numerical partial derivatives can, however, be formed quite easily as we will see.

If we had the same number of measurements as elements in the state

vector, we could immediately invert the matrix  $A$ , and solve (4) for the first order correction to the initial state vector  $\mathbf{x}_0$ . If there are more measurements than the number of elements in the state vector,  $A$  is no longer square, so we multiply both sides of (4) by the transpose of  $A$ :

$$A^T \Delta z = A^T A \Delta x$$

$$\text{or} \quad \Delta x = (A^T A)^{-1} A^T \Delta z \quad (5)$$

where  $A^T A$  is now square and invertible (assuming it is non-singular).

Equation (5) is the classical equation for least-squares estimation. Since we have linearized a non-linear problem, this equation must be solved iteratively in what is effectively a multi-dimensional Newton-Raphson technique (Smith [9]).

When there is more than one type of measurement as in our case (GPS position and velocity measurements), the measurements must be weighted appropriately such that each measurement residual contributes to the state correction in a way which is proportional to the measurement accuracy. If equation (4) is multiplied by a vector containing the inverse of the standard deviations of each of the measurements being considered, i.e.,

$$W^T = \begin{bmatrix} \frac{1}{\sigma_{x_1}} \\ \frac{1}{\sigma_{x_2}} \\ \circ \\ \circ \\ \circ \\ \frac{1}{\sigma_{v_{z_n}}} \end{bmatrix}$$

we have now expressed each residual in compatible units (standard deviations).  
The equation to be solved becomes:

$$W \Delta z = W A \Delta x$$

$$A^T W^T W \Delta z = A^T W^T W A \Delta x$$

$$\text{or } \Delta x = (A^T W^T W A)^{-1} A^T W^T W \Delta z \quad (6)$$

which is the classical equation for weighted least-squares. The matrix  $W^T W$  is a diagonal matrix whose elements are the inverse of the measurement variances. Once again, since this is a linearization of a non-linear problem, (6) must be solved iteratively, a process which is sometimes referred to as Weighted Least-Squares Differential Correction (Bate [3]).  $(A^T W^T W A)^{-1}$  is an important quantity called the Covariance Matrix. The diagonal elements of the Covariance Matrix are the variances (squares of the one-sigma uncertainties) of the state vector elements being estimated, which is why state estimation techniques are



said to carry with them their own error analysis (see Smith [9]).

In most cases, a priori estimates of the measurement errors in  $W$  are available based on analysis of the sources of error in the observation model equations, or based upon empirical data. In the case of GPS measurements, these error sources were discussed in section II.

## B. Implementation of the WLS Estimator

The WLS differential correction orbit determination algorithm is implemented here as follows:

A. Form the vector of measurements (observations)  $\mathbf{m}$ . The program will process measurement data from two sources.

1. Simulated  $x, y, z, v_x, v_y, v_z$  Earth Centered Inertial (ECI) data generated internally by the MATLAB program using a fourth order Runge-Kutta numerical integrator with various levels of measurement error added (either Gaussian or SA errors). The Runge-Kutta routine implemented here is based on that shown in Garcia [8] with additional force modeling as detailed later in section IV.

2. Actual GPS Navigation Solution data (position/velocity) from NASA's EUVE spacecraft. Since this data was obtained in an Earth-Centered, Earth-Fixed (ECEF) coordinate system, it is transformed to ECI by an off-line "C" program prior to input to the orbit determination algorithm.

B. Using a given initial estimate of the state  $\mathbf{x}_0$ , generate predicted values of the measurements  $\mathbf{m}(\mathbf{x}_0)$  corresponding to each time point of the measurements  $(x_p, y_p, z_p, v_{xp}, v_{yp}, v_{zp})$ .

C. Form the vector of measurement residuals  $\Delta\mathbf{z}$ , the observed minus

predicted values of the measurements.

D. Form the matrix of partial derivatives  $A$  via the finite difference definition of the partial derivative:

$$\frac{\partial m_t}{\partial \xi} = \frac{m_t(x_0, y_0, \dots, \xi_0 + \Delta \xi, \dots, z_{z0}) - m_t(x_0, y_0, \dots, \xi, \dots, z_{z0})}{\Delta \xi} \quad (7)$$

for each measurement  $m_i$  at time  $t_i$ . This will be accomplished by generating a trajectory from  $x_0$ , and either six or seven other trajectories (depending on whether we wish to solve for the  $B$  factor) resulting from small changes in each of the elements of  $x_0$  (denoted  $\Delta \xi$  in equation (7)) and the corresponding changes to the predicted measurements. In this way, the  $A$  matrix is built column-by-column.

E. Using given a priori measurement weights for each measurement type, build the matrix  $W$  and solve for the correction to  $x_0$ :

$$\Delta x = (A^T W^T W A)^{-1} A^T W^T W \Delta z$$

Note: MATLAB performs the matrix inversion by Gaussian Elimination. However, depending on how close the matrix is to singular it may be necessary to use the technique of Singular Value Decomposition (SVD). This involves the decomposition of the matrix to be inverted into a product of three matrices, two of which are orthogonal and the other diagonal. The

matrix may then (most of the time) be easily inverted. If the result is still near singular, examination and modification of the diagonal matrix can often lead to a solution (see Press [7]). Singular or badly conditioned matrices are not uncommon in orbit estimation problems [14], therefore SVD has been implemented for this research to perform the matrix inversion.

F. Form the new estimate of the state:

$$\mathbf{x}_1 = \mathbf{x}_0 + \Delta\mathbf{x}$$

G. Using  $\mathbf{x}_1$  as the new set of initial conditions, iterate through this process again. Exit either when  $\Delta\mathbf{x}$  is very small, or when the weighted RMS of the residuals does not change appreciably between iterations, or when a set number of maximum iterations has been reached. The weighted RMS is defined as:

$$\text{WRMS} = \sqrt{\frac{\sum_{i=1}^n \left( \frac{\Delta z_i}{\sigma_i} \right)^2}{n}}$$

where  $n$  is the total number of measurements.

A coordinate transformation of the residuals to radial, in-track, and cross-track coordinates (RIC) will often reveal important information about orbit determination and prediction errors since radial and cross-track errors are often periodic and bounded while in-track errors usually display both

periodic oscillations and secular growth. The conversion from cartesian to RIC coordinates is accomplished as follows:

If we form unit vectors in the direction of  $\mathbf{r}$  and  $\mathbf{v}$ , i.e.:

$$\hat{\mathbf{r}} = \frac{\dot{\mathbf{r}}}{|\dot{\mathbf{r}}|}$$

$$\hat{\mathbf{v}} = \frac{\dot{\mathbf{v}}}{|\dot{\mathbf{v}}|}$$

We may then form cross-track, in-track, and radial unit vectors as follows:

$$\hat{\mathbf{C}} = \hat{\mathbf{r}} \times \hat{\mathbf{v}}$$

$$\hat{\mathbf{I}} = \hat{\mathbf{v}}$$

$$\hat{\mathbf{R}} = \hat{\mathbf{I}} \times \hat{\mathbf{C}}$$

We then form the residual vector  $\Delta \mathbf{r}$  at each time point:

$$\Delta \vec{\mathbf{r}} = \Delta x \hat{\mathbf{i}} + \Delta y \hat{\mathbf{j}} + \Delta z \hat{\mathbf{k}}$$

Where  $\Delta x$ ,  $\Delta y$ , and  $\Delta z$  are the observed minus predicted values of  $x$ ,  $y$ , and  $z$ . The residuals expressed in RIC coordinates are then:

$$\vec{\mathbf{R}} = \Delta \vec{\mathbf{r}} \cdot \hat{\mathbf{R}}$$

$$\vec{\mathbf{I}} = \Delta \vec{\mathbf{r}} \cdot \hat{\mathbf{I}}$$

$$\vec{\mathbf{C}} = \Delta \vec{\mathbf{r}} \cdot \hat{\mathbf{C}}$$

RIC Coordinates are depicted in Figure 4 below:

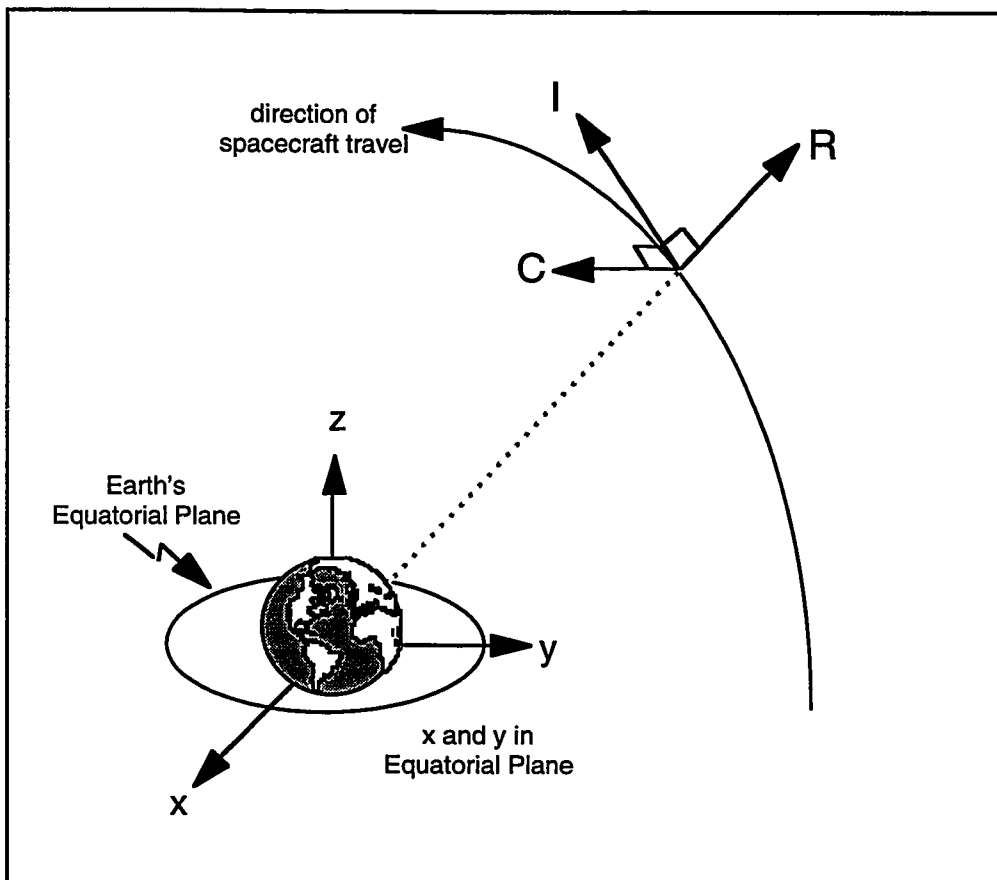


Figure 4: RIC Coordinates

## IV. Numerical Integration of the Equations of Motion

### A. Cowell Formulation of the Equations of Motion

In order to generate predicted measurements we must form the equations of motion of the spacecraft and, using the given initial conditions  $\mathbf{x}_0$ , propagate the state forward to each measurement time. We choose the so-called Cowell formulation of the equations of motion. These equations will be numerically integrated via standard fourth order Runge-Kutta techniques. The Cowell formulation is:

$$\frac{d^2}{dt^2} \hat{\mathbf{r}} + \frac{\mu}{r^3} \hat{\mathbf{r}} = \hat{\mathbf{a}}_p$$

where  $\mu = GM_e$  (the gravitational constant of the Earth), the second term on the left is the gravitational acceleration if the Earth were a spherically symmetric mass distribution (i.e. two-body Keplerian motion), and  $\mathbf{a}_p$  is the sum of any perturbative accelerations due to factors including:

1. asymmetric mass distribution of the Earth.
2. non-conservative perturbations such as atmospheric drag.
3. Solar-lunar-planetary gravitational perturbations.
4. Space vehicle thrusting.

This differs from other, somewhat more complex techniques such as Encke's method, which integrates the difference between the accelerations of a reference orbit (e.g. pure Keplerian motion) and the true or perturbed orbit (see Bate [3]).

For this research, the primary perturbative accelerations to be considered will be due to Earth asphericity and atmospheric drag. For an arbitrary mass distribution such as that of the Earth, it is convenient to form the geopotential as an expansion in spherical harmonics. A form of the geopotential expansion which is independent of longitude (i.e., the "zonal" harmonics which show axial symmetry about the Earth's spin axis) is:

$$\Phi(r,\theta) = \frac{\mu}{r} \cdot \left[ 1 - \sum_{k=2}^{\infty} J_k \cdot \left(\frac{R_e}{r}\right)^k \cdot P_k(\sin\theta) \right] \quad (8)$$

where the  $P_k(\sin(\theta))$  are Legendre Polynomials of order  $k$ ,  $\theta$  is the geocentric latitude,  $\mu = GM_e$ , and  $J_k$  are coefficients which have been determined empirically by observing the motion of satellites. In fact, these coefficients have been determined by optimal estimation techniques similar to those discussed here. That is, previous estimates of the  $J_k$  have been differentially corrected based on actual track data by including these terms as "solve-for parameters" in the state vector. The so-called " $J_2$ " term, which accounts for most of the gravitational perturbation due to the Earth's equatorial bulge, is



several orders of magnitude larger than the other effects due to  $J_3$ ,  $J_4$ , etc. (see Figure 5 below. The data for Figures 5 through 7 was produced by the OASYS software tool [14]).

From Figure 5 below, at the EUVE orbital altitude it is clear that ephemeris errors can be decreased from over 100 km to within a few kilometers over a 12 hour span by incorporation of the  $J_2$  term.

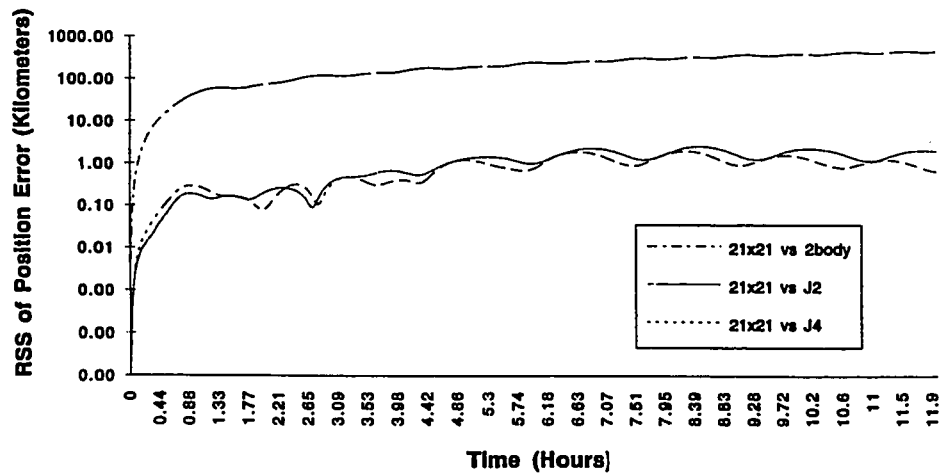


Figure 5: RSS of X, Y, Z Errors Due Geopotential Truncation

Figure 6 below shows the difference between the ephemeris predictions of a model which accounts for geopotential terms through  $J_4$  versus a  $J_2$  propagation, and Figure 7 compares the predictions of a precise gravity model (21x21 zonal and tesseral terms) against a  $J_4$  zonal model.

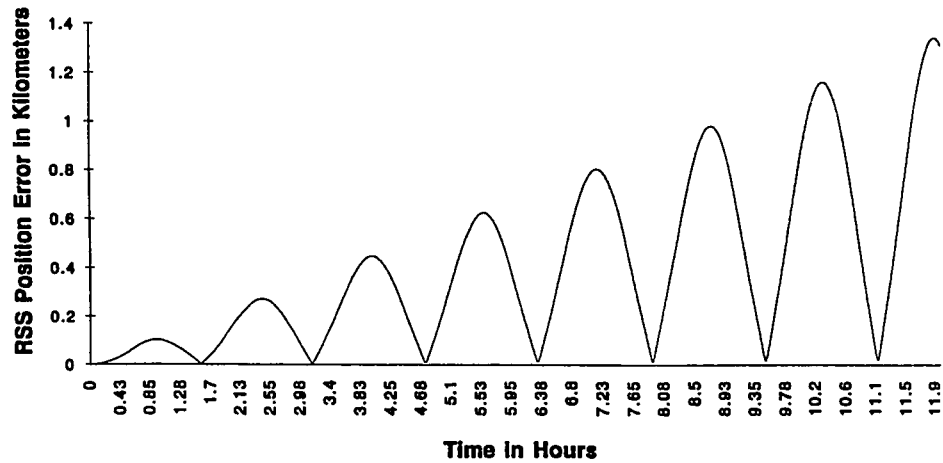


Figure 6: Ephemeris Differences ( $J_2$  vs  $J_4$ )

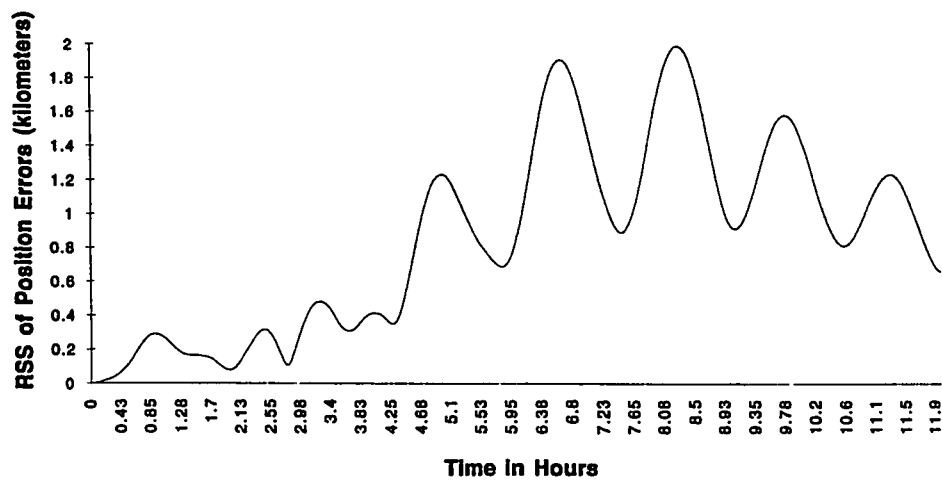


Figure 7: Ephemeris Differences ( $J_4$  vs 21x21)

From these figures we see that inclusion of zonal terms up through  $J_4$

yields a similar accuracy improvement (relative to  $J_2$ ) as can be achieved if all additional zonal/tesseral terms through  $21 \times 21$  are used. For this reason, we will truncate the geopotential after  $J_4$  (zonals only). However, we should therefore expect trajectory position modeling errors on the order of 0.5 to 1 km over prediction spans of a few hours. This will be important when interpreting the results of EUVE orbit determination in Section V.

The gravitational acceleration is obtained by taking the gradient of the geopotential:

$$\vec{a}_g = \nabla\Phi = \left[ \frac{\partial\Phi}{\partial x} \cdot \hat{i} + \frac{\partial\Phi}{\partial y} \cdot \hat{j} + \frac{\partial\Phi}{\partial z} \cdot \hat{k} \right]$$

The first few Legendre Polynomials are:

$$P_0(x) = 1$$

$$P_1(x) = x$$

$$P_2(x) = (3x^2 - 1)/2$$

$$P_3(x) = (5x^3 - 3x)/2$$

$$P_4(x) = (35x^4 - 30x^2 + 3)/8$$

Inserting  $P_2(x)$  through  $P_4(x)$  into (8) and noting that  $\sin(\theta) = z/r$  we have:

$$\begin{aligned}\Phi = & \frac{\mu}{r} \cdot \left[ 1 - \left( \frac{J_2}{2} \cdot \left( \frac{R_e}{r} \right)^2 \left( 3 \cdot \left( \frac{z}{r} \right)^2 - 1 \right) \right) \right. \\ & - \left( \frac{J_3}{2} \cdot \left( \frac{R_e}{r} \right)^3 \cdot \left( 5 \cdot \left( \frac{z}{r} \right)^3 - 3 \cdot \left( \frac{z}{r} \right) \right) \right) \\ & \left. - \left( \frac{J_4}{8} \cdot \left( \frac{R_e}{r} \right)^4 \cdot \left( 35 \cdot \left( \frac{z}{r} \right)^4 - 30 \cdot \left( \frac{z}{r} \right)^2 + 3 \right) + \dots \right] \end{aligned}$$

Taking the partial derivatives and noting that  $r = (x^2 + y^2 + z^2)^{1/2}$  we have:

$$\begin{aligned}\ddot{x} = \frac{\partial \Phi}{\partial x} = & -\frac{\mu x}{r^3} \cdot \left[ 1 + \left( \frac{3J_2}{2} \cdot \left( \frac{R_e}{r} \right)^2 \cdot \left( 1 - 5 \cdot \left( \frac{z}{r} \right)^2 \right) \right) \right. \\ & + \left( \frac{5J_3}{2} \cdot \left( \frac{R_e}{r} \right)^3 \cdot \left( 3 - 7 \cdot \left( \frac{z}{r} \right)^2 \right) \cdot \left( \frac{z}{r} \right) \right) \\ & \left. - \left( \frac{5J_4}{8} \cdot \left( \frac{R_e}{r} \right)^4 \cdot \left( 3 - 42 \cdot \left( \frac{z}{r} \right)^2 + 63 \cdot \left( \frac{z}{r} \right)^4 \right) \right) + \dots \right] \end{aligned}$$

$$\ddot{y} = \frac{\partial \Phi}{\partial y} = \frac{y}{x} \cdot \ddot{x}$$

$$\begin{aligned}\ddot{z} = \frac{\partial \Phi}{\partial z} = & -\frac{\mu z}{r^3} \cdot \left[ 1 + \left( \frac{3J_2}{2} \cdot \left( \frac{R_e}{r} \right)^2 \cdot \left( 3 - 5 \cdot \left( \frac{z}{r} \right)^2 \right) \right) \right. \\ & + \left( \frac{3J_3}{2} \cdot \left( \frac{R_e}{r} \right)^3 \cdot \left( 10 \cdot \left( \frac{z}{r} \right) - \left( \frac{35}{3} \right) \left( \frac{z}{r} \right)^3 - \frac{r}{z} \right) \right) \\ & \left. - \frac{5J_4}{8} \cdot \left( \frac{R_e}{r} \right)^4 \cdot \left( 15 - 70 \left( \frac{z}{r} \right)^2 + 63 \left( \frac{z}{r} \right)^4 \right) + \dots \right] \end{aligned}$$

where terms beyond  $J_4$  are not shown. The above equations have been

implemented in the Cowell equations of motion for this research. It should be noted that references [2] and [3] both contain typographical errors in the equations describing these acceleration terms.

Another potentially significant perturbation, that due to atmospheric drag, can be modeled as follows:

$$\begin{aligned} \frac{d^2}{dt^2} \vec{r}_d &= -\frac{1}{2} \cdot \frac{c_d \cdot A}{m} \cdot \rho \cdot v_a^2 \hat{v}_a \\ &= -\frac{1}{2} \cdot B \cdot \rho \cdot v_a^2 \hat{v}_a \end{aligned} \quad (9)$$

where  $c_d$  is the unitless drag coefficient,  $A$  is the spacecraft cross-sectional area in the direction of the velocity vector,  $m$  is the spacecraft mass,  $\rho$  is the atmospheric density, and  $v_a$  is the magnitude of the spacecraft velocity vector relative to the rotating atmosphere of the Earth. Numerous models of the Earth's atmosphere exist, but their predictions for the density of the upper atmosphere are only approximate since this is highly correlated with unpredictable solar activity. Since  $B$  is a direct multiplier in equation (9), estimation of  $B$  will reduce the effect of errors in the modeling of  $\rho$  on the determination of the drag force.

If we assume that the atmosphere of the Earth is rotating with the Earth at a constant angular velocity  $\omega$  (i.e., the rotational rate of the Earth), then we may use the transformation:

$$\left. \frac{d\vec{r}}{dt} \right|_R = \left. \frac{d\vec{r}}{dt} \right|_F + \vec{r} \times \vec{\omega}$$

where the subscripts R and F refer to rotating and fixed reference frames respectively. Therefore, the velocity components to be inserted into equation (9) are:

$$v_{ax} = v_x + \omega y$$

$$v_{ay} = v_y - \omega x$$

$$v_{az} = v_z$$

where  $v_x$ ,  $v_y$ , and  $v_z$  are the components of the spacecraft velocity in the inertial reference frame.

Since we will be investigating orbit determination techniques for spacecraft such as EUVE in nearly circular low Earth orbits, we will make the simplifying assumption that the atmospheric density is approximately constant for the EUVE orbit. From tabulated values of atmospheric density [13], we will assume a value of  $1 \times 10^{-13}$  kilograms per cubic meter at the EUVE orbital altitude of approximately 500 km (see Figure 8 below).

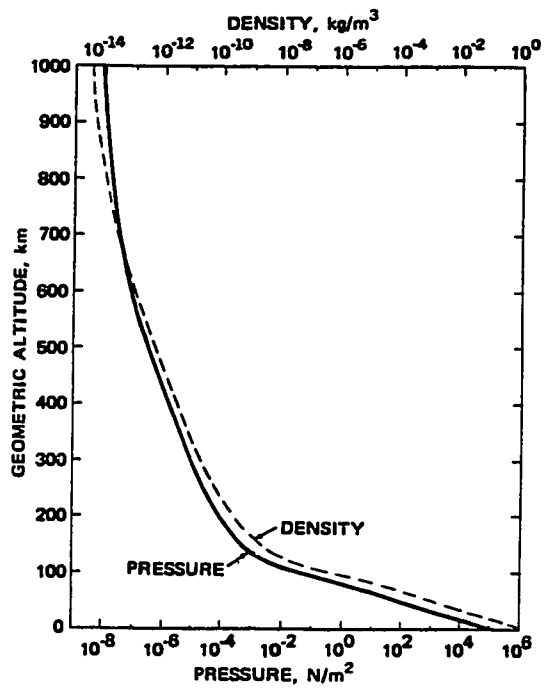


Figure 8: Atmospheric Density as Function of Altitude [13]

## B. Fourth Order Runge-Kutta Numerical Integration

Since the Cowell formulation of the equations of motion is a second order vector ordinary differential equation, this can be reduced to two first order vector ODEs by introducing a new variable  $\mathbf{v}$ . The two first order vector ODEs to be solved are then (see Bate [3]):

$$\frac{d}{dt} \hat{\mathbf{r}} = \hat{\mathbf{v}}$$

$$\frac{d}{dt} \hat{\mathbf{v}} = \hat{\mathbf{a}}_p - \frac{\mu}{r^3} \cdot \hat{\mathbf{r}}$$

which is equivalent to the following six scalar equations:

$$\frac{dx}{dt} = v_x$$

$$\frac{dy}{dt} = v_y$$

$$\frac{dz}{dt} = v_z$$

$$\frac{dv_x}{dt} = a_{px} - \frac{\mu}{r^3} \cdot x \tag{10}$$

$$\frac{dv_y}{dt} = a_{py} - \frac{\mu}{r^3} \cdot y$$

$$\frac{dv_z}{dt} = a_{pz} - \frac{\mu}{r^3} \cdot z$$



Here  $a_p$  will include gravitational perturbations through  $J_4$  and the acceleration due to atmospheric drag.

These equations are of the form:

$$\frac{d}{dt}\dot{\mathbf{y}}(t) = \dot{\mathbf{f}}(\dot{\mathbf{y}}, t)$$

where  $\mathbf{y}$  is the vector of initial conditions ( $x, y, z, v_x, v_y, v_z$ ) and  $\mathbf{f}(\mathbf{y}, t)$  are known functions which are the derivative of the state (i.e. the right hand sides of equations (10) above).

The fourth order Runge-Kutta algorithm which will be used to numerically integrate the Cowell equations of motion is (see Garcia [8]):

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}$$

where:

$$k_1 = h f(t_n, \mathbf{y}_n)$$

$$k_2 = h f(t_n + h/2, \mathbf{y}_n + k_1/2)$$

$$k_3 = h f(t_n + h/2, \mathbf{y}_n + k_2/2)$$

$$k_4 = h f(t_n + h, \mathbf{y}_n + k_3)$$

Here  $h$  is the integration step-size and  $n$  is the integration step number. The

equations of motion of eccentric orbits are best integrated with some form of adaptive step-size control which varies the integration step-size to make it as large as possible while guaranteeing that the truncation error will not exceed some maximum value. However, for the nearly circular orbit of EUVE being considered here a fixed step-size may be used. This is especially advantageous when there is a fixed time interval between tracking data measurements since the state may be propagated directly from one measurement time to the next.

Figure 9 below shows the variation in integration step-size when using adaptive step-size control for the EUVE orbit with a maximum error tolerance of  $1.0 \times 10^{-5}$ .

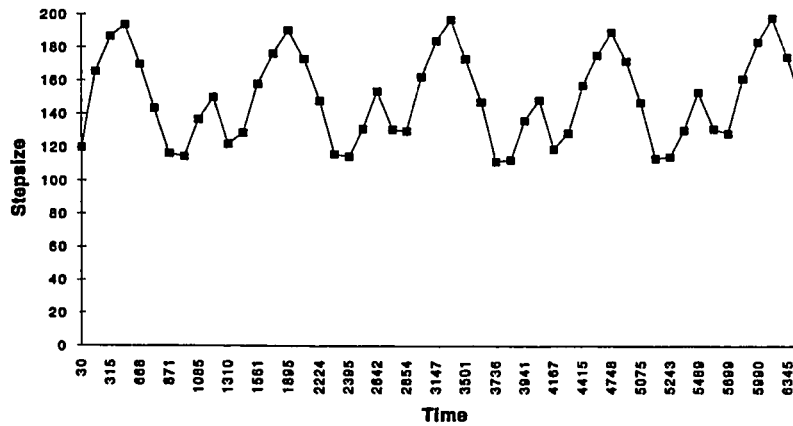


Figure 9: Adaptive Step-Size for EUVE Orbit (seconds)

As can be seen from the figure, the adaptively chosen step-size never falls below 100 seconds. For this research, a fixed stepsize of 30 seconds will be used and the error tolerance should never be exceeded (a 30 second stepsize guarantees  $1.0 \times 10^{-7}$  maximum error).

## V. Results and Discussion

The results to be analyzed here are in two parts:

Part A. We will generate simulated GPS position and velocity measurements using the fourth order Runge-Kutta propagator with two types of measurement error (Gaussian and "SA-like"). This data will then be processed by the Weighted Least-Squares (WLS) batch orbit estimation routine. It is expected that the results of this processing should be superior to part B below because both the data generation and the orbit estimation routines will be internally self-consistent with respect to force modeling and trajectory propagation.

For each type of measurement error three different orbit fits will be performed and contrasted:

1. Fits to position data only.
2. Fits to velocity data only.
3. Fits to both position and velocity data simultaneously.

The intention here is to determine how well a batch WLS orbit estimation routine handles Selective Availability measurement errors relative to Gaussian measurement errors. An additional objective is to determine the relative merits of position data, velocity data, and a combination of both. Some care, however, must be taken in generalizing these results to the processing of actual measurement data since there are additional

measurement and force model errors which are not included in this analysis.

**Part B.** The WLS orbit estimation routine will then be used to process actual GPS position and velocity data from the NASA Extreme Ultra-Violet Explorer spacecraft. Two data sets will be analyzed: one containing the effects of Selective Availability and the other without SA effects. For each of these data sets, three types of orbit fits will be performed (as in Part A above) and the results will be analyzed.

### Part A

#### Simulated Navigation Solution Data Processing with Gaussian and SA Errors

Using the fourth order Runge-Kutta integrator, a set of initial conditions were propagated which are representative of the actual EUVE orbit. A description of the EUVE orbital parameters is provided in Table 2 below:

Parameter	Value
eccentricity	0.00104
inclination	28.45 deg
apogee height	533.1 km
perigee height	518.7 km

Table 2: EUVE Orbital Parameters

## A.1 Simulated Orbit Fits with Gaussian Errors

The initial conditions were propagated over a time span of twelve hours. This trajectory was then saved as the "truth" trajectory. Gaussian-distributed errors with a standard deviation of 35 meters were then added to the "truth" data to form the simulated measurements. The time span and magnitude of error were chosen for consistency with the SA error fits as will be seen later. The integration stepsize was 30 seconds, but measurement data was created every 10 integration steps (5 minute intervals).

The initial conditions were then perturbed along each axis by one kilometer in position and one meter per second in velocity. These initial conditions were then used as the a priori estimate of the spacecraft state vector to be differentially corrected by the WLS estimator using the simulated measurements.

First, the x, y, and z measurements were processed alone (no velocity data). The results are shown in Figure 10 below. As should be expected, the mean of the measurement residuals was essentially zero upon convergence (approximately six iterations).

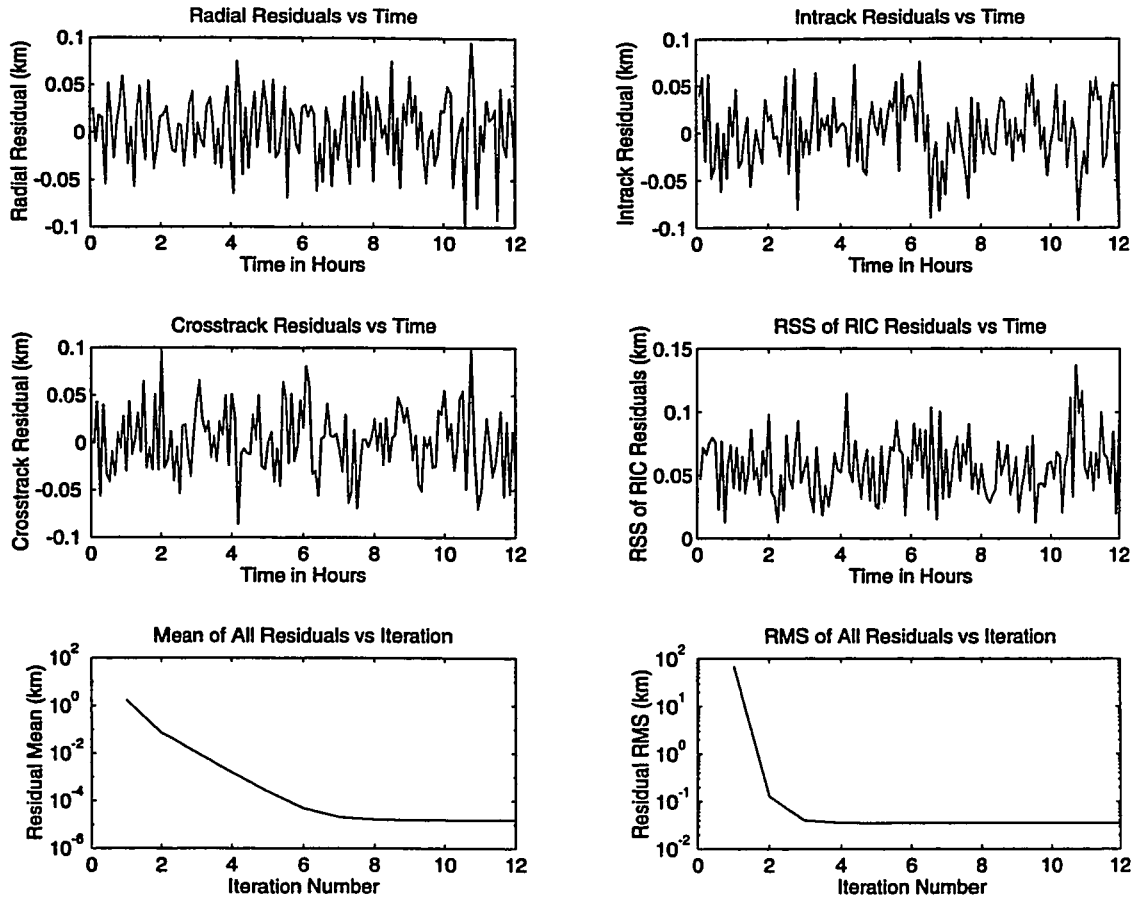


Figure 10: Final Iteration Gaussian Position Fit Data (sigma = 35m)

The final corrected state was then propagated 24 hours and compared to the "truth" trajectory in order to assess the predictive capability of the solution well beyond the measurement span. The results are shown in Figure 11 below:

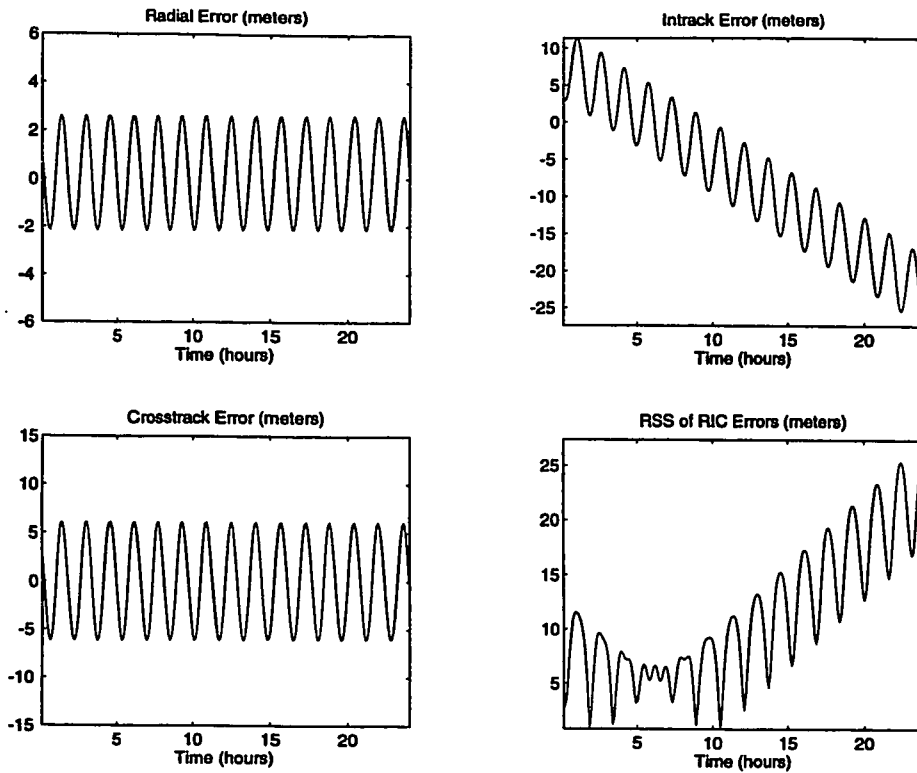


Figure 11: Final Propagated State vs Truth (Gaussian Position Fit)

As can be seen in Figure 11, a maximum error of 25 meters occurs near the end of the prediction span, with the in-track error dominating the secular growth as expected. Cross-track and radial errors are periodic and bounded to within approximately 5 meters.

With the same perturbed initial conditions, orbit determination was then performed using only simulated velocity measurement data with one-sigma Gaussian errors of 0.40 meters/second. Again, this standard deviation was chosen to maintain consistency with the SA orbit fits.

Finally, using the same perturbed initial conditions, a twelve hour orbit fit was performed using both position and velocity data simultaneously with measurement weights of 35 meters and 0.4 meters/second respectively. Graphical results of the WLS position/velocity fit, as well as the velocity-only fit, can be found in Appendix B. Table 3 below summarizes and compares the results obtained from each of the three types of orbit fits with Gaussian measurement errors.

	True Minus Converged Solution (m, cm/s)						24 hr mean RSS of RIC Errors	Converged Residual RMS
	$\Delta x$	$\Delta y$	$\Delta z$	$\Delta v_x$	$\Delta v_y$	$\Delta v_z$		
Position Data	-5.1	1.1	-3.0	0.5	-0.6	0.1	11.3 m	35.29 m
Velocity Data	-116.1	-26.2	-38.7	4.4	-4.7	-9.0	71.9 m	0.40 m/s
Both Position and Velocity Data	-5.6	1.2	-3.3	0.5	-0.6	0.1	10.8 m	35.29 m 0.41 m/s

Table 3: Summary Results of Gaussian Measurement Error Fits

From Table 3 we see that good results were obtained for the position-only fit as well as the WLS position/velocity fit. The fit to velocity data alone was significantly less accurate than either of the other two methods, even though the converged residual RMS equaled the a priori measurement standard deviation.



## A.2 Simulated Orbit Fits With Selective Availability Effects

Thus far we have examined WLS orbit determination accuracy when the error characteristics of the measurements were Gaussian. We will now investigate the effect of measurement errors which display the "colored noise" characteristics of Selective Availability.

A routine was coded in MATLAB to generate pseudo-range errors at one second intervals which mimic the effects of SA. The algorithm used does not duplicate the precise effects of the true SA algorithm used by the GPS spacecraft, but the resulting data has nearly equivalent statistical characteristics. The SA generation algorithm implemented here is due to Zyla [15]. The SA data is generated recursively as follows:

$$s_{k+1} = a_1 s_k + a_2 s_{k-1} + a_3 s_{k-2} + b_1 \eta_{k+1} + b_2 \eta_k + b_3 \eta_{k-1}$$

where the coefficients  $a_1, a_2, a_3, b_1, b_2,$  and  $b_3$  were determined empirically, and  $\eta$  is Normally distributed with mean zero and variance one. The general technique by which empirical data is used to determine the best model which may have produced the data is called "Model Identification". The specific method employed by Zyla is referred to as the Auto-Regressive Integrated Moving Average (ARIMA) technique. Further details on the techniques of Model Identification may be found in Gelb [6].

A one hour time span of simulated pseudo-range errors generated for this research is shown in Figure 12 below.

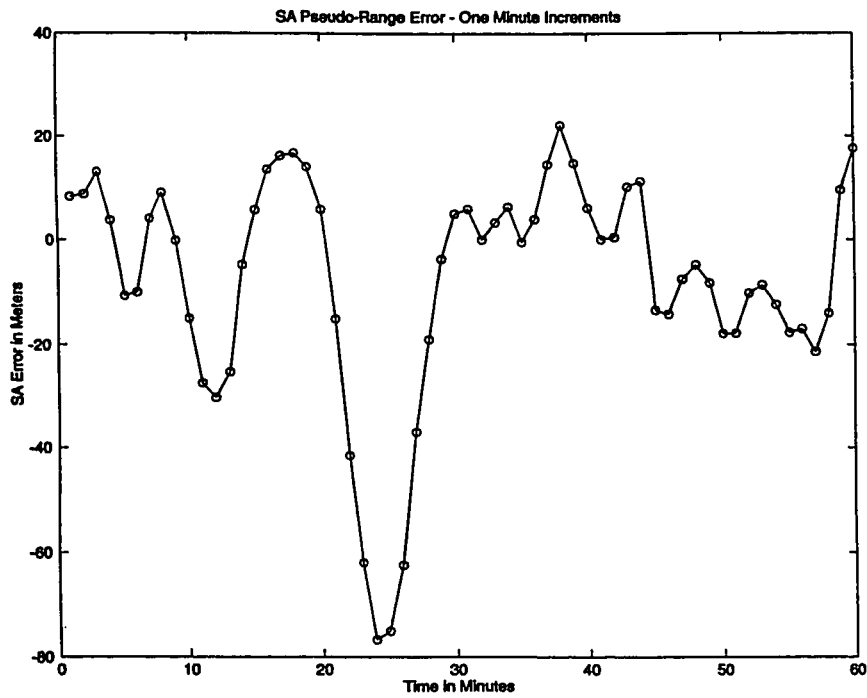


Figure 12: Simulated SA Pseudo-Range Errors (Zyla ARIMA 3x2)

Simulated SA pseudo-range errors were generated for four GPS spacecraft in an arbitrary (though fairly typical) assumed orientation and the vector sum of these errors was taken. Figure 13 below shows the assumed simulation geometry. Since all GPS receivers have "satellite selection" algorithms which attempt to choose the GPS satellites in view that will result in the best Navigation Solution geometry, a configuration of GPS satellites similar

to Figure 13 often results.

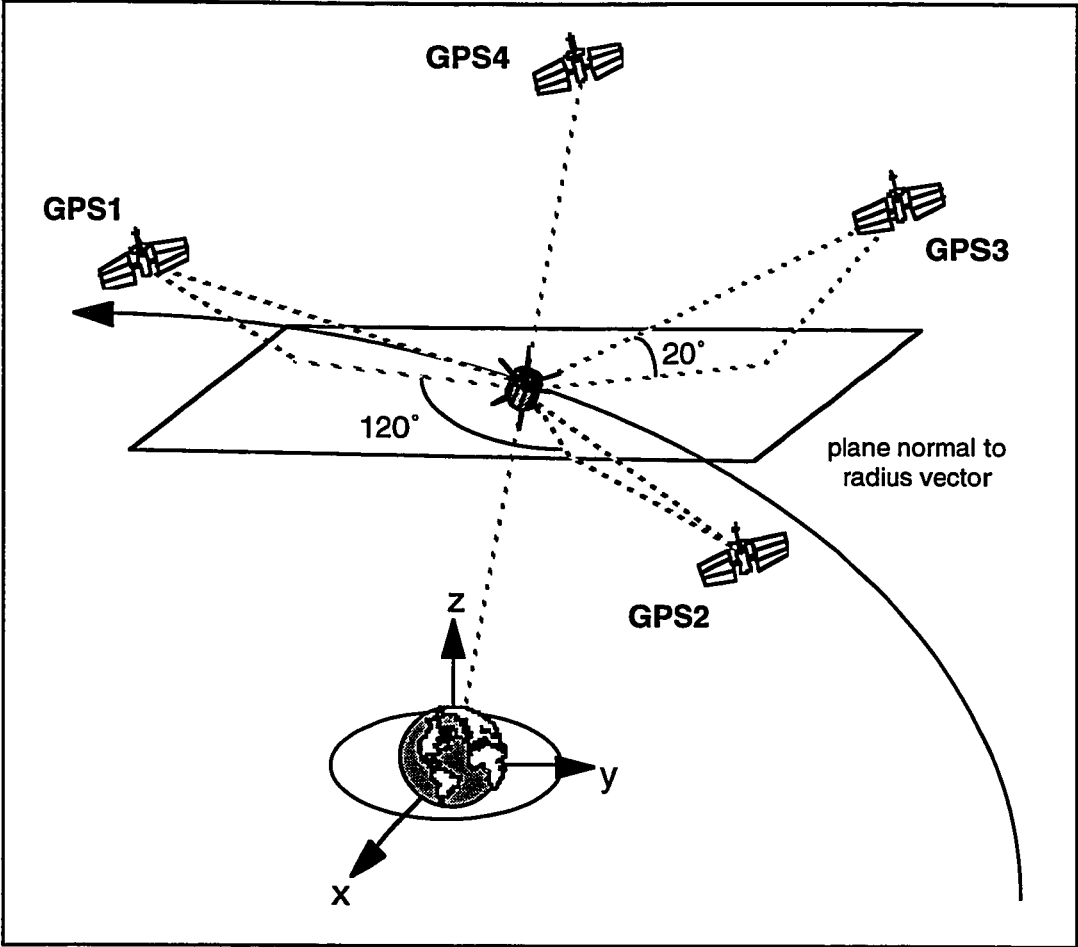


Figure 13: Simulation Geometry

Figure 14 below shows a representative example of a one hour time span of the resulting horizontal and vertical SA errors which were generated for this simulation (compare with Figure 3a).

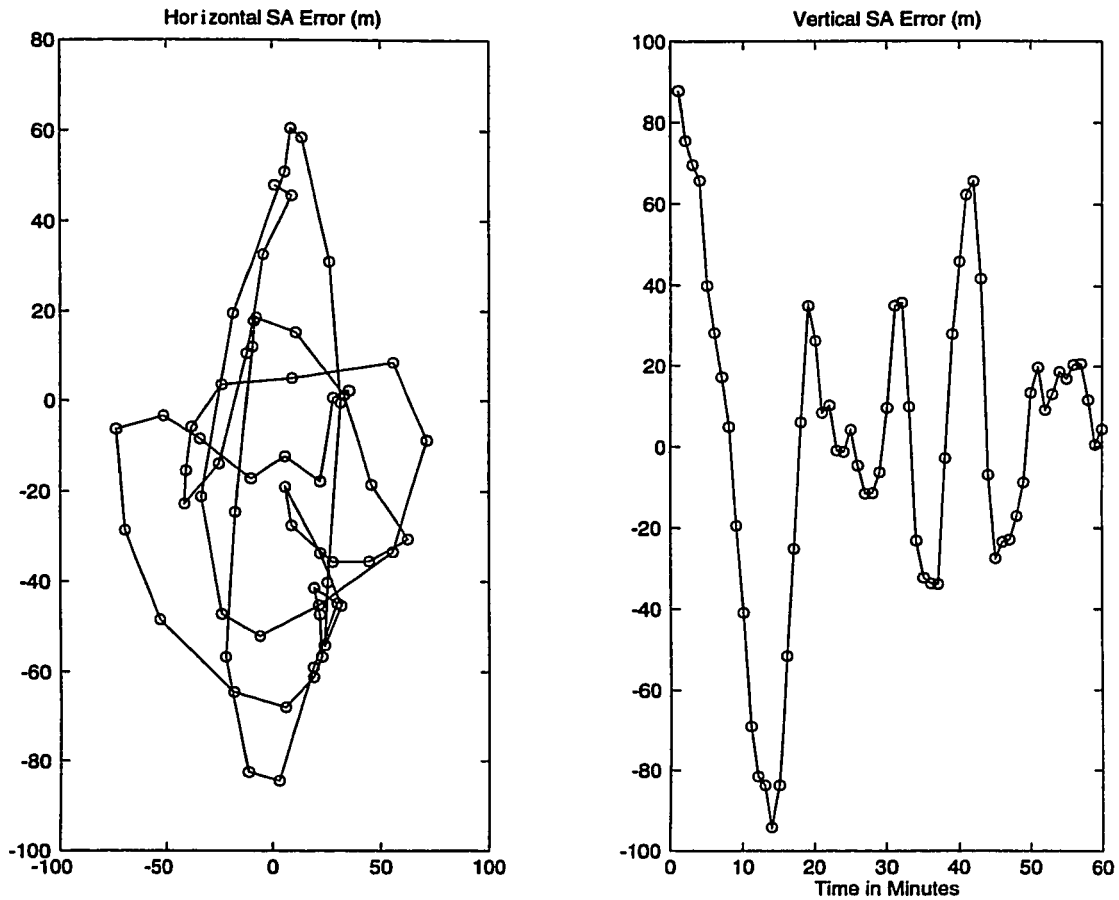


Figure 14: Simulated Horizontal/Vertical SA Errors (one hour- Zyla ARIMA 3x2)

Although it is not obvious from Figures 12 and 14, when observed over a sufficiently long period the SA error distribution begins to exhibit distinctly Gaussian characteristics. Figure 15 below shows the error distribution of simulated SA errors when viewed over a time span of twelve hours.

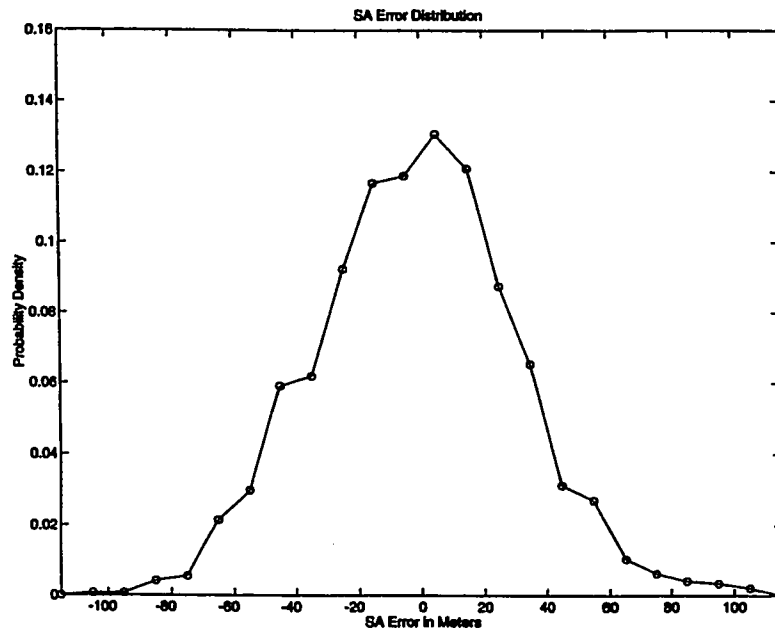


Figure 15: 12 Hour Error Distribution of Simulated SA Data

The statistics of the SA data generated for this research, expressed in RIC coordinates, is shown in Table 4 below. It is the result of a vector summation of SA errors from four GPS satellites in the configuration shown in Figure 13. In order to provide additional realism, several "satellite switches" are included in the simulated data in order to mimic the discontinuity which results when the on-board GPS receiver selects a new GPS satellite as one satellite fades and another rises.

Direction	Mean (meters)	Standard Deviation (meters)
Radial	-3.06	34.83
In-Track	0.80	33.56
Cross-Track	-1.54	31.66

Table 4: SA Error Statistics for this Simulation (12 hour span)

These simulated SA errors were then superimposed on simulated x, y, and z measurement data. A twelve hour fit span was selected in order to allow the Gaussian nature of the Selective Availability errors to exhibit itself. As before, the true initial conditions were perturbed by one kilometer and one meter per second in each axis to initialize the estimation. Figures 16 and 17 below show the orbit determination results and subsequent propagation of the converged solution as compared with "truth" over a 24 hour time span.

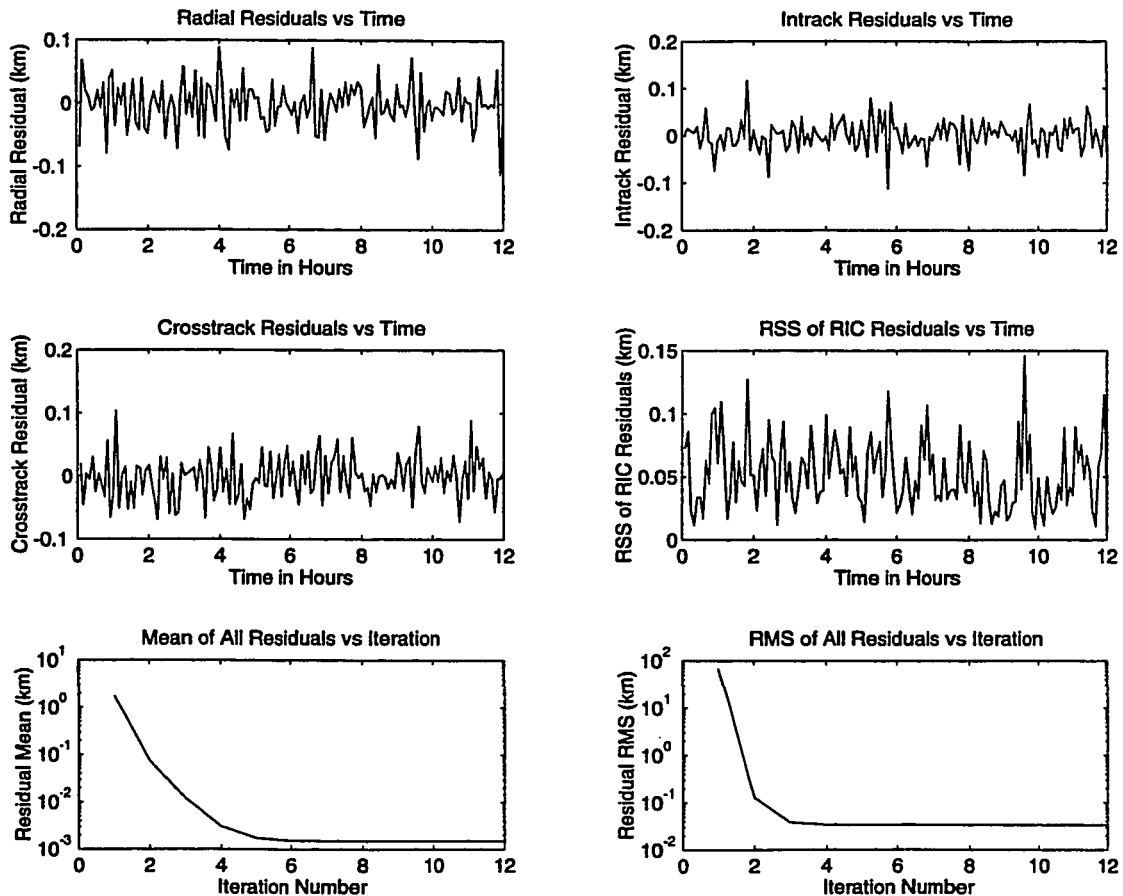


Figure 16: Final Iteration Data for Position Fit with SA

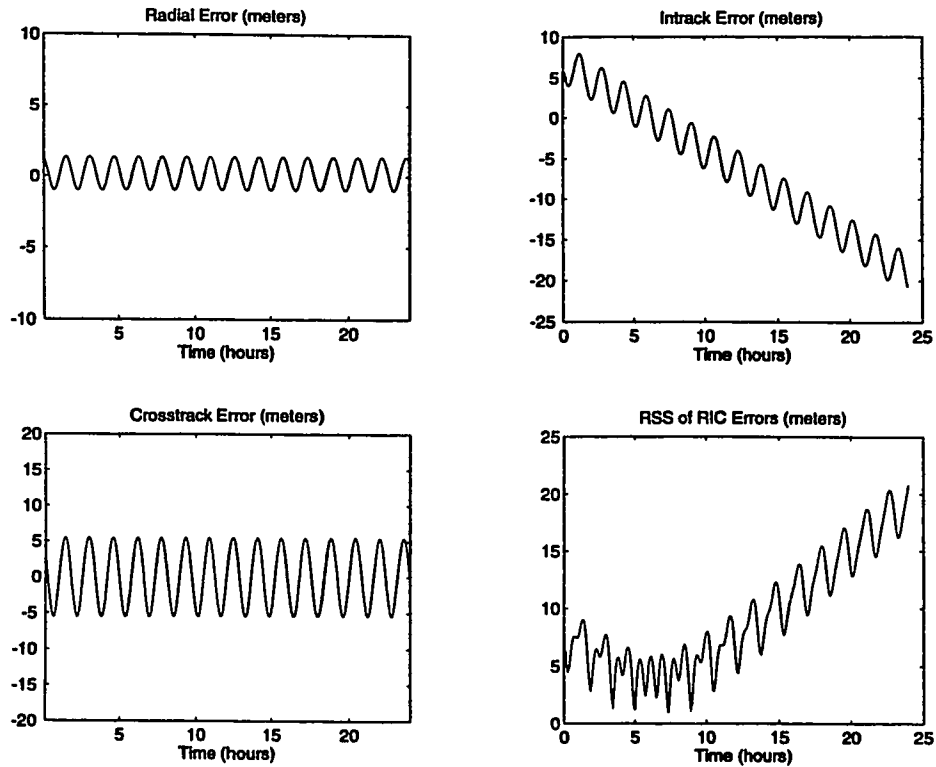


Figure 17: Final Iteration Solution vs Truth (Position Fit with SA)

The maximum propagated error over 24 hours did not exceed 20 meters and the RMS error over that span was 9.3 meters. For the 12 hour span during which data was present the RMS error was 5.7 meters.

Next, a 12 hour orbit fit using only velocity data was performed. Zyla ARIMA 3x2 SA errors were superimposed on the measurements, scaled such that the standard deviation of the errors was 0.40 meters per second to correspond

with the current observed velocity error distribution (Conley [17]). See Appendix B for graphical results.

Finally, both position and velocity measurements were processed simultaneously with measurement weights of 35 meters and 0.40 meters per second respectively (see Appendix B).

Table 5 summarizes and compares the results of the three orbit fits with SA measurement errors.

	True Minus Converged Solution (m, cm/s)						24 hr mean RSS of RIC Errors	Converged Residual RMS
	$\Delta x$	$\Delta y$	$\Delta z$	$\Delta v_x$	$\Delta v_y$	$\Delta v_z$		
Position Data	-7.0	-0.3	-3.1	0.5	-0.6	0.02	9.3 m	33.7 m
Velocity Data	-37.7	-7.3	-32.6	0.15	1.5	-5.3	79.9 m	0.40 m/s
Both Position and Velocity Data	-6.7	0.1	-3.4	0.5	-0.6	0.06	9.3 m	33.7 m 0.41 m/s

Table 5: Summary Results of SA Measurement Error Fits

Once again, we see results of essentially equivalent accuracy for the position-only and WLS position/velocity fits.



Figure 18 below summarizes the comparison between orbit estimation/prediction accuracy for Gaussian versus Selective Availability measurement errors with respect to a key figure of merit, i.e., the 24 hour mean RSS of propagated RIC errors relative to "truth".

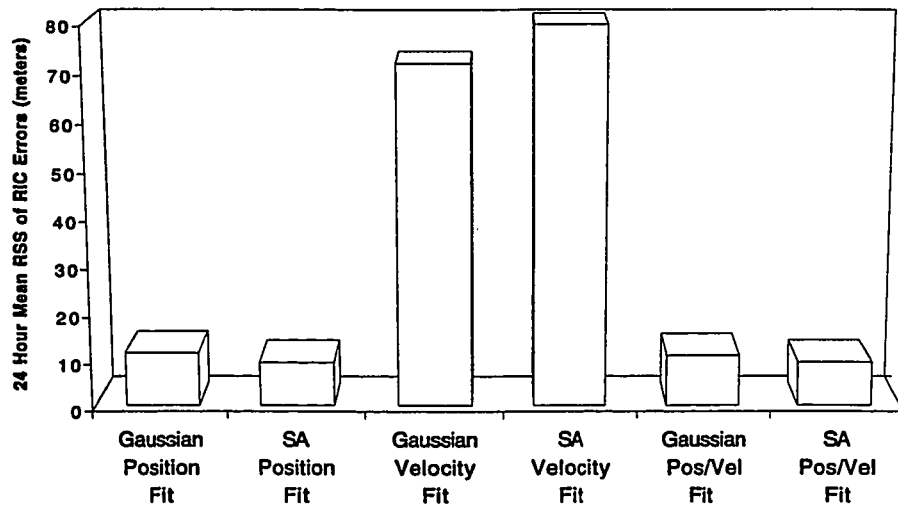


Figure 18: Comparison of Gaussian vs SA Orbit Estimation Results

## **Part B**

### **Processing of EUVE GPS Navigation Solution Data**

We will now use the WLS batch estimator to process GPS data from the EUVE spacecraft. Two data sets were obtained and will be examined. The first data set is from 14 September 1992 when the DOD had not yet turned on the effects of Selective Availability. The second data set was taken on 22 September 1992 during a period when SA was in effect.

Since the measurement and dynamic models used in the WLS orbit estimation routine implemented for this research are not high fidelity, the objective here is not to determine the ultimate potential accuracy of WLS processing of GPS data. Instead, we seek to examine in a qualitative sense the nature of the data to give additional insight into problems which may be encountered when processing real data.

#### **B.1 Description of the EUVE GPS Receiver**

The Extreme Ultra-Violet Explorer was launched in June 1992. It carries a single frequency (L1) P-code receiver capable of processing pseudo-range and carrier phase data from six GPS satellites simultaneously from each of two antennas (twelve independent channels). The receiver performs an on-board least-squares point position estimate (Navigation Solution) every ten seconds and downlinks this data (along with the raw observables) via TDRSS satellites to

the Goddard Space Flight Center where it is stored. The EUVE GPS receiver has no on-board Kalman Filter to smooth the data. In addition, its GPS antennas are placed directly on the spacecraft body causing fairly frequent multipath errors (see Gold [1]).

The EUVE solution data was provided in the WGS-84 Earth Centered Earth Fixed (ECEF) coordinate system. Therefore, the data was first rotated to an Earth Centered Inertial (ECI) True-of-Date coordinate system prior to input to the WLS estimator. The relationship between the ECI and ECEF coordinate systems is shown in Figure 19 below.

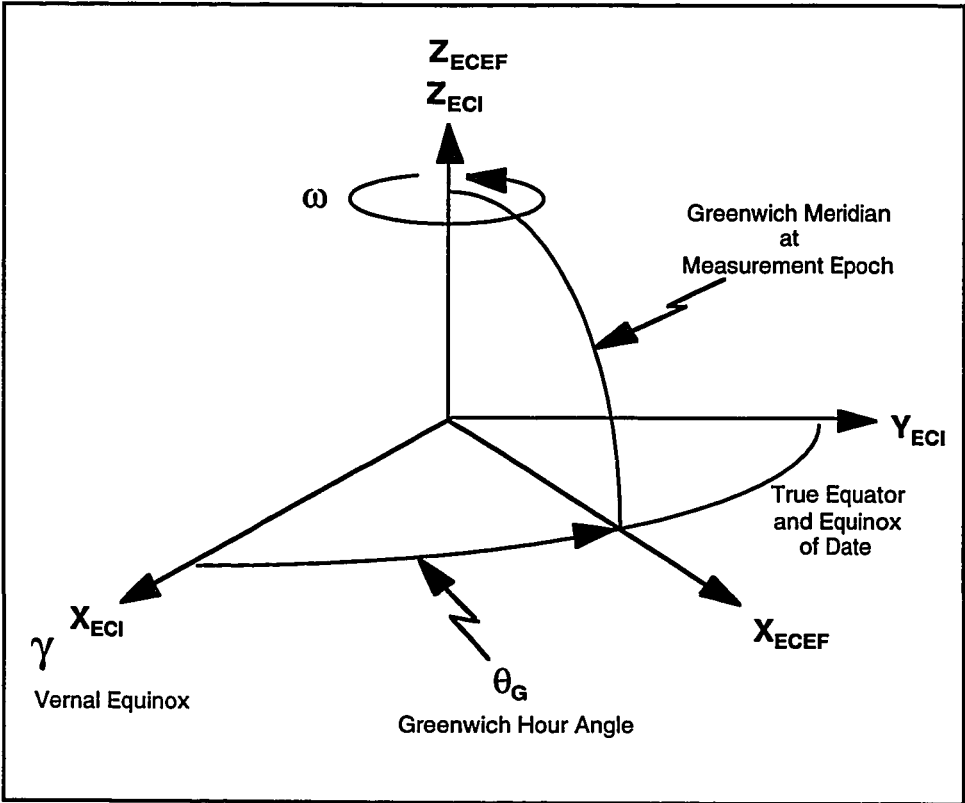


Figure 19: True of Date ECI and ECEF Coordinates

## B.2 EUVE Orbit Fit Results

To initialize the orbit estimation routine, the first Navigation Solution state in the data span was used as the a priori state vector. In addition, although poorly observable over a span of a few hours, the ballistic coefficient  $B$  was included as a solve-for parameter in all of the EUVE orbit fits. It is likely that some unmodeled geopotential acceleration will alias into the drag parameter, helping to minimize the observation residuals.

As an initial check on the data quality, the first iteration measurement residuals were plotted. Figure 20 below shows these residuals from the 14 September 1992 data set.

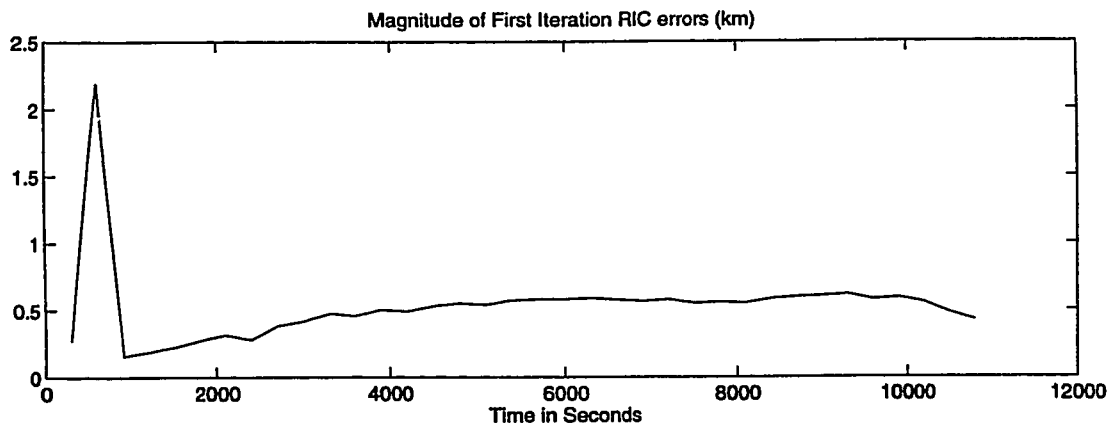


Figure 20: First Iteration EUVE RIC Position Residuals, 14 Sept. 1992

It is clear from the above figure that a short span of data from the beginning of the data set is suspect and must be edited out. A similar problem

was observed near the end of the 22 September 1992 data set, and this was also edited out.

As was done in Part A, three orbit fits were performed with each data set. Final iteration summary graphs for each may be found in Appendix B. We proceed now to discuss the salient results of these orbit fits.

First, as regards the fits to position-only data, a direct comparison of the final iteration RSS RIC residuals is shown in Figure 21 below for the two data sets (with and without SA, 3-sigma outliers removed). For consistency, the same fit span (approximately three hours) was used in both cases.

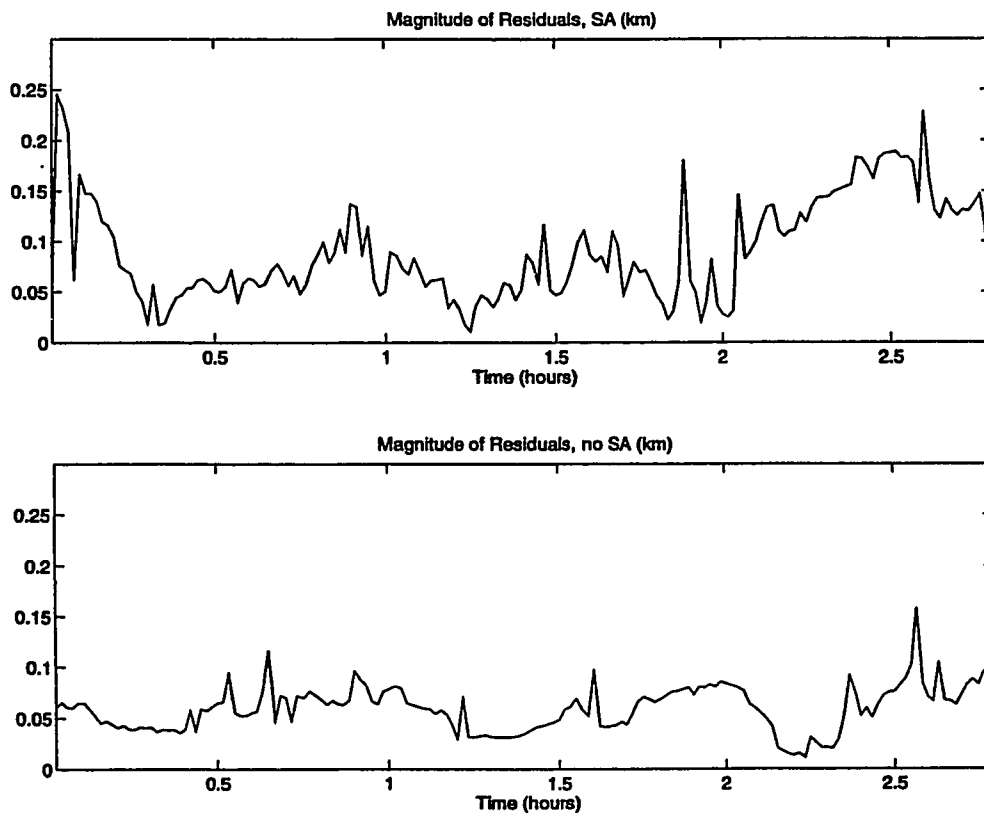


Figure 21: Final Iteration EUVE Position Residuals, With and Without SA

The magnitude of the position residuals for the data without SA is approximately 50 meters. This is somewhat lower than what might have been expected given that geopotential terms beyond  $J_4$  have been truncated in the force model (see Figure 7). It is likely that, to some extent, geopotential modeling errors have been aliased into corrections to each element of the state vector.

Although somewhat difficult to discern since the residuals appear to contain a combination of both systematic measurement errors (such as multipath) and force modeling errors, it is fairly evident that an additional error source exists in the first graph of Figure 21. The error appears to be varying between zero and approximately 150 meters, and so we would surmise that SA is a contributor. This was also reflected in an increase in the standard deviation of the RSS RIC residuals from 32.2 meters (no SA) to 61.5 meters (with SA).

When each measurement type was processed separately, the mean of the measurement residuals was near zero as expected (see Appendix B). However, when WLS orbit fits were performed using position and velocity data simultaneously, a bias of 1.1 meters per second appeared in the radial velocity residuals for both data sets (with and without SA). See Figure 22 below.

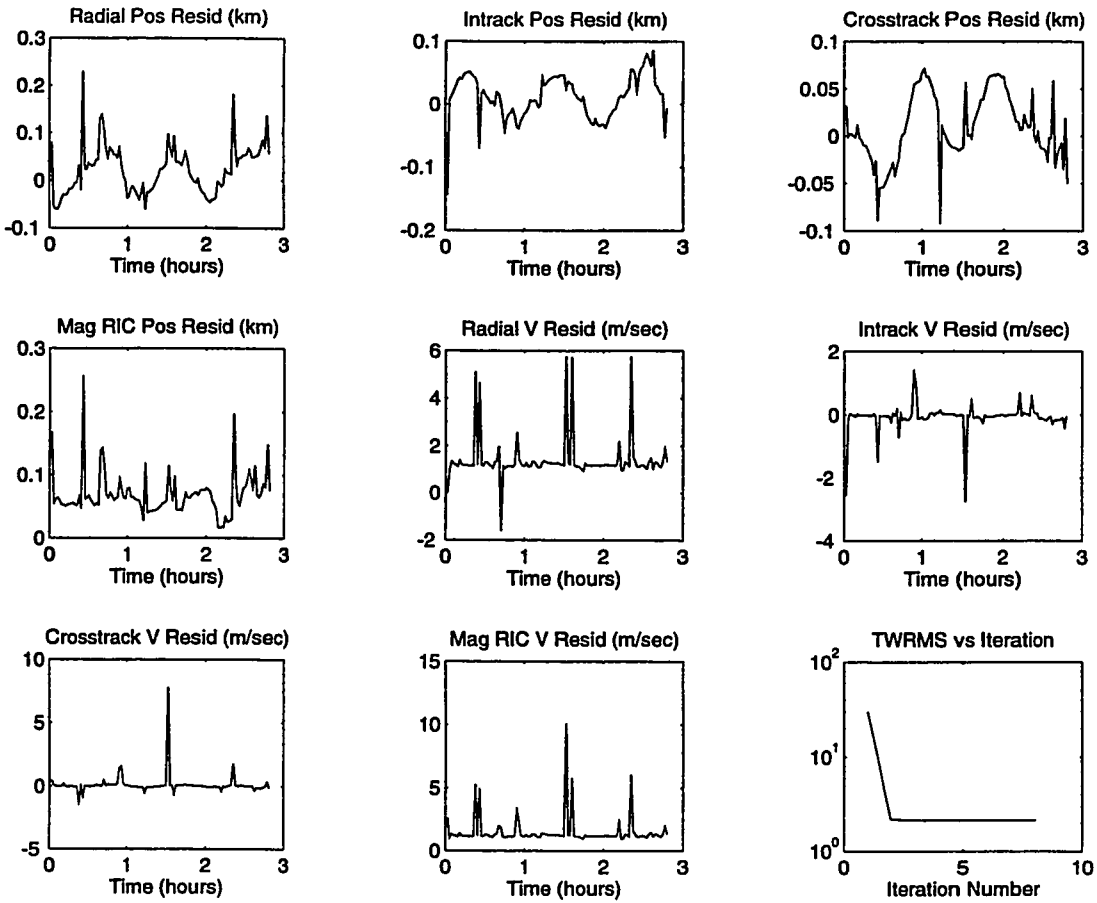


Figure 22: Final Iteration EUVE Position/Velocity Fit Data, No SA

Since the radial velocity bias appeared only when both position and velocity were processed together, this would indicate that an inconsistency exists between the two data types for the EUVE data sets examined here.

An initial hypothesis was that the velocity measurements contained a bias which somehow manifested itself in the radial direction. To test this

hypothesis, the orbit estimation routine was modified to solve for x, y, and z velocity biases. This involved the addition of three columns to the matrix of partial derivatives,  $A$ , such that:

$$\begin{aligned} \frac{\partial m_j}{\partial b_k} &= 0 \text{ for } j \neq k \\ &= 1 \text{ for } j = k \end{aligned}$$

where  $b_k$  is a bias in the  $k^{\text{th}}$  measurement type ( $j, k = 1, 2, \dots 6$ ). The addition of these three solve-for parameters did not, however, remove the observed bias in radial velocity, and no appreciable bias in any of the x, y, and z velocity measurements was found.

The next possibility considered was the existence of a bias in the time-tags of the velocity measurements. Since position and velocity are based on independent measurements of pseudo-range and carrier phase data, it is possible that the epochs of each position/velocity pair be non-coincident and potentially inconsistent, especially since velocity is determined by integrating carrier phase over some finite time interval.

The WLS estimator was modified again to solve for a velocity measurement time bias. In this case, a single additional column was added to the  $A$  matrix by time-shifting the a priori propagated ephemeris and comparing the x, y, and z velocity shift relative to the time shift. When this was done, a velocity time-tag bias of 0.14 seconds was found. Figure 23 below shows the effect of a 0.14 second time bias on the radial, in-track, and cross-track components of velocity.



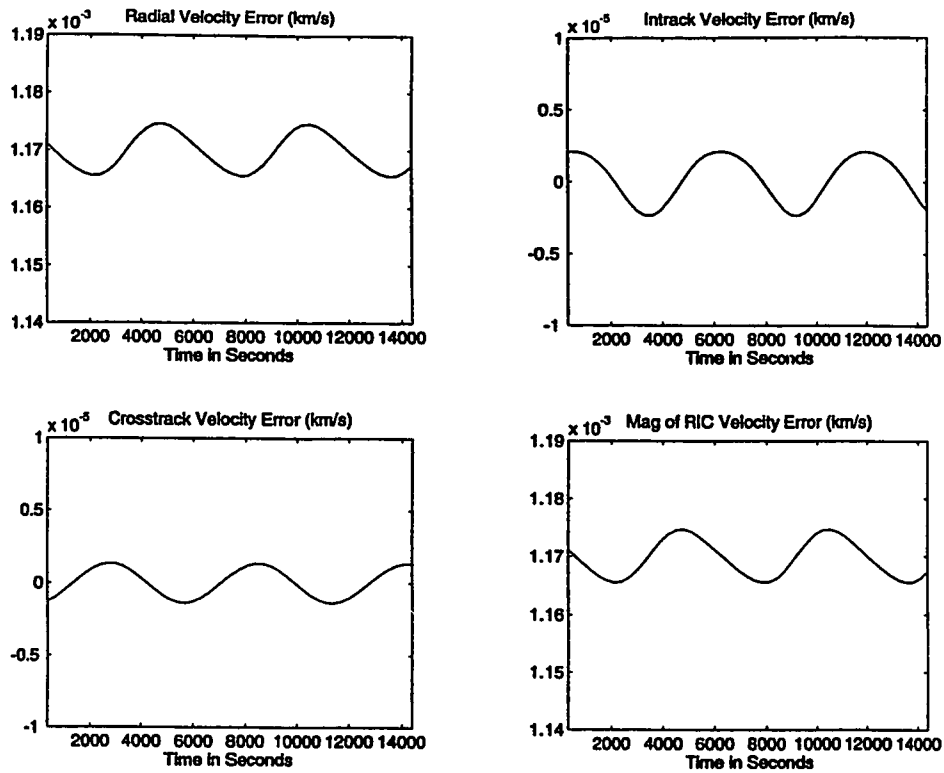


Figure 23: Effect of 0.14 Second Time Bias on Velocity Data

As can be seen from Figure 23, a 0.14 second time bias introduces a nearly constant bias of approximately 1.1 meters per second to radial velocity, but the effect on in-track and cross-track velocity is two orders of magnitude smaller and un-biased. We therefore conclude that this time bias is the likely source of the measurement error.

## **Part C**

### **Conclusions**

The simulation presented here allows us to conclude that Weighted Least-Squares Batch Estimation is a viable orbit determination technique for the processing of GPS Navigation Solution Data in the presence of Selective Availability effects. Results indicate that the SA contribution to the over-all orbit determination error budget will be on the order of 6 meters over a 12 hour fit span, assuming that the SA model used here accurately reflects the true SA error distribution. Other techniques such as Kalman Filtering are also important, especially if the orbit solution is required in near-realtime, but WLS Batch Estimation should be considered an important tool to "defeat SA" since we are able to consider long spans of data in the orbit solution, allowing the Gaussian nature of SA to manifest itself.

Although it is possible that adjustments or additional features (such as incorporation of a position/velocity cross-correlation matrix) to the WLS estimator may provide improved performance, the results here indicate that the addition of velocity data does not significantly improve orbit accuracy beyond that achievable with position data alone. Velocity data neither improved nor degraded the WLS position/velocity fits, and the processing of velocity data alone led to significantly less accurate orbit solutions than position-only fits, possibly due to the velocity measurement error characteristics simulated here. Nevertheless, the use of velocity data should not be discounted, especially in cases where sufficient quantities of position data may not be available.

From the results of orbit fits to actual Navigation Solution data from the EUVE spacecraft it is clear that although SA may be the largest source of measurement error, other measurement errors must also be dealt with such as multipath and ionospheric delays. In addition, even larger error sources than SA are occasionally evident in the EUVE data, perhaps due to loss of receiver lock.

It is therefore important to implement a robust data editing capability in conjunction with the WLS estimator. Accurate force modeling is also important in order to distinguish measurement error from force model error. However, since geopotential models of degree and order 50 (or even higher) are readily available, as well as dynamic atmospheric density models to assist in drag estimation, force model errors below 10 meters are achievable even in low Earth orbit (see Gold [1]).

As we saw, when the EUVE velocity data was used in conjunction with position data, a bias in the velocity time-tags was detected. This points out the advantage to processing independent measurement types simultaneously in order to "cross check" the data and possibly expose measurement errors which might otherwise go undetected. It is therefore important to implement the capability to solve for such measurement biases.

## VI. Future Work

The research presented here may be extended in a number of different areas including the following:

1. Modification of the algorithm to process the raw GPS measurements of pseudo-range and carrier phase.
2. Improved measurement error modeling such as GPS spacecraft orbital position errors, ionospheric errors, antenna phase center, etc. Robust data editing techniques should also be explored.
3. Improved force modeling including geopotential, atmospheric drag, and third body gravitational perturbations.
4. Use of Differential GPS techniques.

As we have seen, a major source of measurement error in the GPS Navigation Solution data is due to the effects of Selective Availability. One way to remove the SA errors is to use Differential GPS (DGPS) techniques. These techniques may be applied in “measurement space” or in “solution space,” i.e. the SA errors may be removed from the pseudo-range and carrier phase data or from the Navigation Solutions, either prior to or during the orbit estimation process.

The basic idea is to incorporate information taken from reference GPS receivers at known locations around the world (the locations are known to millimeter level). Since the locations of these GPS receivers are known, any

difference between the real-time, GPS derived position and the true position will be due (primarily) to the effects of SA. The most common way to eliminate SA is to form "differenced data". This involves the formation of a new derived measurement by subtracting pseudo-range or carrier phase measurements taken simultaneously from two GPS receivers or from two GPS satellites. In this way, a number of errors which are common to each measurement may be eliminated or reduced. As previously mentioned, this requires access to data from a world-wide network of GPS receivers such as was used by JPL for TOPEX/POSEIDON.

## VII. Acknowledgements

I would like to thank the following individuals for their important contributions to this thesis:

First, I would like to thank my thesis advisor Dr. Alejandro Garcia for his excellent comments and feedback throughout the progression of the thesis, including his suggestion to implement Singular Value Decomposition for the matrix inversion in the Differential Correction equations. This proved to be quite useful as poorly conditioned matrices were encountered on a number of occasions during the EUVE processing.

My thanks also go to Kevin Lee of McDonnell Douglas Corporation for providing the EUVE GPS data used in this analysis. My appreciation also goes to Dr. Lou Zyla of McDonnell Douglas for our discussions on methods of simulating SA, and for the use of the SA generation algorithm Zyla ARIMA 3x2.

I would also like to thank Mark Frank of Lockheed Corporation for his many excellent suggestions during a number of technical discussions during the preparation of the thesis.

Finally, my thanks go to Kjell Stakkestad of KinetX Corporation for providing the astrodynamic routines used in the transformation of the EUVE data from Earth-Fixed to ECI True of Date.

## VIII. References

- [1] Gold, Bertiger, Wu, Yunck, Muellerschoen, *GPS Orbit Determination for the Extreme Ultra-Violet Explorer*, Journal of the Institute of Navigation, Washington D.C., May 1994.
- [2] Escobal, P.R., *Methods of Orbit Determination*, © 1983 John Wiley & Sons.
- [3] Bate, Mueller, White, *Fundamentals of Astrodynamics*, © 1971 Dover.
- [4] Battin, R.H., *An Introduction to the Mathematics & Methods of Astrodynamics*, © 1987 AIAA Education Series.
- [5] Leick, Alfred, *GPS Satellite Surveying*, © 1990 Wiley & Sons.
- [6] Gelb, Arthur (Editor), *Applied Optimal Estimation*, © 1974 M.I.T. Press.
- [7] Press, W.H. et. al., *Numerical Recipes in C*, © 1988 Cambridge Press.
- [8] Garcia, A.J., *Numerical Methods for Physics*, Prentice Hall, Englewood Cliffs New Jersey, © 1994.
- [9] Smith, R.H., *A Short Tutorial on Batch and Sequential Estimation*, The Aerospace Corporation, Aerospace Report No. TOR-92 (2489)-1, January 1992.
- [10] Hamill, P., *Lecture Notes on Celestial Mechanics*, San Jose State University, 1991.

## References Continued

- [11] May, Marvin B., *Measuring Velocity with GPS*, GPS World, Eugene Oregon, September 1992.
- [12] Van Graas, Frank, *Coping with SA/AS*, Ohio University, Navtech Seminars 1992.
- [13] Weast, R.C. (Editor), *Handbook of Chemistry and Physics*, 66th edition, CRC Press Inc., Boca Raton, Florida.
- [14] *OASYS Mathematical Foundations*, Second Edition, Integral Systems Inc., © 1994.
- [15] Lear, Montez, Rater, Zyla, *The Effect of Selective Availability on Orbiting Space Vehicles Equipped with SPS GPS Receivers*, Institute of Navigation, Washington D.C., September 1992.
- [16] *Papers Published in Navigation*, (the GPS "Red Books"), Institute of Navigation, Washington D.C., © 1980.
- [17] Conley, Rob, *GPS Performance: What is Normal?*, Journal of the Institute of Navigation, Washington D.C., Fall 1993.
- [18] Yunck, T. P., *A New Chapter in Precise Orbit Determination*, GPS World, Eugene Oregon, October 1992.
- [19] *Navstar GPS User Equipment*, U.S. Government Publication, Public Release Version, February 1991.
- [20] Wells, David, *Guide to GPS Positioning*, © May 1987 Canadian GPS Associates, Fredericton, N.B., Canada.



**Appendix A**  
**Program Listings**

## A.1 EUVWLS\_OD

```
% Program to perform Weighted Least-Squares Batch
% orbit estimation using GPS Navigation Solution
% data (Position and Velocity) from the EUVE spacecraft.

clear;

tau = 30.0;    % integration stepsize (seconds)
maxiter = 6;  % maximum iterations desired
skip = 4;     % skip every nth integration point to
              % get predicted/actual observation times
Bsolve = 1;   % 0 = no solve, 1 = solve for drag param

savedata = 1; % 0 = do not save, 1 = save
sigmaP = 35;  % position sigma in meters
sigmaV = .40; % velocity sigma in meters/sec
endobs = 500; % keep first n obs in GPS data file
tol = .001;   % convergence criteria for WRMS

sigmaPkm = sigmaP/1000.; % Convert to km
sigmaVkm = sigmaV/1000.;

dx = .5;      % delta x for partial deriv (km)
dy = .5;
dz = .5;
dxdot = .001; % km/sec
dydot = .001;
dzdot = .001;

Binit = 2.3*(.01)*(.01)/500.; % Apriori Cd*A/M in MKS units.
dB = Binit;                  % for partial deriv

load INERT6x30.dat; % GPS Nav Solution data at 30 second
load INERT6y30.dat; % increments from EUVE spacecraft
load INERT6z30.dat; % in TOD ECI coordinates, km, km/sec
load INERT6xdot30.dat;
load INERT6ydot30.dat;
load INERT6zdot30.dat;

qq = size(INERT6x30,1);
disp('size of initial file is:');
disp(qq);

%----- edit data from end of span -----
% it was determined that only the first 500 obs were valid

for i = 1:endobs,
    edtINERT6x30(i) = INERT6x30(i); % create new edited obs files
```

```

edtINERT6y30(i) = INERT6y30(i);
edtINERT6z30(i) = INERT6z30(i);
edtINERT6xdot30(i) = INERT6xdot30(i);
edtINERT6ydot30(i) = INERT6ydot30(i);
edtINERT6zdot30(i) = INERT6zdot30(i);
end

clear INERT6x30;
clear INERT6y30;
clear INERT6z30;
clear INERT6xdot30;
clear INERT6ydot30;
clear INERT6zdot30;

%----- use first point in files as initial conditions-----

rinit(1) = edtINERT6x30(1);
rinit(2) = edtINERT6y30(1);
rinit(3) = edtINERT6z30(1);
vinit(1) = edtINERT6xdot30(1);
vinit(2) = edtINERT6ydot30(1);
vinit(3) = edtINERT6zdot30(1);

%-----remove first point in file-----
% it is assumed that the epoch of initial conditions is 30 seconds before
% the first obs point

nn = size(edtINERT6x30,2);
nsteps = nn - 1;      % total number of integration steps
disp('nsteps is:');
disp(nsteps);

for k = 2:nn,
    Bigxobs(k-1) = edtINERT6x30(k);
    Bigyobs(k-1) = edtINERT6y30(k);
    Bigzobs(k-1) = edtINERT6z30(k);
    Bigxdotobs(k-1) = edtINERT6xdot30(k);
    Bigydotobs(k-1) = edtINERT6ydot30(k);
    Bigzdotobs(k-1) = edtINERT6zdot30(k);
end
%----- thin data by skip factor-----
i = 0;
for k = 1:nsteps,
    if rem(k,skip) == 0,
        i = i + 1;
        xobs(i) = Bigxobs(k);      % final observation data is thinned
        yobs(i) = Bigyobs(k);      % by skip factor
        zobs(i) = Bigzobs(k);
        xdotobs(i) = Bigxdotobs(k);
        ydotobs(i) = Bigydotobs(k);
        zdotobs(i) = Bigzdotobs(k);
    end
end

```

```

end
end

nobs = size(xobs,2);
disp('nobs is:');
disp(nobs);

clear Bigxdotobs;
clear Bigydotobs;
clear Bigzdotobs;
clear Bigxdotdotobs;
clear Bigydotdotobs;
clear Bigzdotdotobs;
clear edtINERT6x30;
clear edtINERT6y30;
clear edtINERT6z30;
clear edtINERT6xdot30;
clear edtINERT6ydot30;
clear edtINERT6zdot30;

%----- Create Matrix of measurement variances (WTW) -----
-

for i = 1:(3*nobs),
    W(i) = (1/sigmaPkm)^2.; % 1/kmeters^2
end

for i = ((3*nobs)+1):(6*nobs),
    W(i) = (1/sigmaVkm)^2.; % 1/(kmeters/sec)^2
end

WTW = sparse(diag(W)); % nxn diagonal matrix of 1/variances

clear W;

%-----set apriori initial conditions-----
--

r = rinit;
v = vinit;
B = Binit;
%=====Main Loop =====

for j = 1:maxiter,

disp('iteration number: ');
disp(j);
time = 0.0; % initial time is assumed equal to zero

% generate initial unperturbed ephemeris

i = 0;

```

```

for k = 1:nsteps,
state = [ r(1) r(2) r(3) v(1) v(2) v(3)];
state = rk4(state,time,tau,'J4gravPlusDrag',B); % 4th order Runge Kutta
r = [state(1) state(2) state(3)]; % integration step
v = [state(4) state(5) state(6)];
time = time + tau;
if rem(k,skip) == 0, % skip to obs times
i = i + 1;
propx(i) = r(1); % kilometers
propy(i) = r(2);
propz(i) = r(3);
propxdot(i) = v(1); % kilometers/sec
propydot(i) = v(2);
propzdot(i) = v(3);
proptime(i) = time;
rmag = norm(r); % calculate RIC unit vectors
vmag = norm(v);
rhat = r/rmag;
vhat = v/vmag;
Chat(i,1) = rhat(2)*vhat(3) - rhat(3)*vhat(2); % cross product
Chat(i,2) = rhat(3)*vhat(1) - rhat(1)*vhat(3);
Chat(i,3) = rhat(1)*vhat(2) - rhat(2)*vhat(1);
Ihat(i,1) = vhat(1);
Ihat(i,2) = vhat(2);
Ihat(i,3) = vhat(3);
Rhat(i,1) = vhat(2)*Chat(i,3) - vhat(3)*Chat(i,2);
Rhat(i,2) = vhat(3)*Chat(i,1) - vhat(1)*Chat(i,3);
Rhat(i,3) = vhat(1)*Chat(i,2) - vhat(2)*Chat(i,1);
end
end

disp(time);

```

% generate x-perturbed ephemeris

```

time = 0.;
r = rinit;
r(1) = r(1) + dx;
v = vinit;
i = 0;
for k = 1:nsteps,
state = [ r(1) r(2) r(3) v(1) v(2) v(3)];
state = rk4(state,time,tau,'J4gravPlusDrag',B);
r = [state(1) state(2) state(3)];
v = [state(4) state(5) state(6)];
time = time + tau;
if rem(k,skip) == 0,
i = i + 1;
pertdxx(i) = r(1);
pertdxy(i) = r(2);
pertdxz(i) = r(3);
pertdxvx(i) = v(1);

```

```

    pertdxvy(i) = v(2);
    pertdxvz(i) = v(3);
end
end
disp(time);

                                % calculate first column of A matrix (partial derivatives)
for i = 1:nobs,
    numer = pertdxx(i) - propx(i);
    A(i,1) = numer/dx;

    numer = pertdxy(i) - propy(i);
    A(i+nobs,1) = numer/dx;

    numer = pertdxz(i) - propz(i);
    A(i+(2*nobs),1) = numer/dx;

    numer = pertdxvx(i) - propxdot(i);
    A(i+(3*nobs),1) = numer/dx;

    numer = pertdxvy(i) - propydot(i);
    A(i+(4*nobs),1) = numer/dx;

    numer = pertdxvz(i) - propzdot(i);
    A(i+(5*nobs),1) = numer/dx;
end

clear pertdxx;
clear pertdxy;
clear pertdxz;
clear pertdxvx;
clear pertdxvy;
clear pertdxvz;

                                % generate y-perturbed ephemeris
time = 0.;
r = rinit;
r(2) = r(2) + dy;
v = vinit;
i = 0;
for k = 1:nsteps,
    state = [ r(1) r(2) r(3) v(1) v(2) v(3)];
    state = rk4(state,time,tau,'J4gravPlusDrag',B);
    r = [state(1) state(2) state(3)];
    v = [state(4) state(5) state(6)];
    time = time + tau;
    if rem(k,skip) == 0,
        i = i + 1;
        pertdyx(i) = r(1);
        pertdyy(i) = r(2);
        pertdyz(i) = r(3);
    end
end

```

```

        pertdyvx(i) = v(1);
        pertdyvy(i) = v(2);
        pertdyvz(i) = v(3);
    end
end
disp(time);

                                % calculate second column of A matrix
for i = 1:nobs,
    numer = pertdyx(i) - propx(i);
    A(i,2) = numer/dy;

    numer = pertdyy(i) - propy(i);
    A(i+nobs,2) = numer/dy;

    numer = pertdyz(i) - propz(i);
    A(i+(2*nobs),2) = yy;

    numer = pertdyvx(i) - propxdot(i);
    A(i+(3*nobs),2) = numer/dy;

    numer = pertdyvy(i) - propydot(i);
    A(i+(4*nobs),2) = numer/dy;

    numer = pertdyvz(i) - propzdot(i);
    A(i+(5*nobs),2) = numer/dy;
end

clear pertdyx;
clear pertdyy;
clear pertdyz;
clear pertdyvx;
clear pertdyvy;
clear pertdyvz;

                                % generate z-perturbed ephemeris
time = 0.;
r = rinit;
r(3) = r(3) + dz;
v = vinit;
i = 0;
for k = 1:nsteps,
    state = [ r(1) r(2) r(3) v(1) v(2) v(3)];
    state = rk4(state,time,tau,'J4gravPlusDrag',B);
    r = [state(1) state(2) state(3)];
    v = [state(4) state(5) state(6)];
    time = time + tau;
    if rem(k,skip) == 0,
        i = i + 1;
        pertdzx(i) = r(1);
    end
end

```

```

    pertdzy(i) = r(2);
    pertdzz(i) = r(3);
    pertdzvx(i) = v(1);
    pertdzvy(i) = v(2);
    pertdzvz(i) = v(3);
end
end
disp(time);

                                % calculate third column of A matrix
for i = 1:nobs,
    numer = pertdxx(i) - propx(i);
    A(i,3) = numer/dz;

    numer = pertdzy(i) - propy(i);
    A(i+nobs,3) = numer/dz;

    numer = pertdzz(i) - propz(i);
    A(i+(2*nobs),3) = numer/dz;

    numer = pertdzvx(i) - propxdot(i);
    A(i+(3*nobs),3) = numer/dz;

    numer = pertdzvy(i) - propydot(i);
    A(i+(4*nobs),3) = numer/dz;

    numer = pertdzvz(i) - propzdot(i);
    A(i+(5*nobs),3) = numer/dz;
end

clear pertdxx;
clear pertdzy;
clear pertdzz;
clear pertdzvx;
clear pertdzvy;
clear pertdzvz;

                                % generate xdot-perturbed ephemeris
time = 0.;
r = rinit;
v = vinit;
v(1) = v(1) + dxdot;
i = 0;
for k = 1:nsteps,
    state = [ r(1) r(2) r(3) v(1) v(2) v(3)];
    state = rk4(state,time,tau,'J4gravPlusDrag',B);
    r = [state(1) state(2) state(3)];
    v = [state(4) state(5) state(6)];
    time = time + tau;
    if rem(k,skip) == 0,
        i = i + 1;
    end
end

```



```

    pertdxdotx(i) = r(1);
    pertdxdoty(i) = r(2);
    pertdxdotz(i) = r(3);
    pertdxdotvx(i) = v(1);
    pertdxdotvy(i) = v(2);
    pertdxdotvz(i) = v(3);
end
end
disp(time);

                                % calculate fourth column of A matrix
for i = 1:nobs,
    numer = pertdxdotx(i) - propx(i);
    A(i,4) = numer/dxdot;

    numer = pertdxdoty(i) - propy(i);
    A(i+nobs,4) = numer/dxdot;

    numer = pertdxdotz(i) - propz(i);
    A(i+(2*nobs),4) = numer/dxdot;

    numer = pertdxdotvx(i) - propxdot(i);
    A(i+(3*nobs),4) = numer/dxdot;

    numer = pertdxdotvy(i) - propydot(i);
    A(i+(4*nobs),4) = numer/dxdot;

    numer = pertdxdotvz(i) - propzdot(i);
    A(i+(5*nobs),4) = numer/dxdot;
end

clear pertdxdotx;
clear pertdxdoty;
clear pertdxdotz;
clear pertdxdotvx;
clear pertdxdotvy;
clear pertdxdotvz;

                                % generate ydot-perturbed ephemeris
time = 0.;
r = rinit;
v = vinit;
v(2) = v(2) + dydot;
i = 0;
for k = 1:nsteps,
    state = [ r(1) r(2) r(3) v(1) v(2) v(3)];
    state = rk4(state,time,tau,'J4gravPlusDrag',B);
    r = [state(1) state(2) state(3)];
    v = [state(4) state(5) state(6)];
    time = time + tau;
    if rem(k,skip) == 0,

```

```

    i = i + 1;
    pertdydotx(i) = r(1);
    pertdydoty(i) = r(2);
    pertdydotz(i) = r(3);
    pertdydotvx(i) = v(1);
    pertdydotvy(i) = v(2);
    pertdydotvz(i) = v(3);
end
end
disp(time);

                                % calculate fifth column of A matrix
for i = 1:nobs,
    numer = pertdydotx(i) - propx(i);
    A(i,5) = numer/dydot;

    numer = pertdydoty(i) - propy(i);
    A(i+nobs,5) = numer/dydot;

    numer = pertdydotz(i) - propz(i);
    A(i+(2*nobs),5) = numer/dydot;

    numer = pertdydotvx(i) - propxdot(i);
    A(i+(3*nobs),5) = numer/dydot;

    numer = pertdydotvy(i) - propydot(i);
    A(i+(4*nobs),5) = numer/dydot;

    numer = pertdydotvz(i) - propzdot(i);
    A(i+(5*nobs),5) = numer/dydot;
end

clear pertdydotx;
clear pertdydoty;
clear pertdydotz;
clear pertdydotvx;
clear pertdydotvy;
clear pertdydotvz;

                                % generate zdot-perturbed ephemeris
time = 0.;
r = rinit;
v = vinit;
v(3) = v(3) + dzdot;
i = 0;
for k = 1:nsteps,
    state = [ r(1) r(2) r(3) v(1) v(2) v(3)];
    state = rk4(state,time,tau,'J4gravPlusDrag',B);
    r = [state(1) state(2) state(3)];
    v = [state(4) state(5) state(6)];
    time = time + tau;
end

```

```

if rem(k,skip) == 0,
    i = i + 1;
    pertdxdotx(i) = r(1);
    pertdxdoty(i) = r(2);
    pertdxdotz(i) = r(3);
    pertdxdotvx(i) = v(1);
    pertdxdotvy(i) = v(2);
    pertdxdotvz(i) = v(3);
end
end
disp(time);

                                % calculate sixth column of A matrix
for i = 1:nobs,
    numer = pertdxdotx(i) - propx(i);
    A(i,6) = numer/dzdot;

    numer = pertdxdoty(i) - propy(i);
    A(i+nobs,6) = numer/dzdot;

    numer = pertdxdotz(i) - propz(i);
    A(i+(2*nobs),6) = numer/dzdot;

    numer = pertdxdotvx(i) - propxdot(i);
    A(i+(3*nobs),6) = numer/dzdot;

    numer = pertdxdotvy(i) - propydot(i);
    A(i+(4*nobs),6) = numer/dzdot;

    numer = pertdxdotvz(i) - propzdot(i);
    A(i+(5*nobs),6) = numer/dzdot;
end

clear pertdxdotx;
clear pertdxdoty;
clear pertdxdotz;
clear pertdxdotvx;
clear pertdxdotvy;
clear pertdxdotvz;

                                % generate B-perturbed ephemeris
if Bsolve == 1,

    time = 0.;
    r = rinit;
    v = vinit;
    B = B + dB;
    i = 0;
    for k = 1:nsteps,
        state = [ r(1) r(2) r(3) v(1) v(2) v(3)];
        state = rk4(state,time,tau,'J4gravPlusDrag',B);
    end
end

```

```

r = [state(1) state(2) state(3)];
v = [state(4) state(5) state(6)];
time = time + tau;
if rem(k,skip) == 0,
    i = i + 1;
    pertdBx(i) = r(1);
    pertdBy(i) = r(2);
    pertdBz(i) = r(3);
    pertdBvx(i) = v(1);
    pertdBvy(i) = v(2);
    pertdBvz(i) = v(3);
end
end
disp(time);

                                % calculate seventh column of A matrix
for i = 1:nobs,
    numer = pertdBx(i) - propx(i);
    A(i,7) = numer/dB;

    numer = pertdBy(i) - propy(i);
    A(i+nobs,7) = numer/dB;

    numer = pertdBz(i) - propz(i);
    A(i+(2*nobs),7) = numer/dB;

    numer = pertdBvx(i) - propxdot(i);
    A(i+(3*nobs),7) = numer/dB;

    numer = pertdBvy(i) - propydot(i);
    A(i+(4*nobs),7) = numer/dB;

    numer = pertdBvz(i) - propzdot(i);
    A(i+(5*nobs),7) = numer/dB;
end
end

%.....calculate ATA inverse.....

ATWTWA = (A')*WTW*A;
[U,S,V] = svd(ATWTWA); %decompose via SVD
d = diag(S);          % column vector of diagonal elements of S

n = size(d,1);       % number of diagonal elements

for i = 1:n,
    if d(i) == 0;
        q(i) = 0.;
        disp('found zero element of S matrix');
    else

```

```

        q(i) = 1./d(i);    % create row vector of 1/diagonal elements
        end
    end
    S1 = diag(q);        % create diagonal matrix with elements of q

    ATAi = V*S1*U';      %matrix inverse by SVD method
    disp('ATA inverse with W by SVD is:');
    disp(ATAi);

    %-----TEST-----
    GE_ATA = (A')*A;
    GE_ATAi = inv(GE_ATA);
    disp('ATA inverse by Gaussian Elimination is:');
    disp(GE_ATAi);

    clear S1;
    clear V;
    clear U;
    clear S;

    %..... calculate residuals.....
    residx = xobs - propx;
    residy = yobs - propy;
    residz = zobs - propz;
    residvx = xdotobs - propxdot;
    residvy = ydotobs - propydot;
    residvz = zdotobs - propzdot;

    resid = residx;      % build vector of residuals
    for i = (nobs+1):(2*nobs),
        resid(i) = residy(i-nobs);
    end
    for i = ((2*nobs)+1):(3*nobs),
        resid(i) = residz(i-(2*nobs));
    end
    for i = ((3*nobs)+1):(4*nobs),
        resid(i) = residvx(i-(3*nobs));
    end
    for i = ((4*nobs)+1):(5*nobs),
        resid(i) = residvy(i-(4*nobs));
    end
    for i = ((5*nobs)+1):(6*nobs),
        resid(i) = residvz(i-(5*nobs));
    end

    residT = resid';

    clear resid;
    clear propx;
    clear propy;

```

```

clear propz;
clear propxdot;
clear propydot;
clear propzdot;

%.....calculate WRMS of residuals.....
sumres = 0.;
sumresP = 0.;
sumresV = 0.;
wsumresP = 0.;
wsumresV = 0.;

for i = 1:nobs,
    sqrd1 = (residx(i))^2; % unweighted
    sqrd2 = (residy(i))^2;
    sqrd3 = (residz(i))^2;
    sqrd4 = (residvx(i))^2;
    sqrd5 = (residvy(i))^2;
    sqrd6 = (residvz(i))^2;
    sumresP = sumresP+sqrd1+sqrd2+sqrd3;
    sumresV = sumresV+sqrd4+sqrd5+sqrd6;

    wsqrd1 = (residx(i)/sigmaPkm)^2; % weighted
    wsqrd2 = (residy(i)/sigmaPkm)^2;
    wsqrd3 = (residz(i)/sigmaPkm)^2;
    wsqrd4 = (residvx(i)/sigmaVkm)^2;
    wsqrd5 = (residvy(i)/sigmaVkm)^2;
    wsqrd6 = (residvz(i)/sigmaVkm)^2;
    wsumresP = wsumresP+wsqrd1+wsqrd2+wsqrd3;
    wsumresV = wsumresV+wsqrd4+wsqrd5+wsqrd6;
end

wsumres = wsumresP + wsumresV;

rmsP = sqrt(sumresP/(3*nobs));
rmsV = sqrt(sumresV/(3*nobs));
TWRMS = sqrt(wsumres/(6*nobs));

disp('TWRMS of residuals is (sigma units): ');
disp(TWRMS);
disp('RMS of position residuals is (km): ');
disp(rmsP);
disp('RMS of velocity residuals is (km/sec): ');
disp(rmsV);

wrmsSave(j) = TWRMS;
rmsPsave(j) = rmsP;
rmsVsave(j) = rmsV;

%.....calculate mean of residuals.....
sumx = 0.;

```

```

sumy = 0.;
sumz = 0.;
sumxd = 0.;
sumyd = 0.;
sumzd = 0.;
for i = 1:nobs,
    sumx = sumx + residx(i); % km
    sumy = sumy + residy(i);
    sumz = sumz + residz(i);
    sumxd = sumxd + residvx(i); % km/sec
    sumyd = sumyd + residvy(i);
    sumzd = sumzd + residvz(i);
end

meanx(j) = sumx/nobs;
meany(j) = sumy/nobs;
meanz(j) = sumz/nobs;
meanxd(j) = sumxd/nobs;
meanyd(j) = sumyd/nobs;
meanzd(j) = sumzd/nobs;

meanP(j) = (meanx(j) + meany(j) + meanz(j))/3.;
meanV(j) = (meanxd(j) + meanyd(j) + meanzd(j))/3.;

Wmean(j) = (meanP(j)/sigmaPkm + meanV(j)/sigmaVkm)/2.;

%.....calculate RIC errors.....
for i = 1:nobs,
    radialP(i) = residx(i)*Rhat(i,1) + residy(i)*Rhat(i,2) + residz(i)*Rhat(i,3);
    intrackP(i) = residx(i)*Ihat(i,1) + residy(i)*Ihat(i,2) + residz(i)*Ihat(i,3);
    crosstrkP(i) = residx(i)*Chat(i,1) + residy(i)*Chat(i,2) + residz(i)*Chat(i,3);
    deltaRICP = [radialP(i) intrackP(i) crosstrkP(i)];
    magRICP(i) = norm(deltaRICP);
end

for i = 1:nobs,
    radialV(i) = residvx(i)*Rhat(i,1) + residvy(i)*Rhat(i,2) + residvz(i)*Rhat(i,3);
    intrackV(i) = residvx(i)*Ihat(i,1) + residvy(i)*Ihat(i,2) + residvz(i)*Ihat(i,3);
    crosstrkV(i) = residvx(i)*Chat(i,1) + residvy(i)*Chat(i,2) + residvz(i)*Chat(i,3);
    deltaRICV = [radialV(i) intrackV(i) crosstrkV(i)];
    magRICV(i) = norm(deltaRICV);
end

clear residx;
clear residy;
clear residz;
clear residvx;
clear residvy;
clear residvz;
clear Rhat;
clear Ihat;

```

```

clear Chat;

%.....generate plots.....
subplot(331),
    plot(proptime,radialP)
    title('Radial P Residual in km')
subplot(332),
    plot(proptime,intrackP)
    title('Intrack P Residual in km')
subplot(333),
    plot(proptime,crosstrkP)
    title('Crosstrack P Residual in km')
subplot(334),
    plot(proptime,magRICP)
    title('Mag of RIC P Residuals in km')
subplot(335),
    plot(proptime,radialV)
    title('Radial V Residual in km/sec')
subplot(336),
    plot(proptime,intrackV)
    title('Intrack V Residual in km/sec')
subplot(337),
    plot(proptime,crosstrkV)
    title('Crosstrack V Residual in km/sec')
subplot(338),
    plot(proptime,magRICV)
    title('Mag of RIC V Residuals in km/sec')
subplot(111)

if j < maxiter,
    clear radialP;
    clear intrackP;
    clear crosstrkP;
    clear magRICP;
    clear radialV;
    clear intrackV;
    clear crosstrkV;
    clear magRICV;
    clear proptime;
end

% .....calculate correction to state.....

deltaX = ATAi*(A')*WTW*residT;

disp('correction to X is : ');
disp(deltaX);

%-----TEST-----
GE_deltaX = GE_ATAi*(A')*residT;
disp('Gauss Elimination correction to X is : ');

```



```

disp(GE_deltaX);

%-----eigenvalues-----
E_ATAi = eig(ATAi);
disp('eigenvalues of Cov Matrix are: ');
disp(E_ATAi);

clear ATAi;
clear A;
clear residT;

%----- compose state vector -----
if Bsolve == 1,
    oldX = [rinit(1) rinit(2) rinit(3) vinit(1) vinit(2) vinit(3) Binit];
else
    oldX = [rinit(1) rinit(2) rinit(3) vinit(1) vinit(2) vinit(3)];
end

oldXT = oldX';

newX = oldXT + deltaX;
disp('new X is: ');
disp(newX);

if Bsolve == 1,
    Bsave(j) = newX(7);
end

%.....set initial values of state.....
r = [newX(1) newX(2) newX(3)];
v = [newX(4) newX(5) newX(6)];
rinit = r;
vinit = v;

if Bsolve == 1,
    Binit = newX(7);
    B = Binit;
end

%----- exit if delta WRMS < tolerance -----
if j > 1,
    if abs(wrmssave(j) - wrmssave(j-1))) < tol,
        j = maxiter + 1;
        disp('converged');
    end
end

end
%===== end of main loop =====
if savedata == 1,

```

```

proptimeT = proptime';
radialPT = radialP';
intrackPT = intrackP';
crosstrkPT = crosstrkP';
magRICPT = magRICP';
radialVT = radialV';
intrackVT = intrackV';
crosstrkVT = crosstrkV';
magRICVT = magRICV';

save timeT.dat proptimeT -ascii;
save radialPT.dat radialPT -ascii;
save intrackPT.dat intrackPT -ascii;
save crosstrkPT.dat crosstrkPT -ascii;
save magRICPT.dat magRICPT -ascii;

save radialVT.dat radialVT -ascii;
save intrackVT.dat intrackVT -ascii;
save crosstrkVT.dat crosstrkVT -ascii;
save magRICVT.dat magRICVT -ascii;

save rmsP.dat rmsPsave -ascii;
save rmsV.dat rmsVsave -ascii;
save twrms.dat wrmssave -ascii;

save Wmean.dat Wmean -ascii;
save meanP.dat meanP -ascii;
save meanV.dat meanV -ascii;

if Bsolve ==1,
    save B.dat Bsave -ascii;
end

end

```

## A.2 J4gravPlusDrag

```

function deriv = J4gravPlusDrag(time,y,B)
% This program takes a set of initial conditions y (the state) where
% y = [r(1) r(2) r(3) v(1) v(2) v(3)] in km and km/sec
% and the Drag Parameter B, and returns the derivative of the state
% deriv = [dr(1)/dt dr(2)/dt dr(3)/dt dv(1)/dt dv(2)/dt dv(3)/dt]
% The time is not used in this version

Re = 6378.140;      % Equatorial radius of Earth in Km
J2 = 1082.627e-6;  % second zonal
J3 = -2.536414e-6; % third zonal
J4 = -1.623350e-6; % fourth zonal
GM = 398600.5;    % Grav. const. in km cubed per sec squared

omegaE = 7.292115856e-5; % rotation rate of Earth in radians/sec
rho = 5.0e-13;          % atmos density assumed constant at 500km altitude (kg/m
sqrd)

r = [y(1) y(2) y(3)]; % Separate the state vector into r, v components
v = [y(4) y(5) y(6)];

%-----Compute acceleration due to drag-----

Vrel(1) = v(1) + omegaE*r(2); % components of velocity relative to
Vrel(2) = v(2) - omegaE*r(1); % rotating atmosphere
Vrel(3) = v(3);
magVrel = norm(Vrel);
unitVrel = Vrel/magVrel;

drag(1) = - (1/2)*B*rho*(Vrel(1)^2.)*unitVrel(1);
drag(2) = - (1/2)*B*rho*(Vrel(2)^2.)*unitVrel(2);
drag(3) = - (1/2)*B*rho*(Vrel(3)^2.)*unitVrel(3);

%-----Compute gravitational acceleration-----
%----- include accelerations through J4 zonal-----

nr = norm(r);
a = -GM*((r(1)/nr)/nr^2.);
aa = -GM*((r(3)/nr)/nr^2.);
b = Re/nr;
bb = b^2.;
bbb = b^3.;
bbbb = bb^2.;
c = r(3)/nr;
cc = c^2.;

```

```

ccc = c^3.;
cccc = cc^2.;

J2accel1 = J2*(3./2.)*bb*(1.0 - 5.0*cc);
J3accel1 = J3*(5./2.)*bbb*(3. - 7.*cc)*c;
J4accel1 = -J4*(5./8.)*bbbb*(3. - 42.*cc + 63.*cccc);
accelg(1) = a*(1.0 + J2accel1 + J3accel1 + J4accel1);

accelg(2) = (r(2)/r(1))*accelg(1);

J2accel3 = J2*(3./2.)*bb*(3.0 - 5.0*cc);
J3accel3 = J3*(3./2.)*bbb*(10.*c - (35./3.)*ccc - (nr/r(3)));
J4accel3 = -J4*(5./8.)*bbbb*(15. - 70.*cc + 63.*cccc);
accelg(3) = aa*(1.0 + J2accel3 + J3accel3 + J4accel3);

sumaccel(1) = accelg(1) + drag(1);
sumaccel(2) = accelg(2) + drag(2);
sumaccel(3) = accelg(3) + drag(3);

deriv = [v(1) v(2) v(3) sumaccel(1) sumaccel(2) sumaccel(3)];
return;

```

### A.3 rk4 (due to Garcia, [8])

```
function xout = rk4(x,t,tau,derivsRK,param)
% Runge-Kutta integrator (4th order)
% Input arguments -
% x = current value of dependent variable
% t = independent variable (usually time)
% tau = step size (usually timestep)
% derivsRK = right hand side of the ODE; derivsRK is the
%           name of the function with returns dx/dt
%           Calling format derivsRK(t,x,param).
% param = extra parameters passed to derivsRK
% Output arguments -
% xout = new value of x after a step of size tau
half_tau = 0.5*tau;
F1 = feval(derivsRK,t,x,param);
th = t + half_tau;
xt = x + half_tau*F1;
F2 = feval(derivsRK,th,xt,param);
xt = x + half_tau*F2;
F3 = feval(derivsRK,th,xt,param);
th = t + tau;
xt = x + tau*F3;
F4 = feval(derivsRK,th,xt,param);

xout = x + tau/6.*(F1 + F4 + 2.*(F2+F3));

return;
```

## A.4 Zyla3x2

```
% Program to generate simulate SA by the method of Zyla ARIMA 3x2
% The data must be generated in one second time steps
clear;

nsteps = 12000;    % number of seconds of SA to generate

a1 = 2.671523955;  % fit coefficients
a2 = -2.345122555;
a3 = 0.6735717069;
b1 = 1.708581101;
b2 = -3.328413711;
b3 = 1.629545591;

s(1) = +4;        % initialization - these may be varied
s(2) = -1.1;
s(3) = -3;

for i = 1:nsteps,
    n(i) = randn; % Normal with mean 0, variance 1 - N(0,1)
    plottime(i) = i; % 1 sec increments
end

for k = 3:(nsteps - 1),
    s(k+1) = a1*s(k) + a2*s(k-1) + a3*s(k-2) + b1*n(k+1) + b2*n(k) + b3*n(k-1);
end

meanSA = mean(s);
disp('mean is : ');
disp(mean);

sigmaSA = std(s);
disp('sigma is : ');
disp(sigma);

save sasave.txt s -ascii;

subplot(111),
    plot(plottime,s)
    title('SA PR Error vs Time')
subplot(111)
```

## A.5 simGauss\_hdr

```
% Header file to generate GPS Navigation Solution data
% (position/velocity) with Gaussian errors. This header
% file may be inserted to replace the data setup portion of EUVWLS_OD
clear;

tau = 30.0;      % integration stepsize
maxiter = 8;    % maximum iterations
hours = 12;     % fit span in hours
skip = 10;      % skip every nth integration point
Bsolve = 0;     % 0 = no solve, 1 = solve
% tol = 0.001; % convergence criteria

savedata = 1;   % 0 = do not save, 1 = save
sigmaP = 35;    % position sigma in meters
sigmaV = .40;   % velocity sigma in meters/sec

m1 = 1000/sigmaP; % to scale random data in km
m2 = 1000/sigmaV; % to scale random data in km/sec

nsteps = (3600*hours)/tau;
dx = .5;       % kilometers
dy = .5;
dz = .5;
dxdot = .001; % km/sec
dydot = .001;
dxdot = .001;

rinit = [3042.8 -5257.4 -3274.9]; %perturbed initial condition (km)
vinit = [6.6522 3.6776 0.2531]; % km per sec
Binit = 2.3*(.01)*(0.01)/500.;

rtrue = [3043.8 -5256.4 -3275.9]; %true initial condition
vtrue = [6.6532 3.6786 0.2541];
Btrue = 2.3*(.01)*(0.01)/500.;

%----- generate simulated obs and true trajectory-----
r = rtrue;
v = vtrue;
B = Btrue;
time = 0.; % t = 0 at initial r,v

i = 0;
for k = 1:nsteps,
    state = [ r(1) r(2) r(3) v(1) v(2) v(3)];
    state = rk4(state,time,tau,'J4gravPlusDrag',B);
    r = [state(1) state(2) state(3)];
    v = [state(4) state(5) state(6)];
```

```

time = time + tau;
if rem(k,skip) == 0,
    i = i + 1;
    xobs(i) = r(1);
    yobs(i) = r(2);
    zobs(i) = r(3);
    xdotobs(i) = v(1);
    ydotobs(i) = v(2);
    zdotobs(i) = v(3);
end
end
disp(time);

nobs = size(xobs,2);
disp('number of observations is:');
disp(nobs);

        % add Gaussian errors

noise1 = randn(size(xobs))/m1;
noise2 = randn(size(yobs))/m1;
noise3 = randn(size(zobs))/m1;
xobs = xobs + noise1;
yobs = yobs + noise2;
zobs = zobs + noise3;

noise4 = randn(size(xobs))/m2;
noise5 = randn(size(yobs))/m2;
noise6 = randn(size(zobs))/m2;
xdotobs = xdotobs + noise4;
ydotobs = ydotobs + noise5;
zdotobs = zdotobs + noise6;

%----- Create Matrix of measurement variances (WTW) -----
sigmaPkm = sigmaP/1000.;
sigmaVkm = sigmaV/1000.;
for i = 1:(3*nobs),
    W(i) = (1/sigmaPkm)^2.; % 1/kmeters
end

for i = ((3*nobs)+1):(6*nobs),
    W(i) = (1/sigmaVkm)^2.; % 1/kmeters/sec
end

WTW = sparse(diag(W)); % nxn diagonal matrix of 1/variances

r = rinit;
v = vinit;
B = Binit;
%===== Main Loop =====
%----- enter main orbit estimation loop -----

```



## A.6 simSA\_hdr

```
% Header file to prepare simulated Navigation Solution data
% (position/velocity) with SA effects for input to the WLS
% Batch Estimator EUVWLS_OD
clear;

tau = 30.0; % integration stepsize (seconds)
maxiter = 6; % max iterations
hours = 12; % fit span in hours
skip = 10; % skip every nth integration point

Bsolve = 0; % 0 = no solve, 1 = solve
savedata = 1; % 0 = do not save, 1 = save
sigmaP = 35; % position sigma in meters
sigmaV = .40; % velocity sigma in meters/sec
tol = 0.001; % convergence criteria

sigmaPkm = sigmaP/1000.;
sigmaVkm = sigmaV/1000.;
nsteps = (3600*hours)/tau;
dx = .5; % kilometers
dy = .5;
dz = .5;
dxdot = .001; % km/sec
dydot = .001;
dzdot = .001;

rinit = [3042.8 -5257.4 -3274.9]; %perturbed initial condition (km)
vinit = [6.6522 3.6776 0.2531]; % km per sec
Binit = 2.3*(.01)*(0.01)/500.;

rtrue = [3043.8 -5256.4 -3275.9]; %true initial condition
vtrue = [6.6532 3.6786 0.2541];
Btrue = 2.3*(.01)*(0.01)/500.;

load R_SA.dat; % Orthogonal SA errors from Zyla ARIMA 3x2
load I_SA.dat; % at 30 second increments
load C_SA.dat;
load R_SA_vel.dat; % uncorrelated data for SA velocity errors
load I_SA_vel.dat;
load C_SA_vel.dat;

R_SA = R_SA/1000.; % convert to km
I_SA = I_SA/1000.;
C_SA = C_SA/1000.;
R_SA_vel = R_SA_vel/1000.;
I_SA_vel = I_SA_vel/1000.;
```

```

C_SA_vel = C_SA_vel/1000.;

%----- generate simulated obs and true trajectory-----
r = rtrue;
v = vtrue;
B = Btrue;
time = 0;      % t = 0 at initial r,v

                                % generate truth obs
i = 0;
for k = 1:nsteps,
    state = [ r(1) r(2) r(3) v(1) v(2) v(3)];
    state = rk4(state,time,tau,'J4gravPlusDrag',B);
    r = [state(1) state(2) state(3)];
    v = [state(4) state(5) state(6)];
    time = time + tau;
    if rem(k,skip) == 0,
        i = i + 1;
        xobs(i) = r(1);
        yobs(i) = r(2);
        zobs(i) = r(3);
        xdotobs(i) = v(1);
        ydotobs(i) = v(2);
        zdotobs(i) = v(3);
        rmag = norm(r);      % calculate RIC unit vectors
        vmag = norm(v);
        rhat = r/rmag;
        vhat = v/vmag;
        Chat(i,1) = rhat(2)*vhat(3) - rhat(3)*vhat(2);
        Chat(i,2) = rhat(3)*vhat(1) - rhat(1)*vhat(3);
        Chat(i,3) = rhat(1)*vhat(2) - rhat(2)*vhat(1);
        lhat(i,1) = vhat(1);
        lhat(i,2) = vhat(2);
        lhat(i,3) = vhat(3);
        Rhath(i,1) = vhat(2)*Chat(i,3) - vhat(3)*Chat(i,2);
        Rhath(i,2) = vhat(3)*Chat(i,1) - vhat(1)*Chat(i,3);
        Rhath(i,3) = vhat(1)*Chat(i,2) - vhat(2)*Chat(i,1);
    end
end
disp(time);

nobs = size(xobs,2);
disp('nobs is');
disp(nn);

%-----Add SA effects-----

q = size(R_SA,1);
k = 0;
for i = 1:q,
    if rem(i,skip) == 0; % thin data by skip factor

```

```

k = k +1;
newR_SA(k) = R_SA(i);
newI_SA(k) = I_SA(i);
newC_SA(k) = C_SA(i);
newR_SA_vel(k) = R_SA_vel(i);
newI_SA_vel(k) = I_SA_vel(i);
newC_SA_vel(k) = C_SA_vel(i);
end
end

%.....add SA errors to x y z data first .....
%..... create SA error vector by dotting into RIC unit vector .....

for i = 1:nobs,
Vsa = newR_SA(i)*Rhat(i,:) + newI_SA(i)*Ihat(i,:) + newC_SA(i)*Chat(i,:);
SAx = Vsa(1);
SAy = Vsa(2); % decompose x, y, and z components
SAz = Vsa(3);
xobs(i) = xobs(i) + SAx;
yobs(i) = yobs(i) + SAy;
zobs(i) = zobs(i) + SAz;
end

%.....velocity data next .....

sigmaR_SA = .03483; % sigma of SA data file in km (calculated offline)
sigmaI_SA = .03356;
sigmaC_SA = .03166;

scale1 = sigmaVkm/sigmaR_SA;
scale2 = sigmaVkm/sigmaI_SA;
scale3 = sigmaVkm/sigmaC_SA;

newR_SA_vel = newR_SA_vel*scale1; % convert to sigma = sigmaVkm
newI_SA_vel = newI_SA_vel*scale2;
newC_SA_vel = newC_SA_vel*scale3;

for i = 1:nobs,
Vsa = newR_SA_vel(i)*Rhat(i,:) + newI_SA_vel(i)*Ihat(i,:) +
newC_SA_vel(i)*Chat(i,:);
SAx = Vsa(1);
SAy = Vsa(2); % decompose x, y, and z components
SAz = Vsa(3);
xdotobs(i) = xdotobs(i) + SAx;
ydotobs(i) = ydotobs(i) + SAy;
zdotobs(i) = zdotobs(i) + SAz;
end

%----- Create Matrix of measurement variances (WTW) -----

for i = 1:(3*nobs),

```

```

    W(i) = (1/sigmaPkm)^2.; % 1/kmeters
end

for i = ((3*nobs)+1):(6*nobs),
    W(i) = (1/sigmaVkm)^2.; % 1/kmeters/sec
end

WTW = sparse(diag(W));          % nxn diagonal matrix of 1/variances

clear W;
clear R_SA;
clear I_SA;
clear C_SA;
clear newR_SA;
clear newI_SA;
clear newC_SA;
clear R_SA_vel;
clear I_SA_vel;
clear C_SA_vel;
clear newR_SA_vel;
clear newI_SA_vel;
clear newC_SA_vel;

r = rinit;
v = vinit;
B = Binit;

%===== Main Loop =====
%-----enter main orbit estimation loop-----

```

## A.7 Horiz\_VertSA

```
% Program to plot horizontal and vertical components of SA errors
clear;

skip = 2;    % thin factor
start = 480; % where to start in data file

stop = start + 119; % stop after one hour at 60 second increments

load R_SA.dat;      %45000 secs at 30 sec increments (1500 points)
load I_SA.dat;      % data is orthogonal
load C_SA.dat;

for i = start:stop,    %one hour of data
    shortR_SA(i-(start-1)) = R_SA(i);
    shortI_SA(i-(start-1)) = I_SA(i);
    shortC_SA(i-(start-1)) = C_SA(i);
end

nsteps = size(shortR_SA,2);

k = 0;
for i = 1:nsteps,
    if rem(i,skip) == 0,
        k = k + 1;
        newR_SA(k) = shortR_SA(i);
        newI_SA(k) = shortI_SA(i);
        newC_SA(k) = shortC_SA(i);
        isave(k) = k;
    end
end

subplot(121),
    plot(newI_SA,newC_SA,'-',newI_SA,newC_SA,'go');
    title('Horozontal SA Error (m)')
subplot(122),
    plot(isave,newR_SA,'-',isave,newR_SA,'go')
    title('Vertical SA Error (m)')
    xlabel('Time in Minutes')
subplot(111)
```

## A.8 VecSum\_SA

```
% Program to take the vector sum of 4 SA error files
% assuming simulation orientation
clear;

xhat = [1 0 0]; %intrack direction (GPS1)
yhat = [0 1 0]; %crosstrack
zhat = [0 0 1]; %radial (GPS4)

a1 = cos(pi/6.); % 30 deg
a2 = sin(pi/6.);

a3 = cos(pi/9.); % 20 deg
a4 = sin(pi/9.);

V1 = a3*xhat + a4*zhat; % assumed unit vector orientation of GPS SV 1
V2 = -a2*a3*xhat + a1*a3*yhat + a4*zhat; % GPS SV 2
V3 = -a2*a3*xhat - a1*a3*yhat + a4*zhat; % GPS SV 3

load patchT1.dat; % SA error file patched together to
load patchT2.dat; % simulate satellite switches
load patchT3.dat;
load patchT4.dat;

nsteps = size(patchT1,1);

for i = 1:nsteps,
    g1 = patchT1(i)*V1; % SA error vector in direction of GPS 1
    g2 = patchT2(i)*V2;
    g3 = patchT3(i)*V3;
    g4 = patchT4(i)*zhat; % GPS SV 4 assumed overhead

    gvector = g1 + g2 + g3 + g4;

    Isave(i) = gvector(1); % decompose to orthogonal components
    Csave(i) = gvector(2);
    Rsave(i) = gvector(3);
    isave(i) = i;
end

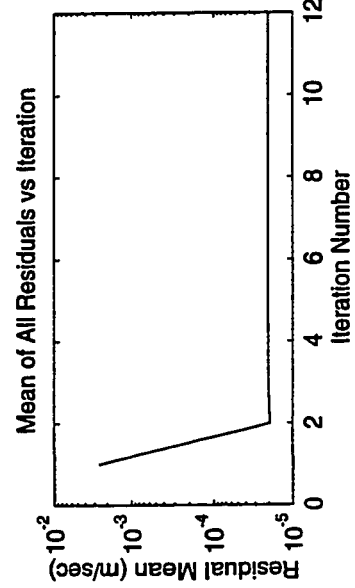
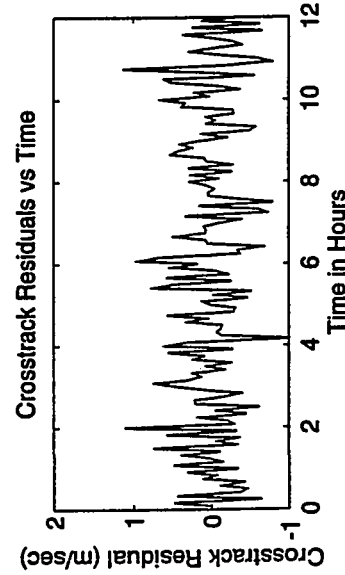
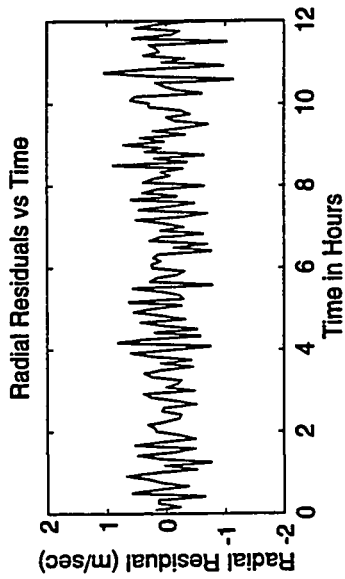
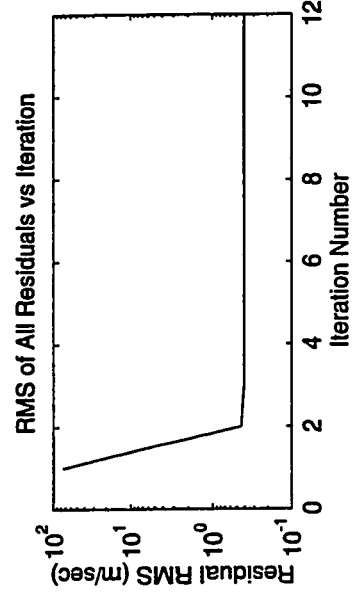
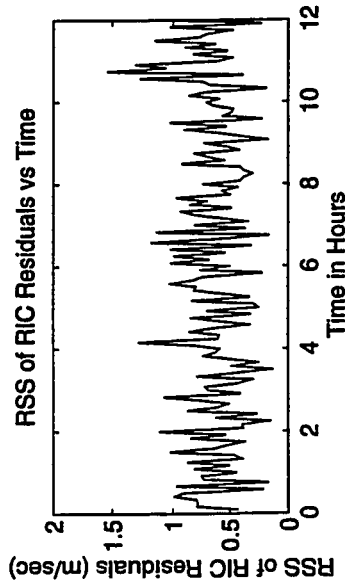
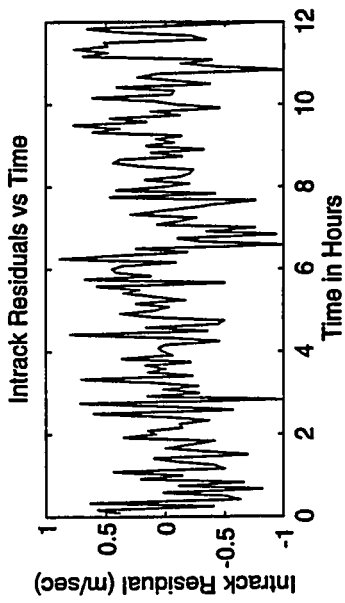
IT = Isave';
CT = Csave';
RT = Rsave';

save I_SA.dat IT -ascii;
save C_SA.dat CT -ascii;
save R_SA.dat RT -ascii;
```

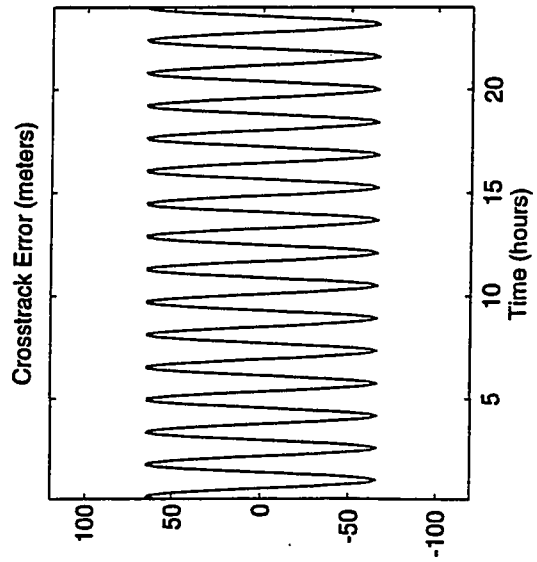
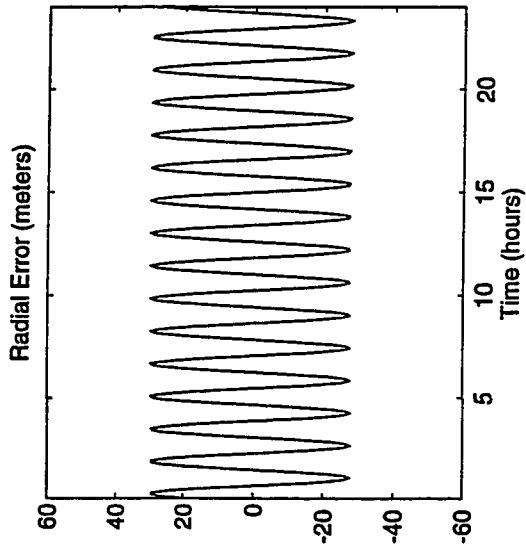
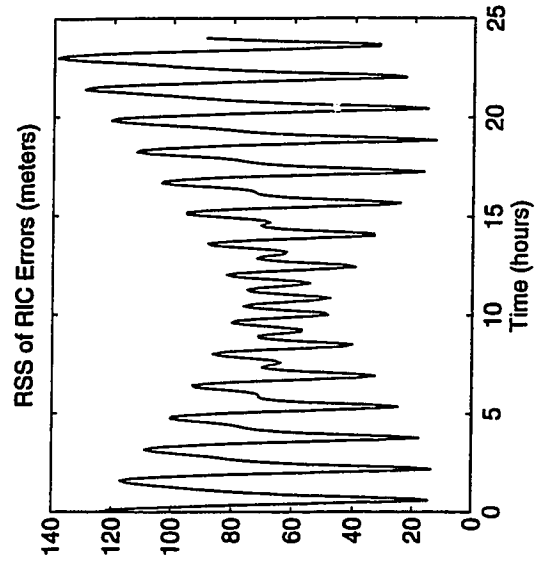
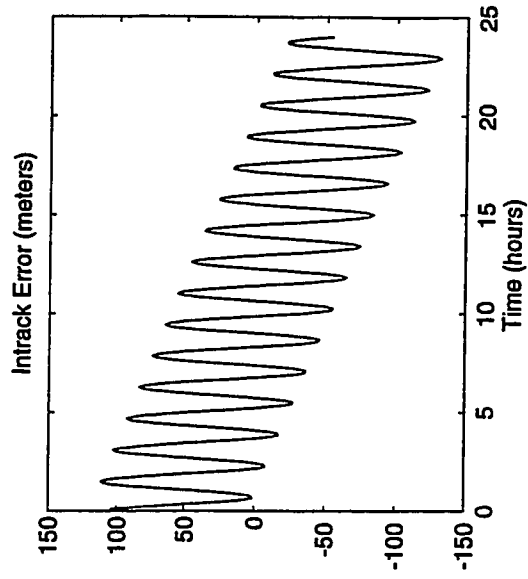
```
subplot(121),  
  plot(Isave,Csave,'-',Isave,Csave,'go');  
  title('SA Horizontal PR Error')  
subplot(122),  
  plot(isave, Rsave)  
  title('SA Vertical PR Error')  
subplot(111)
```

**Appendix B**  
**Graphical Results**

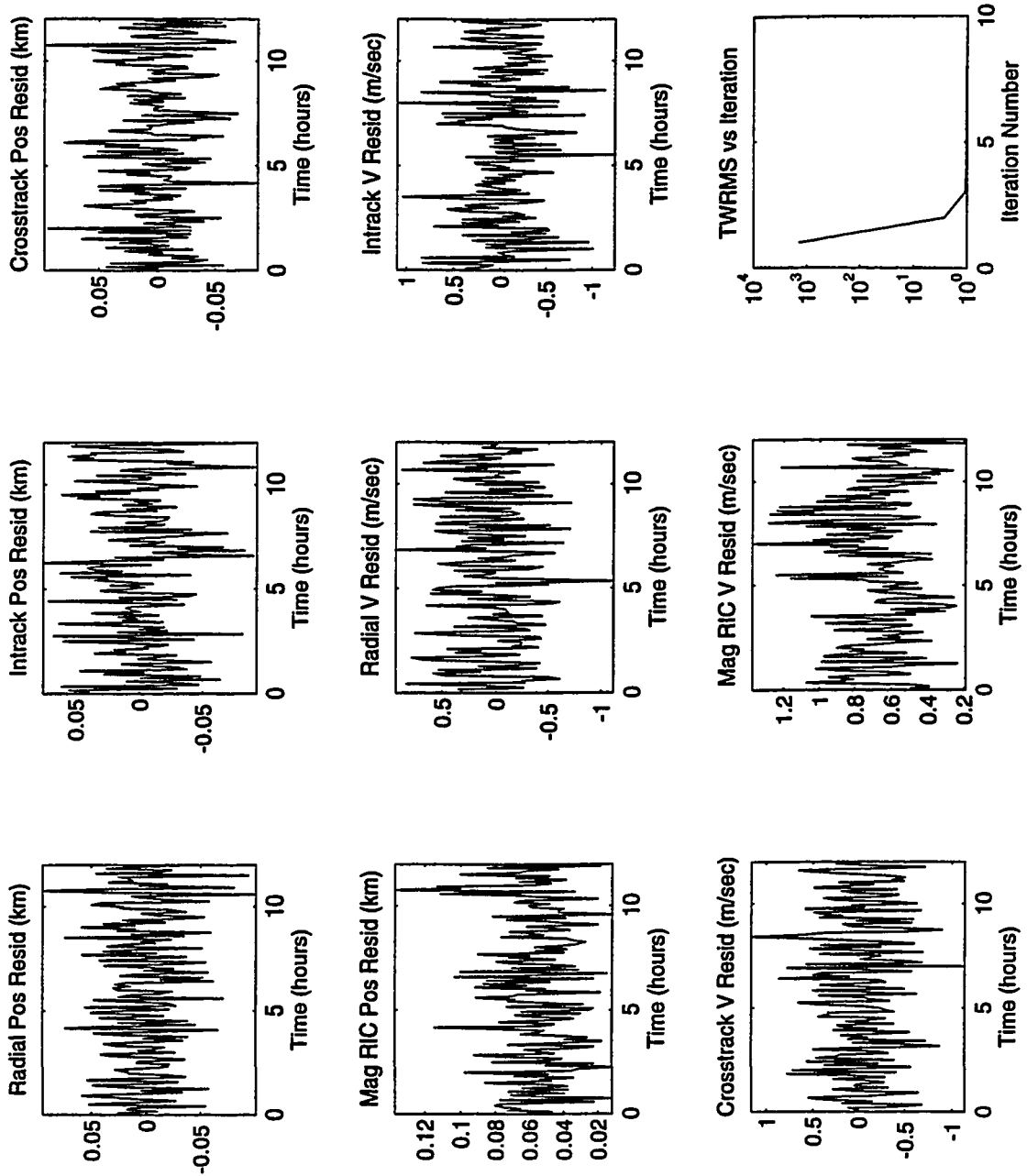




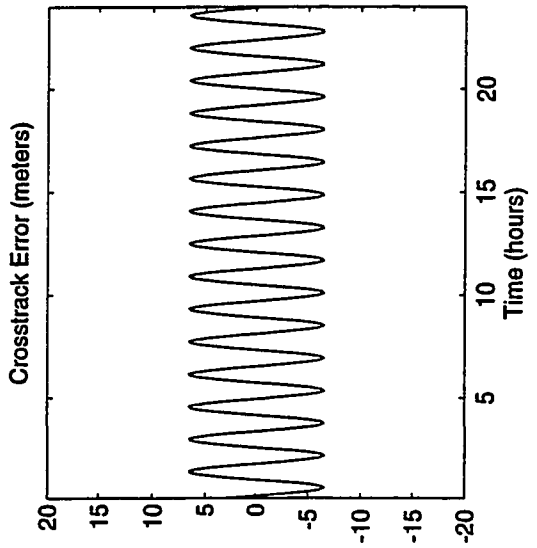
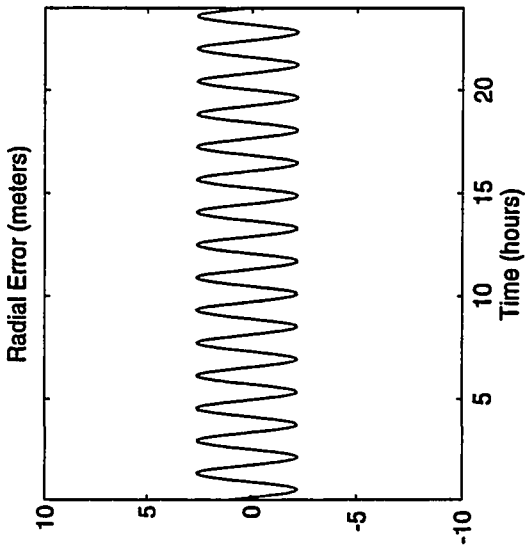
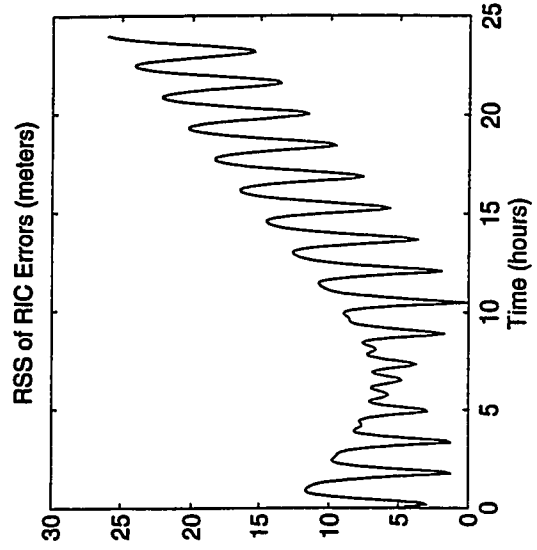
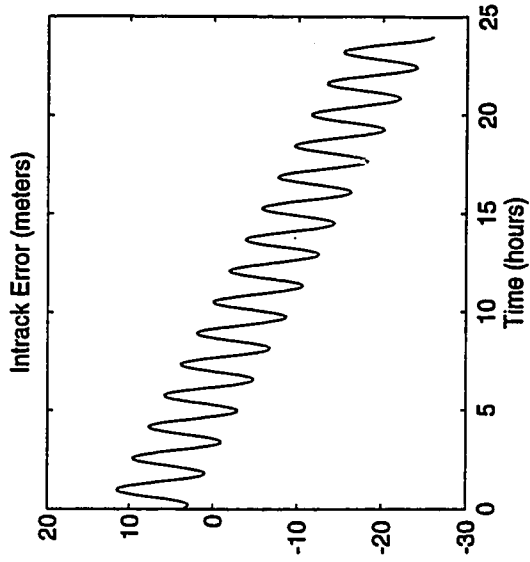
Final Iteration Data Simulated Velocity Fit with Gaussian Errors



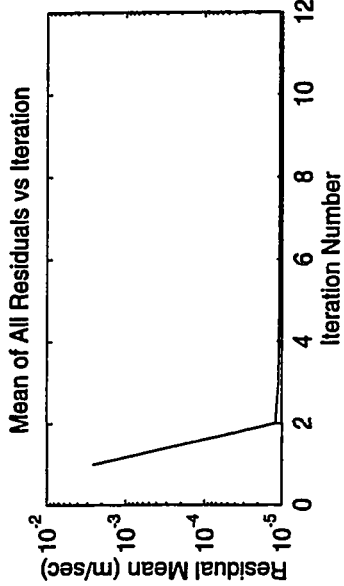
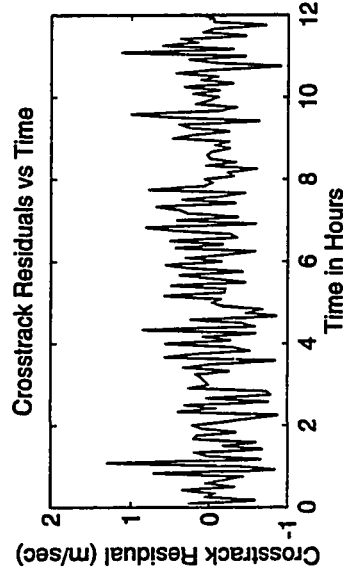
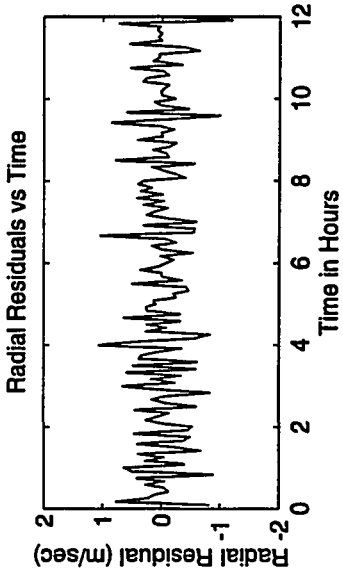
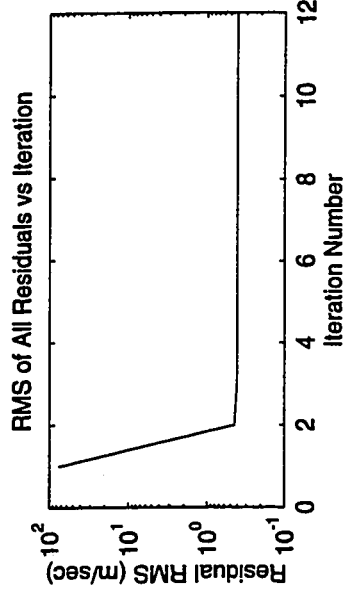
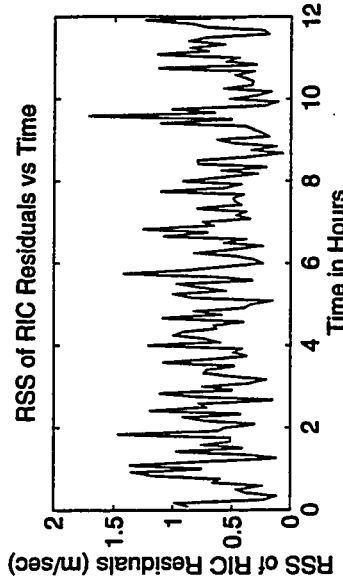
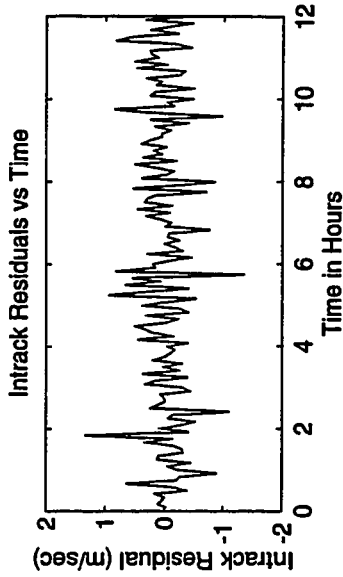
Final Solved-For State Propagated vs Truth, Sim Velocity Fit, Gaussian



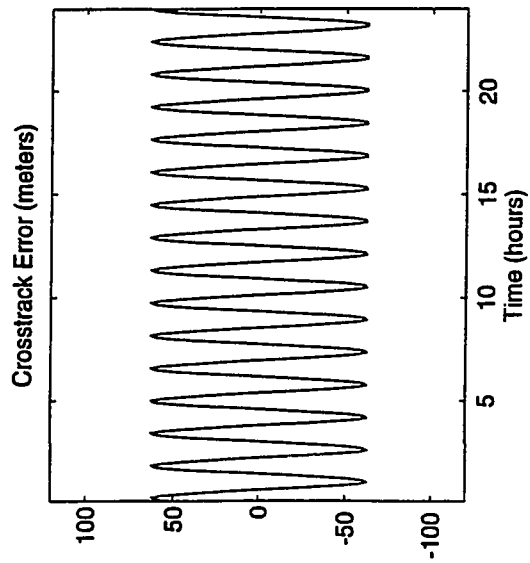
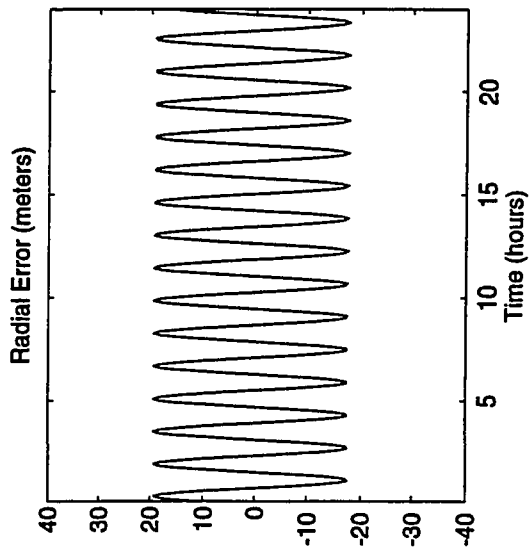
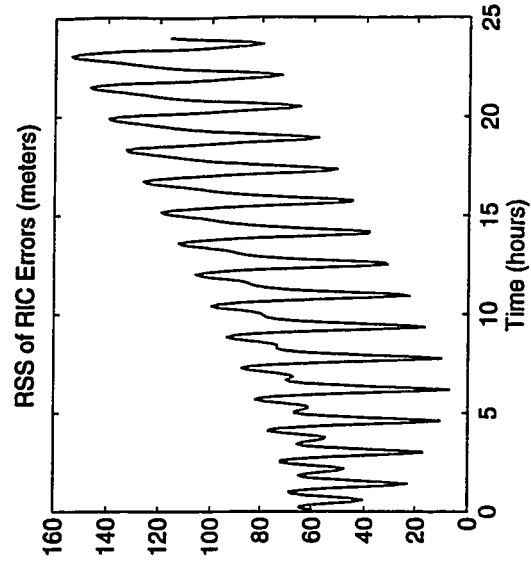
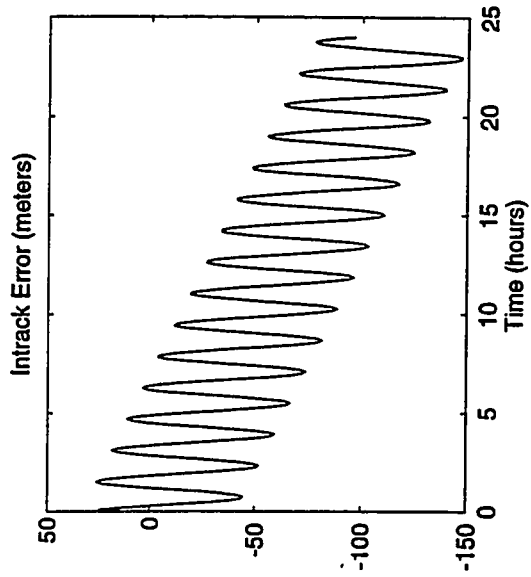
Final Iteration Data Simulated Position/Velocity Fit, Gaussian Errors



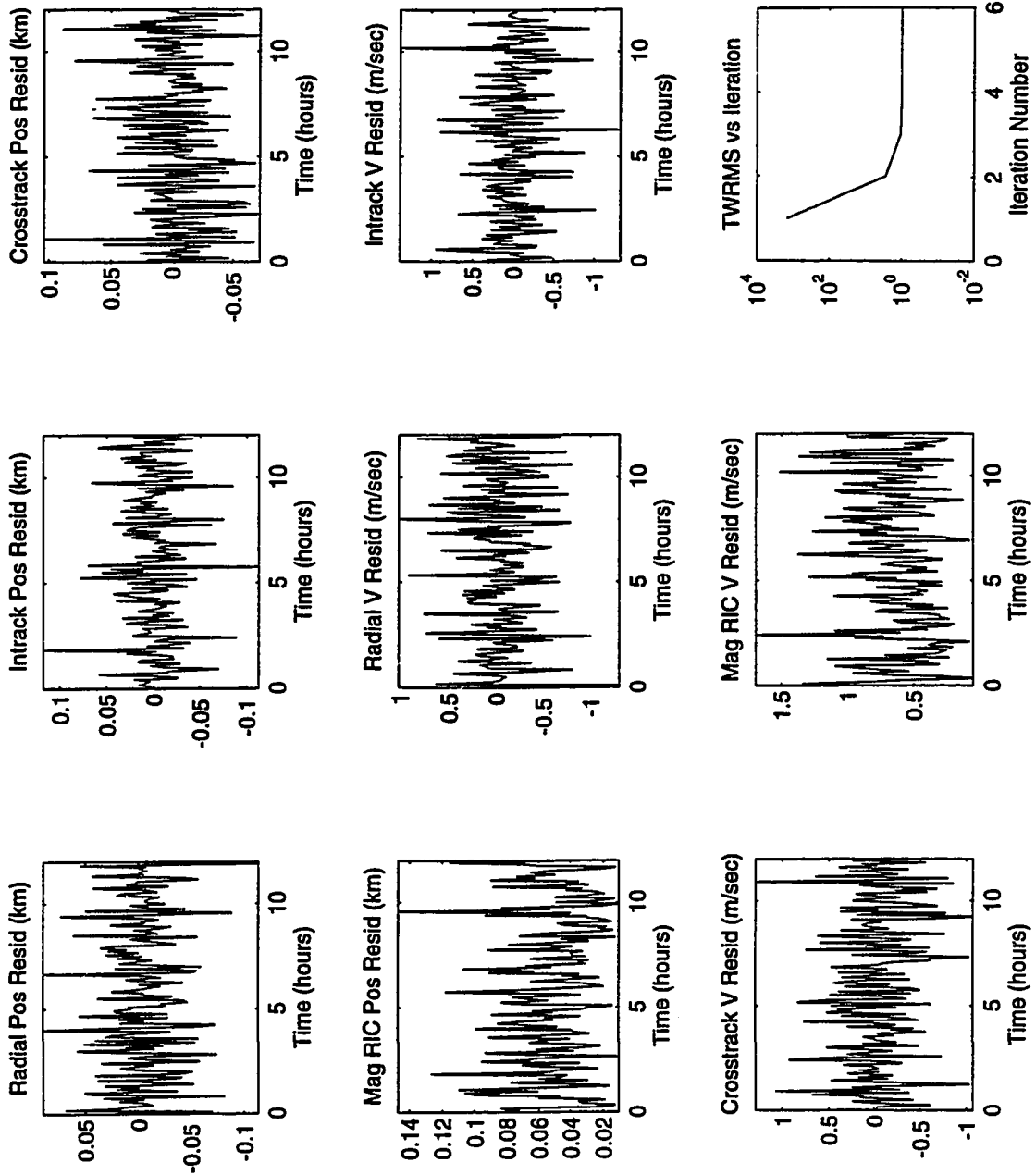
Final Solved-For State Propagated vs Truth, Sim Position/Velocity Fit, Gaussian



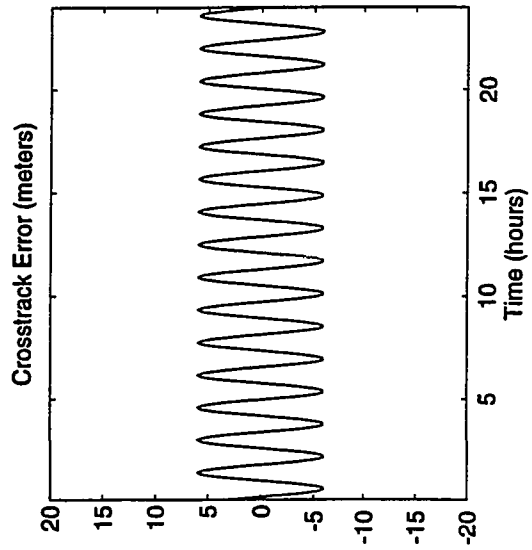
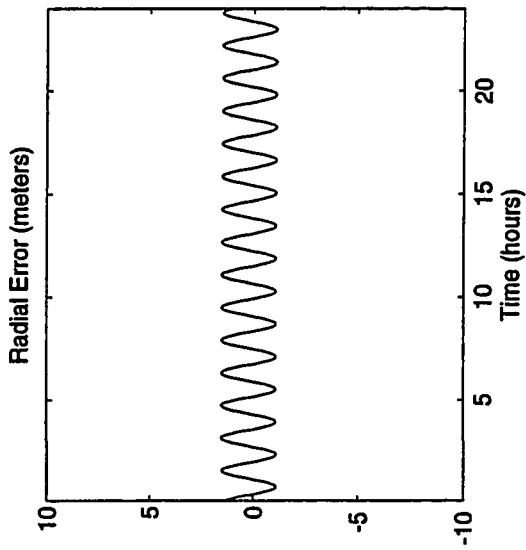
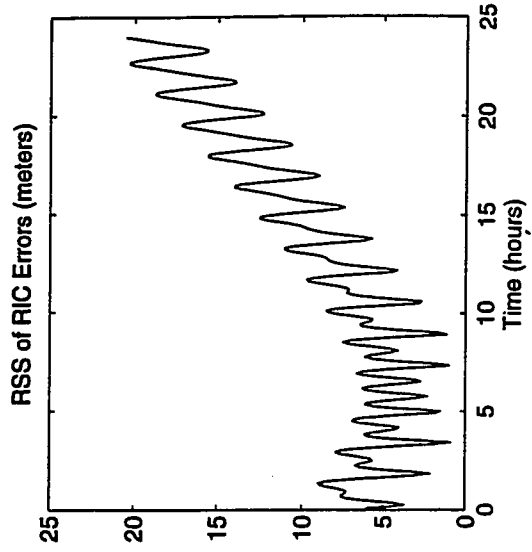
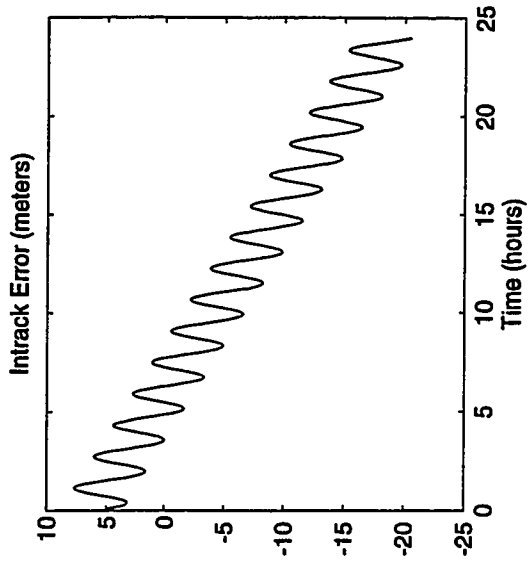
Final Iteration Data Simulated Velocity Fit with SA



Final Solved-For State Propagated vs Truth, Sim Velocity Fit, SA

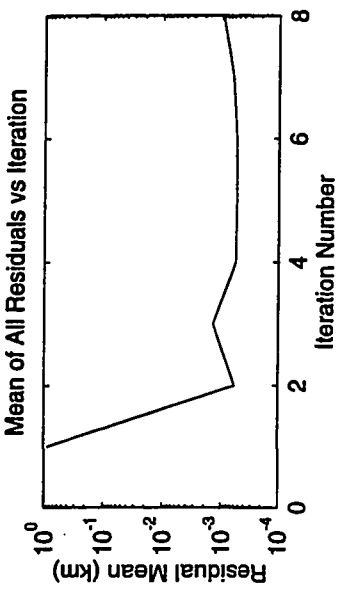
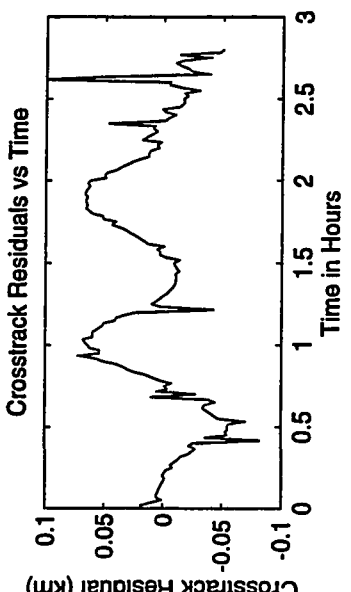
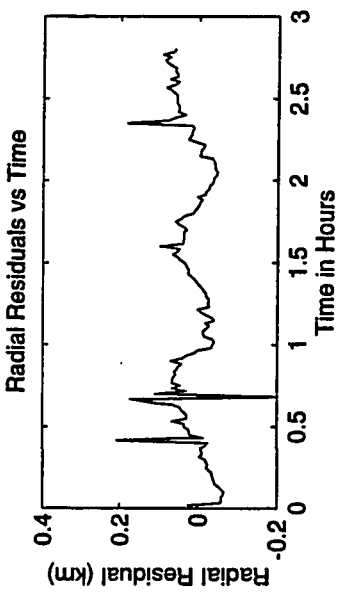
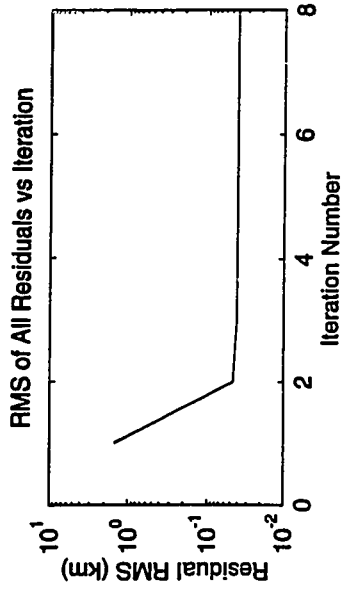
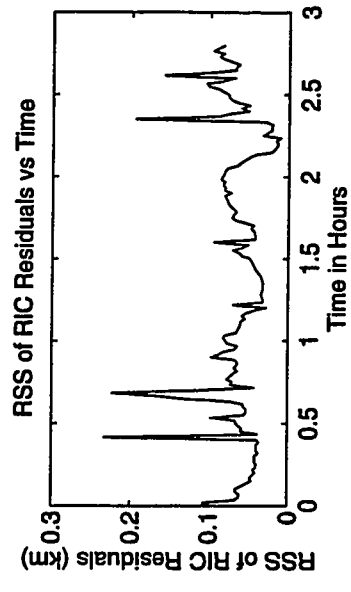
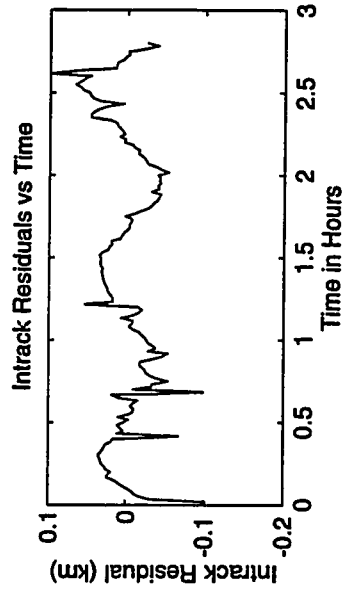


Final Iteration Data Simulated Position/Velocity Fit with SA

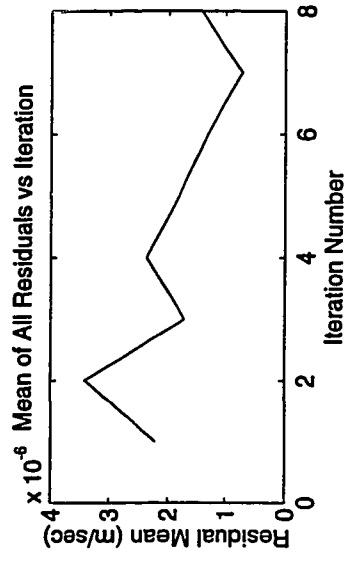
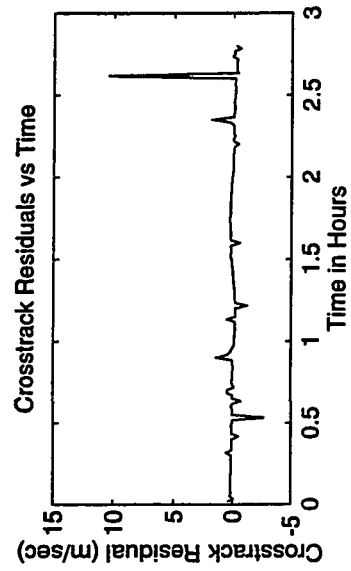
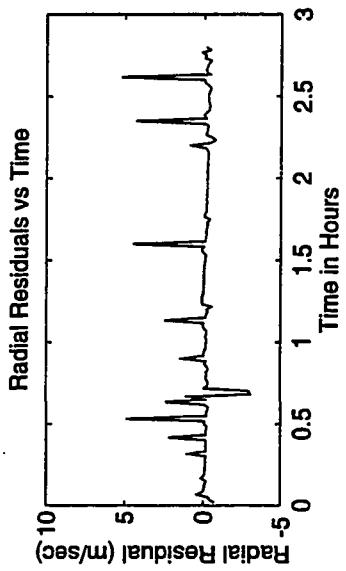
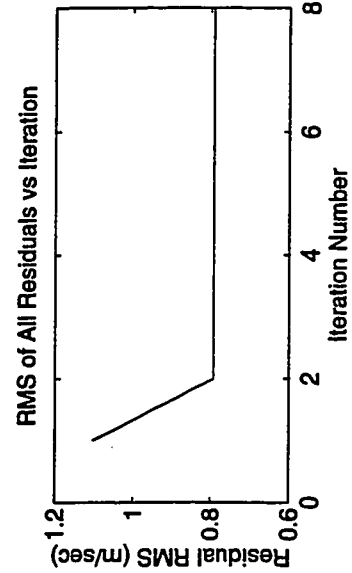
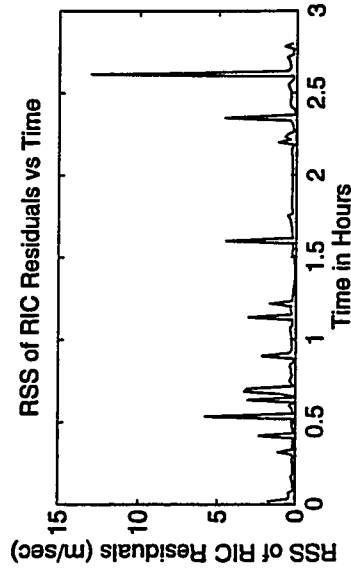
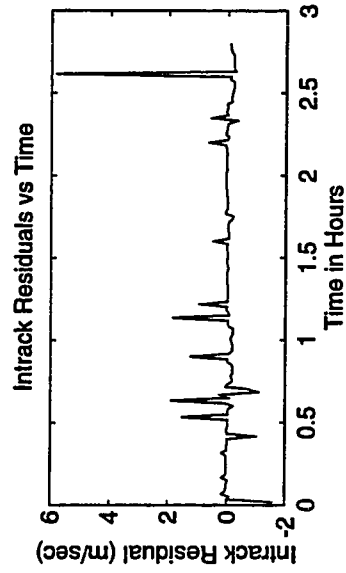


Final Solved-For State Propagated vs Truth, Sim Position/Velocity Fit, SA

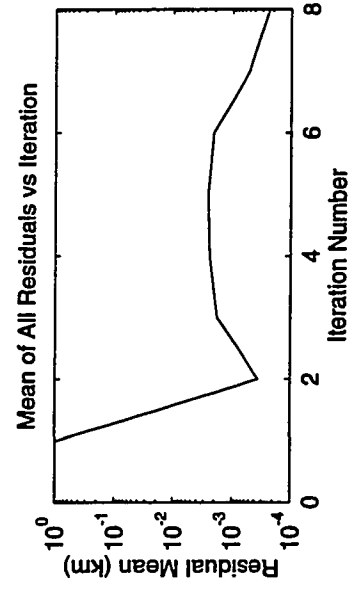
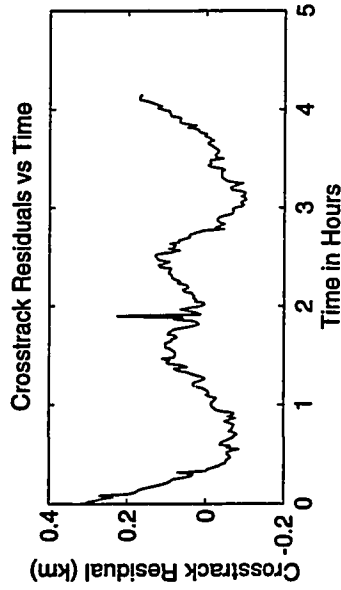
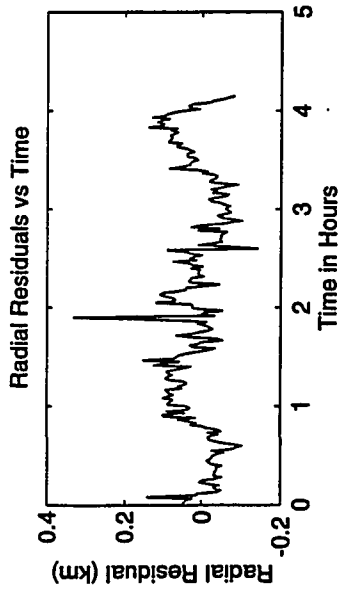
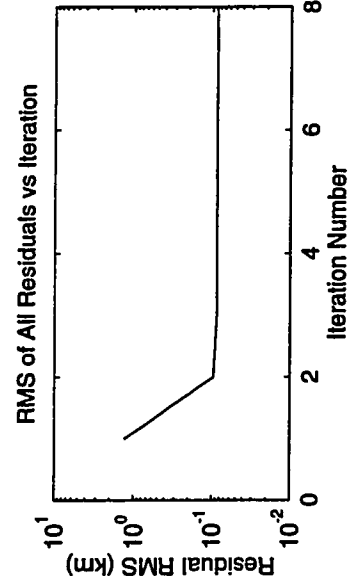
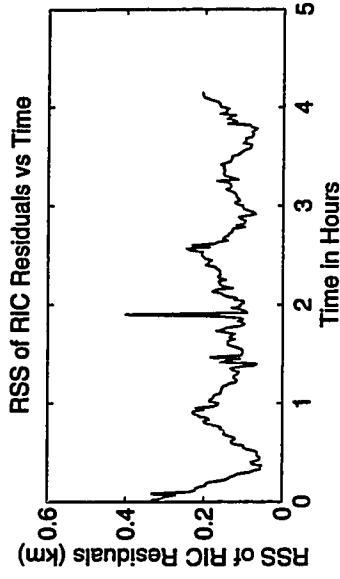
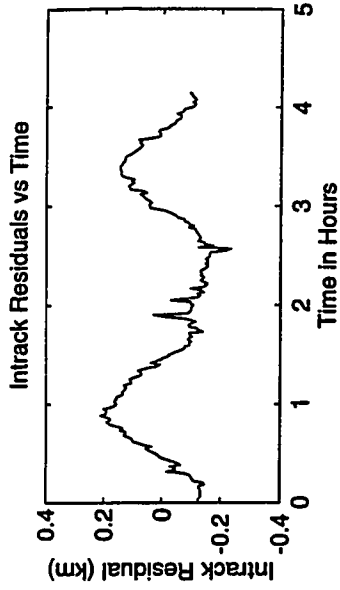




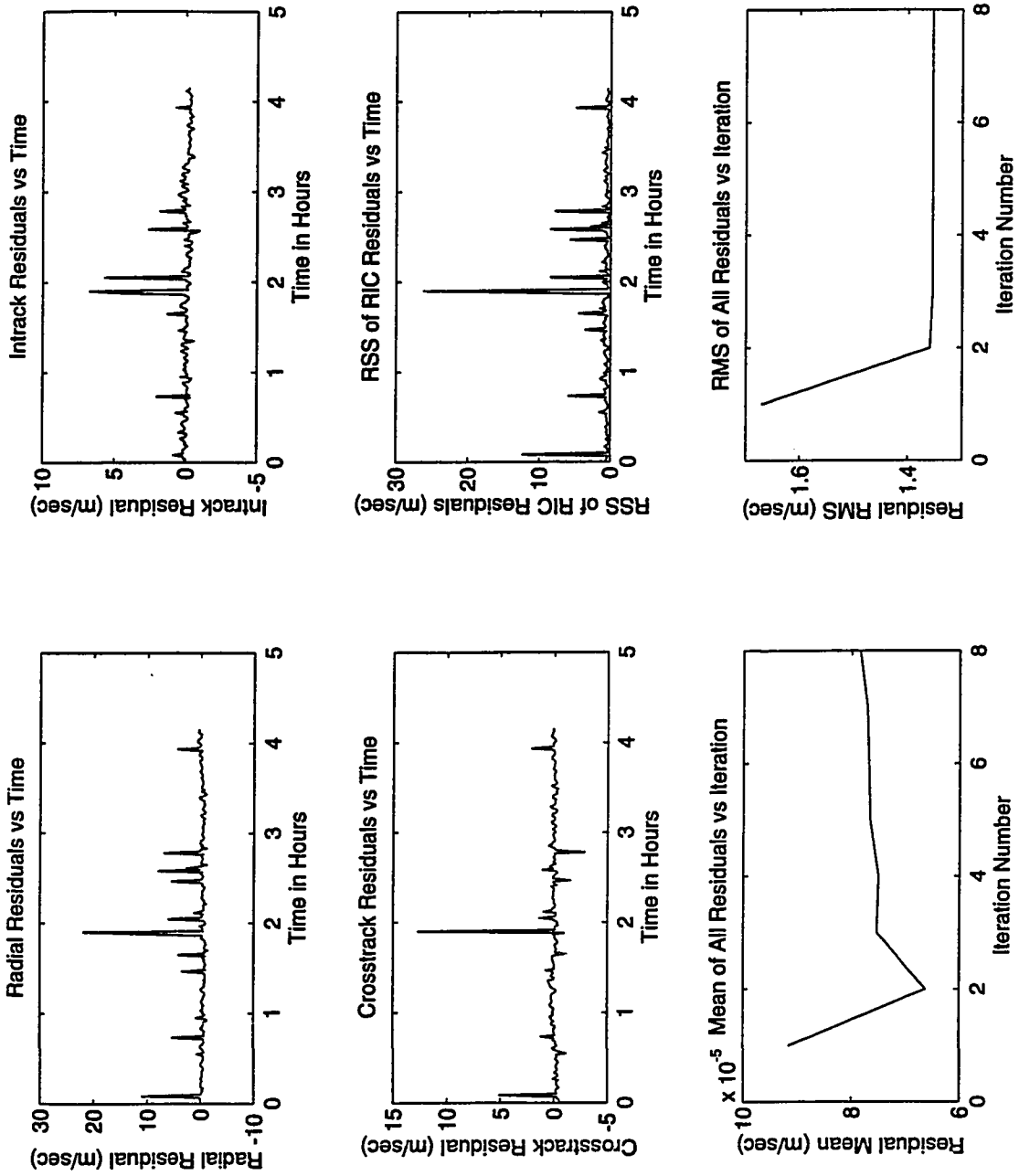
Final Iteration Data EUVE Position Fit, no SA



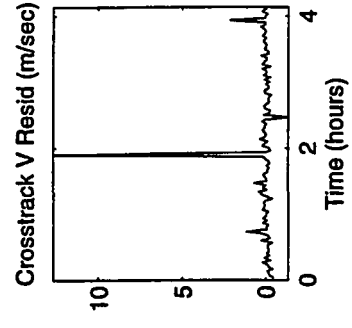
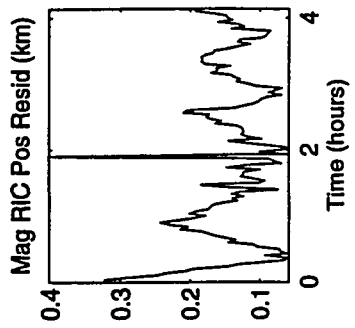
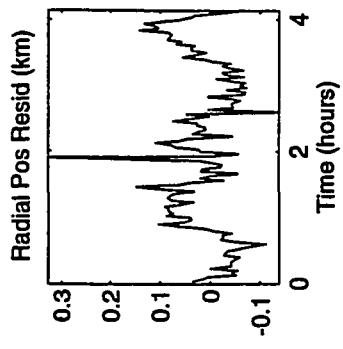
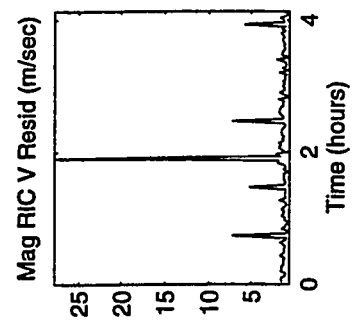
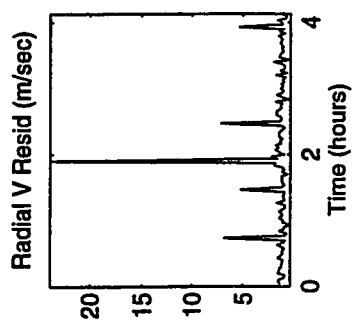
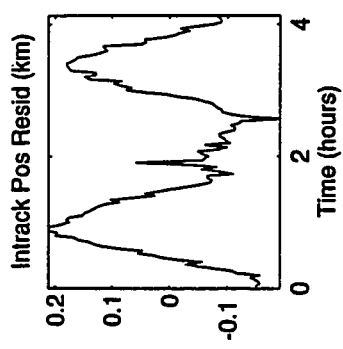
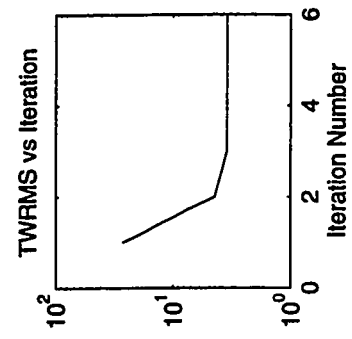
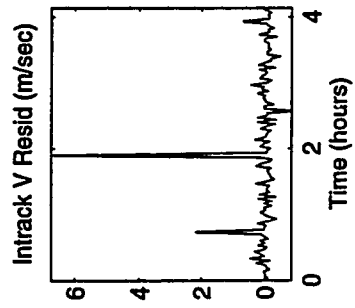
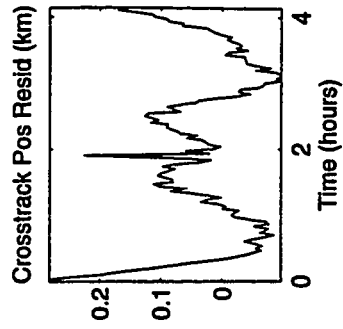
Final Iteration Data EUVE Velocity Fit, no SA



Final Iteration Data EUVE Position Fit, with SA



Final Iteration Data EUVE Velocity Fit, with SA



Final Iteration Data EUVE Position/Velocity Fit, with SA