Air Force Institute of Technology

AFIT Scholar

3-2000

Navigation of Satellite Clusters

Jeffrey S. Davis

Follow this and additional works at: https://scholar.afit.edu/etd

Part of the Astrodynamics Commons

Recommended Citation

NAVIGATION OF SATELLITE CLUSTERS

THESIS

Jeffrey S. Davis, Captain, USAF

AFIT/GSO/ENY/00M-01

20000803 146

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GSO/ENY/00M-01

NAVIGATION OF SATELLITE CLUSTERS

THESIS

Presented to the Faculty of the School of Engineering and Management

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Space Operations

Jeffrey S. Davis, B.S.

Captain, USAF

March 2000

NAVIGATION OF SATELLITE CLUSTERS

Jeffrey S. Davis, B.S.
Captain, USAF

Approved:

William E. Wiesel
_____
Dr. William Wiesel, Ph. D. (Chairman)

6 Jan 00
_____
date

(signature)
_____
Captain Gregory Agnes, Ph. D.

09 Mar 00
_____
date

Steven M Tragesser
_____
Dr. Steven Tragesser, Ph. D.

06 MAR 00
_____
date

Acknowledgments

I would like to express my gratitude and appreciation to Dr. William Wiesel for his outstanding support over the duration of this effort. His assistance with my computer programs and understanding of the problem proved invaluable. However, I'm mostly thankful for never having left his office in despair. I would also like to thank Capt Greg Agnes and Dr. Steven Tragesser for their insights as well.

My heartfelt thanks and love belong to my wife Kelly, daughter Brooke and son Austin. Their understanding, support and willingness to make sacrifices over the past 18 months have made this a positive experience in our lives.

Jeffrey S. Davis

Abstract

The relative position determination of a cluster of satellites in near circular orbit was investigated in previous thesis work. The purpose of this thesis is to extend the concept to cover absolute position determination. A Bayes filter is used for the estimator with dynamics based on the two-body problem extended to account for J_2 perturbations. Measurements consist of combining Global Positioning System (GPS) data for each satellite and range data between the satellites. Simulations were conducted investigating the accuracy obtainable when combining the measurements for input into the filter. Performance results consist of comparing the magnitude of the true error to the filter covariance as a function of time. True errors are also compared to minimum accuracy requirements for a space-based radar. The filter encountered numerical difficulties due to the extreme accuracy requirements and proved unsuccessful in providing usable estimates. The results suggest separating the absolute and relative problems.

Table of Contents

List of Figures

NAVIGATION OF SATELLITE CLUSTERS

I. Introduction

Background

As the development of space-based radar technologies permits shifting missions such as Airborne Warning and Control System (AWACS) and Joint Surveillance and Targeting Attack Radar System (Joint-STARS) surveillance functions to the arena of space, the navigation and control of satellite clusters has become a prime area of interest. The Space Vehicles Directorate of the Air Force Research Laboratory (AFRL) is one agency currently studying how a cluster of satellites each acting as an element of a radar antenna might perform such a mission. The TechSat 21 project envisions a cluster of satellites working together as a single *virtual satellite* while performing various functions, such as a space-based radar (Figure 1). Research is needed to determine how to fly the cluster in formation, so it acts like a single satellite. Before they can perform *formation flying*, the positions of the satellites relative to one another must be known with some degree of accuracy. Furthermore, in order to perform a variety of spaced-based radar applications, the positions should be known to within one tenth of a wavelength of the radar frequency being used; potentially to 1 centimeter (cm). The determination of this value can be found in Appendix A.
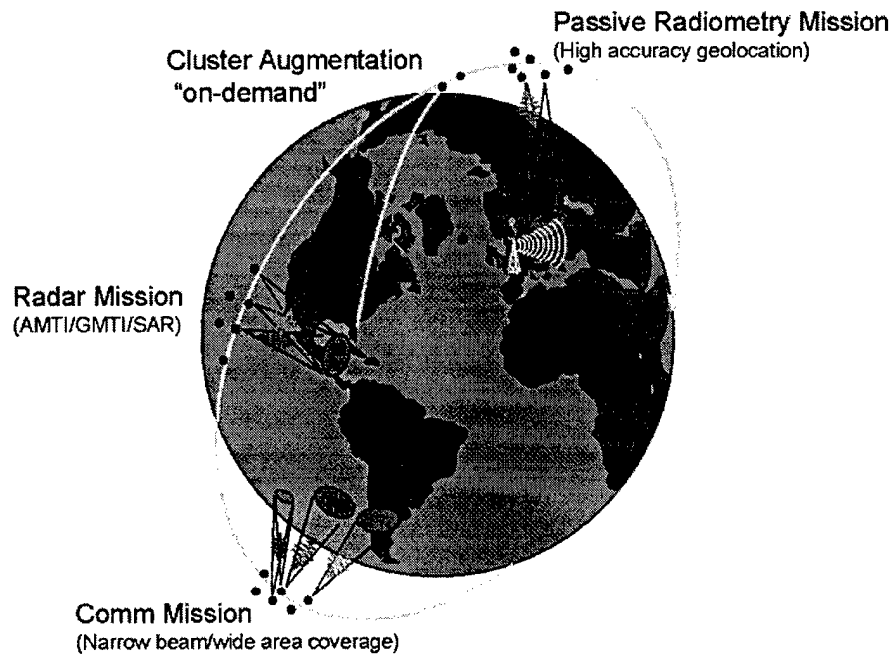
Figure 1. Techsat 21 Missions (13: n. pag.)

Using the Global Positioning System (GPS) to determine the position of the satellites with respect to each other offers a possible solution. Spaced-based GPS receivers may achieve improvement over the accuracy obtained from terrestrial receivers. The GPS constellation of 24 satellites orbits the earth at approximately 20,000 km providing 100 percent coverage for a satellite cluster operating at typical space-based radar altitudes. This may enhance performance of the receivers due to improved satellite viewing geometry. The absence of tropospheric delays offers an additional improvement in performance. Results of postprocessed experiments using differential GPS showed baseline accuracies of approximately 5-10 cm (5:249). Although this level of accuracy may not be achievable with stand-alone spaceborne processing, accurate modeling and a

2

well designed filter should make 1 meter (m) level accuracies possible (5:249). Although real time accuracy for authorized users of Precise Positioning Service (PPS) signals is currently about 5 - 10 meters (8:D-2), this study will assume 1 - 3 meter accuracy will be available for GPS data. However, even at 1 m level accuracy, performance does not meet the required centimeter level of accuracy for some radar applications.

Several previous Air Force Institute of Technology theses studied the concept of using a recursive filter for relative position determination among a cluster of satellites without the use of GPS (4; 9; 14). They used a U-D covariance factorization Kalman filter that operated on range data between the satellites determined from synchronized clock pulses. The relative position error approached a value of approximately 3 cm. The position is relative because only the distance between the satellites is known, not the direction.

Combining the centimeter level accuracy of the range data with the meter level accuracy of GPS may provide a solution for the absolute positions of the satellites to the required accuracy. Applying estimation techniques, in the form of a Bayes filter, to this approach is the focus of this investigation.

Objectives

The purpose, and primary objective, of this investigation was to determine the possible accuracies for position determination among a cluster of satellites using a Bayes Filter. Several other objectives were accomplished in order to achieve the primary objective.

First, simulating the orbits of the satellite cluster was necessary. Although future analysis should be accomplished for a full cluster (8 - 16 satellites), only two were used in this study. This was done under the assumption that what could be performed with two

could be extended to many. The orbits were determined using the two-body equations of motion expanded to include the effects of the nonspherical earth.

Next, a truth model was built to output the true state as well as the measurement data over selected time intervals for input into the filter. The truth model outputs consisted of the range data between satellites, and the GPS determined positions for each of the satellites.

Finally, a Bayes filter was written to output an estimated state. The critical elements of the estimated state were compared to the true state to determine the filter's level of performance.

II. Methodology

Satellite Orbit Selection

The first step towards the eventual development of a truth model is selecting the

desired orbital parameters for the satellite cluster. The criteria is for the satellites to be

approximately 500 meters apart, at their closest, and at an orbital altitude of about 1000

nautical miles (1852 km).
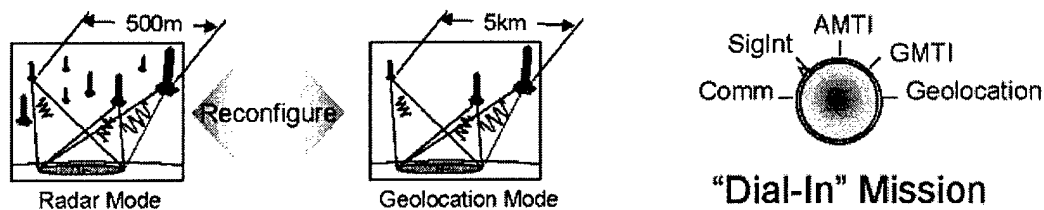
Figure 2. Spacing Criteria for Different Modes. (13:n. pag.)

In order to maintain the cluster integrity, and not drift apart, each of the satellites are

required to have the same orbital period. From Kepler's laws, the orbital period of an

elliptical orbit is a function of the semi-major axis of the orbit (1:33). Therefore, both of

the satellites used in the model had the same semi-major axis. To keep things simple, the

two satellites were placed in the same orbital plane of a near circular orbit with different

perigee times used to maintain the desired separation. Although safety reasons might

prevent placing any of the cluster's satellites this close when within the same along-track

5

plane, it is not a concern for this study. The velocities of the satellites were calculated using

$$v_{cir} = \sqrt{\frac{\mu}{r}} \qquad (1)$$

where

v_{cir} = orbital velocity (circular)

μ = 3.986005X10^14 km^3/s^2 (the earth's gravitational parameter)

r = orbit radius (km)

The time of perigee passage for one of the satellites was then adjusted to create a 500 meter interval. The remaining orbital elements were arbitrarily chosen and entered into an existing program to determine the initial position and velocity vectors for the two satellites.

Propagating the Orbits

With the initial positions and velocities known, the next step was to develop a program to propagate each satellite through its orbit. The orbits were based on two-body orbital dynamics expanded to include the effects of the nonspherical Earth known as J$_2$ perturbations. Other perturbations, such as atmospheric drag, were neglected in part due to their negligible effects at the altitude of interest, 1852 km, and partly to simplify the equations. The desired output from the orbit propagator program was a state vector for

6

each satellite which included the position and the velocity vectors as follows

$$X = \begin{pmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}$$

(2)

The x, y, and z positions are with respect to an earth-centered inertial reference frame and defined in terms of distance units (DU), where 1 DU = 1678.135 km, or the radius of the Earth. To obtain the state vector, the equations of motion to describe the orbits were written and integrated. The equations of motion were just the time rate of change of the state vector,

$$\dot{X} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix}$$

(3)

7

The equations of motion were derived from the gravitational potential function (1:419):

$$V = -\frac{\mu}{\bar{r}} + \frac{\mu r_e^2 J_2}{2\bar{r}^3}\left(3\frac{z^2}{\bar{r}^2} - 1\right) \qquad (4)$$

where

μ=3.986005X10^14 km^3/s^2 (the earth's gravitational parameter)

r$_e$=6378.137 km (the earth's radius)

J2=0.00108263

The acceleration terms were found by setting the following equalities:

$$\ddot{x} = -\frac{\partial V}{\partial x} \qquad (5)$$

$$\ddot{y} = -\frac{\partial V}{\partial y} \qquad (6)$$

$$\ddot{z} = -\frac{\partial V}{\partial z} \qquad (7)$$

Solving each of these led to the following acceleration terms (1:421-422):

$$\ddot{x} = -\frac{\mu x}{\bar{r}^3}\left[1 - J_2\frac{3}{2}\left(\frac{r_e}{\bar{r}}\right)^2\left(5\frac{z^2}{\bar{r}^2} - 1\right)\right] \qquad (8)$$

$$\ddot{y} = -\frac{\mu y}{\bar{r}^3}\left[1 - J_2\frac{3}{2}\left(\frac{r_e}{\bar{r}}\right)^2\left(5\frac{z^2}{\bar{r}^2} - 1\right)\right] \qquad (9)$$

$$\ddot{z} = -\frac{\mu z}{\bar{r}^3}\left[1 + J_2\frac{3}{2}\left(\frac{r_e}{\bar{r}}\right)^2\left(3 - 5\frac{z^2}{\bar{r}^2}\right)\right] \qquad (10)$$

where

$$\bar{r} = \sqrt{x^2 + y^2 + z^2}$$

Along with the equations of motion, the state transition matrix, Φ, was used to propagate the differentials of the state as a function of time. The Φ matrix satisfies the equation

$$\dot{\Phi}(t,t_0) = A(t) \; \Phi(t,t_0) \tag{11}$$

where $A(t) = \dfrac{\partial \dot{X}}{\partial X}$, and was obtained by numerically integrating the equations of variation in parallel with the equations of motion. The equations of variation account for possible variations in the desired orbit and are derived by solving the matrix

$$\frac{\partial \dot{X}}{\partial X} = \begin{bmatrix} \dfrac{\partial V_x}{\partial x} & \dfrac{\partial V_x}{\partial y} & \dfrac{\partial V_x}{\partial z} & \dfrac{\partial V_x}{\partial \dot{x}} & \dfrac{\partial V_x}{\partial \dot{y}} & \dfrac{\partial V_x}{\partial \dot{z}} \\[2mm] \dfrac{\partial V_y}{\partial x} & \dfrac{\partial V_y}{\partial y} & \dfrac{\partial V_y}{\partial z} & \dfrac{\partial V_y}{\partial \dot{x}} & \dfrac{\partial V_y}{\partial \dot{y}} & \dfrac{\partial V_y}{\partial \dot{z}} \\[2mm] \dfrac{\partial V_z}{\partial x} & \dfrac{\partial V_z}{\partial y} & \dfrac{\partial V_z}{\partial z} & \dfrac{\partial V_z}{\partial \dot{x}} & \dfrac{\partial V_z}{\partial \dot{y}} & \dfrac{\partial V_z}{\partial \dot{z}} \\[2mm] \dfrac{\partial \dot{V}_x}{\partial x} & \dfrac{\partial \dot{V}_x}{\partial y} & \dfrac{\partial \dot{V}_x}{\partial z} & \dfrac{\partial \dot{V}_x}{\partial \dot{x}} & \dfrac{\partial \dot{V}_x}{\partial \dot{y}} & \dfrac{\partial \dot{V}_x}{\partial \dot{z}} \\[2mm] \dfrac{\partial \dot{V}_y}{\partial x} & \dfrac{\partial \dot{V}_y}{\partial y} & \dfrac{\partial \dot{V}_y}{\partial z} & \dfrac{\partial \dot{V}_y}{\partial \dot{x}} & \dfrac{\partial \dot{V}_y}{\partial \dot{y}} & \dfrac{\partial \dot{V}_y}{\partial \dot{z}} \\[2mm] \dfrac{\partial \dot{V}_z}{\partial x} & \dfrac{\partial \dot{V}_z}{\partial y} & \dfrac{\partial \dot{V}_z}{\partial z} & \dfrac{\partial \dot{V}_z}{\partial \dot{x}} & \dfrac{\partial \dot{V}_z}{\partial \dot{y}} & \dfrac{\partial \dot{V}_z}{\partial \dot{z}} \end{bmatrix} \tag{12}$$

Calculating the partial derivatives will show A(t) to have the following form (15:78):

$$A(t) = \begin{pmatrix} \phi & I \\ A_{21} & \phi \end{pmatrix} \qquad (13)$$

where

ϕ = 3 X 3 null matrix

I = identity matrix

The A_{21} elements of $A(t)$ are as follows:

$$\frac{\partial \dot{V}_x}{\partial x} = \mu \left[\frac{3 x^2 - \bar{r}^2}{\bar{r}^5} \right] + \frac{3}{2} J_2 \mu r_e^2 \left[\frac{5 \bar{r}^2 z^2 - 35 x^2 z^2}{\bar{r}^9} - \frac{\bar{r}^2 - 5 x^2}{\bar{r}^7} \right] \qquad (14)$$

$$\frac{\partial \dot{V}_x}{\partial y} = \frac{3 \mu x y}{\bar{r}^5} + \frac{15}{2} J_2 \mu r_e^2 \left[\frac{- 7 x y z^2}{\bar{r}^9} + \frac{x y}{\bar{r}^7} \right] \qquad (15)$$

$$\frac{\partial \dot{V}_x}{\partial z} = \frac{3 \mu x z}{\bar{r}^5} + \frac{15}{2} J_2 \mu r_e^2 \left[\frac{2 x z \bar{r}^2 - 7 x z^3}{\bar{r}^9} + \frac{x z}{\bar{r}^7} \right] \qquad (16)$$

$$\frac{\partial \dot{V}_y}{\partial x} = \frac{3 \mu x y}{\bar{r}^5} + \frac{15}{2} J_2 \mu r_e^2 \left[\frac{- 7 x y z^2}{\bar{r}^9} + \frac{x y}{\bar{r}^7} \right] \qquad (17)$$

$$\frac{\partial \dot{V}_y}{\partial y} = \mu \left[\frac{3 y^2 - \bar{r}^2}{\bar{r}^5} \right] + \frac{3}{2} J_2 \mu r_e^2 \left[\frac{5 z^2 \bar{r}^2 - 35 y^2 z^2}{\bar{r}^9} - \frac{\bar{r}^2 - 5 y^2}{\bar{r}^7} \right] \qquad (18)$$

$$\frac{\partial \dot{V}_y}{\partial z} = \frac{3 \mu y z}{\bar{r}^5} + \frac{15}{2} J_2 \mu r_e^2 \left[\frac{2 y z \bar{r}^2 - 7 y z^3}{\bar{r}^9} + \frac{y z}{\bar{r}^7} \right] \qquad (19)$$

10

$$\frac{\partial \dot{V}_z}{\partial x} = \frac{3\mu xz}{\bar{r}^5} + \frac{15}{2} J_2 \mu r_e^2 \left[\frac{2xz\bar{r}^2 - 7xz^3}{\bar{r}^9} + \frac{xz}{\bar{r}^7} \right] \qquad (20)$$

$$\frac{\partial \dot{V}_z}{\partial y} = \frac{3\mu yz}{\bar{r}^5} + \frac{15}{2} J_2 \mu r_e^2 \left[\frac{2yz\bar{r}^2 - 7yz^3}{\bar{r}^9} + \frac{yz}{\bar{r}^7} \right] \qquad (21)$$

$$\frac{\partial \dot{V}_z}{\partial z} = \mu \left[\frac{3z^2 - \bar{r}^2}{\bar{r}^5} \right] + \frac{1}{2} J_2 \mu r_e^2 \left[\frac{105z^4 - 75z^2\bar{r}^2 - 6r^4}{\bar{r}^9} - \frac{3r^2 - 15z^2}{\bar{r}^7} \right] \qquad (22)$$

The routines written to propagate the dynamics consisted of a main program and two subroutines. Changes to existing routines, most significantly the dynamics, were made to fit the scenario for this study. The source code for each routine is listed in Appendix B. The main program, moveit, is a simple dynamics propagator which takes the input state vectors and propagates their orbits from the initial time to the desired end time. It also outputs the state transition matrix, Φ, at the end time as well. The main program makes use of two subroutines, haming and rhs. Haming is an ordinary differential equations integrator. It is a fourth order predictor-corrector algorithm. The subroutine rhs is the routine that actually calculates the equations of motion and the equations of variation. Once the routines were created, they were verified using published scenarios (6:253-256). Once the test satellites were propagated to the proper times, the position and velocity data were input into a program written to output the classical orbital elements given the position and velocity. The J_2 effect, if working correctly, should have produced a

regression of the ascending node as well as a rotation in the argument of perigee. The following equations used for general perturbation theory without numerical integration were used to verify the output (15:81):

$$\Omega = \Omega_0 - \frac{3 J_2 \sqrt{\mu}}{2 a_0^{7/2} \left(1 - e_0^2\right)^2} \cos i_0 \left(t - T_0\right) \tag{23}$$

$$\omega = \omega_0 - \frac{3 J_2 \sqrt{\mu}}{2 a_0^{7/2} \left(1 - e_0^2\right)^2} \left(\frac{5}{2} \sin^2 i_0 - 2\right)\left(t - T_0\right) \tag{24}$$

The amount of nodal regression and argument of perigee rotation shown in the output of the classical elements agreed to several significant digits with the values calculated from the respective equations. The results can be found in Appendix C and were deemed satisfactory for use. The amount of nodal regression was also verified with published data for given inclinations and altitudes and was within reasonable limits (12:262-263).

Truth Model

The truth model needs to output the true state x and the measurements z_t. The measurements are used by the estimation filter to output an estimated state $\hat{x}$. The true state information, for each time step, is already available from the previous routines propagating the orbits. Additionally, the range data between the two satellites is desired and is computed using

$$range(\rho) = \sqrt{\left(x_1 - x_2\right)^2 + \left(y_1 - y_2\right)^2 + \left(z_1 - z_2\right)^2} \tag{25}$$

The measurement data consists of the position vectors of the two satellites along with the range between them. The resulting observation function, $\mathbf{G(x,t)}$, takes the form

$$Z_{perfect} = \begin{Bmatrix} \rho \\ x_{sat1} \\ y_{sat1} \\ z_{sat1} \\ x_{sat2} \\ y_{sat2} \\ z_{sat2} \end{Bmatrix} = \begin{Bmatrix} \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \\ x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \end{Bmatrix} \tag{26}$$

In order to represent the errors associated with obtaining the respective measurements, Gaussian noise is added to corrupt the perfect data captured above resulting in

$$z_{corrupt} = z_{perfect} + \sigma_{noise} \tag{27}$$

The $z_{corrupt}$ data is used to form the inputs to the estimator. The σ_{noise} is the Gaussian noise with an associated instrument covariance of Q. The noise is the representation for errors in computing the range measurements from clock pulses and the positions using a GPS receiver. The noise consists of random numbers created from a uniformly distributed random number generator manipulated to meet the proper covariances. The observation data vector, $\mathbf{z}_i$, is output for each observation time-step t_i.

Bayes Filter

Now that the truth model can represent *real world* measurements, the estimation filter can be developed. Typically the choice between using a Bayes or Kalman filter is determined by inspecting the rank of the state vs. the rank of the measurement data. If the rank of the state is greater, as in this case, then the Kalman filter is the proper choice.

However, due to the very accurate data supplied by the instruments in use, the Kalman filter provides poor response (15:107). Although there are methods to correct the response, it is simpler and more efficient to attempt the use of a Bayes filter for this study.

The Bayes algorithm that follows is adapted from Modern Methods of Orbit Determination (15:96-97). The Bayes algorithm is initiated by bringing in a previous estimate of the state, $\bar{x}(-)$, and its' covariance, $P^{-1}(-)$, at the new epoch. The previous estimate becomes the reference by setting

$$x_{ref} = \bar{x}(-) \tag{28}$$

for each new observation time t_i. Next, the state vectors and Φ matrices are propagated to time t_i. The residual vector, $\mathbf{r}_i$, linearization matrix, $\mathbf{H}_i$, and the observation matrix, $\mathbf{T}_i$ are calculated and

$$r_i = z_i - G \tag{29}$$

As shown, the residuals are just the observed data supplied to the filter minus the predicted values calculated using equation (26). The H matrix is derived from taking the partials of $\mathbf{G}(\mathbf{x},t)$ with respect to the partials of the state X, where X is now

$$X^T = \left(x_1, y_1, z_1, \dot{x}_1, \dot{y}_1, \dot{z}_1, x_2, y_2, z_2, \dot{x}_2, \dot{y}_2, \dot{z}_2 \right) \tag{30}$$

14

The H matrix is

$$
H=\begin{bmatrix}
\frac{x_1-x_2}{\rho} & \frac{y_1-y_2}{\rho} & \frac{z_1-z_2}{\rho} & 0 & 0 & 0 & \frac{x_2-x_1}{\rho} & \frac{y_2-y_1}{\rho} & \frac{z_2-z_1}{\rho} & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\qquad (31)
$$

The observation matrix is found by setting

$$
T_i = H_i \Phi(t_i, t_0) \qquad (32)
$$

The observation covariance matrix is also required. The diagonal entries contain the error values of the instruments and the off-diagonal entries are zero. The zero values imply that the measurements are independent. This is probably not a valid assumption but is adequate for this study. The observation covariance matrix is

$$
Q=\begin{bmatrix}
\sigma^2_{range} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \sigma^2_{GPS} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \sigma^2_{GPS} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \sigma^2_{GPS} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \sigma^2_{GPS} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \sigma^2_{GPS} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \sigma^2_{GPS}
\end{bmatrix}
\qquad (33)
$$

Continuing with the algorithm, sum additional terms

$$
\sum_i T_i^T Q_i^{-1} T_i \qquad (34)
$$

$$\sum_i T_i^T Q_i^{-1} \bar{r}_i \tag{35}$$

Next, obtain the new covariance of the state correction:

$$P^{-1}(+) = P^{-1}(-) + \sum_{i=1}^{N} T_i^T Q_i^{-1} T_i \tag{36}$$

Also obtain the estimate of the state correction:

$$\delta \bar{x}(t_0) = P(+) \left(P^{-1}(-) \left(\bar{x}(-) - x_{ref} \right) + \sum_{i=1}^{N} T_i^T Q_i^{-1} r_{zi} \right) \tag{37}$$

Finally, correct the reference solution and check for convergence:

$$x_{ref+1}(t_0) = x_{ref}(t_0) + \delta \bar{x}(t_0) \tag{38}$$

If the algorithm fails to converge, the process repeats itself starting with the propagation

of the state and Φ matrix. If the process converged, then x_{ref+1} is the estimate with

covariance $P(+)$.

III. Performance Analysis

The truth model and the Bayes filter have been developed. The necessary information for an initial assessment of the filter's performance can be obtained through multiple runs of the models. The information of interest is the true state x_t, the estimated state $\hat{x}$, and the covariance P. The remaining task is to convert the information into a useful output.

The analysis will plot the true errors $|e_i|$ and covariances $\sqrt{P_{ii}}$ of the x, y, and z position estimates for each satellite over a period of 5 orbits. In addition, the magnitude of the true error for each satellite is plotted against the standard deviation of the covariance to get a feel for the overall position error. The magnitude of the true error is given by

$$e_{true} = \sqrt{e_x^2 + e_y^2 + e_z^2}$$

(39)

where e_i is the difference between the Δx, Δy, and Δz components of the true state and the estimated state. The estimates' covariance, or standard deviation, is found by taking the square root of the sum of the squares of the eigenvalues associated with the position components of the estimated state, specifically

$$\sigma = \sqrt{eigenvalue_x^2 + eigenvalue_y^2 + eigenvalue_z^2}$$

(40)

The purpose of the first test was to determine if the filter was stable and could provide a near perfect estimate if supplied with perfect data. The filter was initialized with the true state of the two satellites along with $P^{-1}(-) = 0$. This eliminates the need for initial run of least squares to provide an input estimate (15:106). Next, the filter was fed perfect

17

data from the truth model. Figure 3 shows the results. Only satellite 1's x position results are shown because they were all identical. The covariance remained stable and the true errors were for all purposes zero, as the raw data, in DU's, agreed to within 11 - 13 significant digits. The raw data for this case are included in Appendix C.
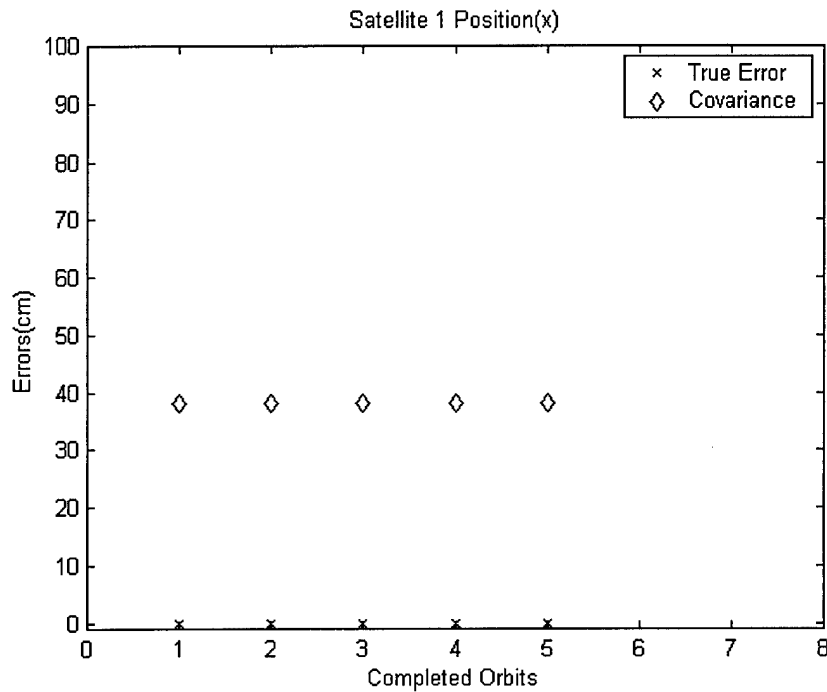
Figure 3. Comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, given perfect data.

Next, the filter was adjusted for various values to explore the range of possible measured range errors versus GPS errors. The values were adjusted by varying the diagonal elements of the observation covariance matrix Q, defined in the previous chapter. Although the range measurement errors were approximately two to three times less than the achieved relative error of 3 cm (9:48), a conservative range measurement

error was arbitrarily defined as 3 cm. The initial case was set up to reflect the expected

performance from GPS and the anticipated range measurement error, 1 m and 3 cm,

respectively. The performance of the filter was unanticipated, as the filter failed to

converge. Although the filter was fine given perfect data, the filter never met the original

convergence criteria when supplied with corrupted data. The original criteria was set at

$$\Delta_{dx} < 0.01\sqrt{P_{ii}} \tag{41}$$

where Δ_{dx} represented the difference between a given component of the estimate and the

observed component and $\sqrt{P_{ii}}$ was that component's covariance. The 0.01 factor was

removed and the process was repeated. This time the filter met the criteria until the 3rd

orbit. Upon inspection, following the first two orbits the changes to the components of

the state remained at the same order of magnitude as the values for $\sqrt{P_{ii}}$. In order to get

an output for what the filter was doing with the estimates for subsequent orbits, the

converging criteria was changed to

$$\Delta_{dx} < 12\sqrt{P_{ii}} \tag{42}$$

The value 12 was determined from experimentation to see the approximate lowest value

that could be entered which would allow the filter to converge. This indicates a change in

the twelve sigma range. Figures 4 - 11 show the various results. The validity of the

output as time progresses is questionable at best, but it lends insight into what is

happening within the filter. The covariances remain steady while the estimates continue

to worsen, and even when a very poor estimate of the state is entered after orbit 4, the

covariance still indicates the data can be trusted. Orbit 5's true error is off the charts. At

19

that point the filter would not converge before the maximum number of iterations had been exceeded. The value plotted was the last best guess estimate the filter had prior to termination.
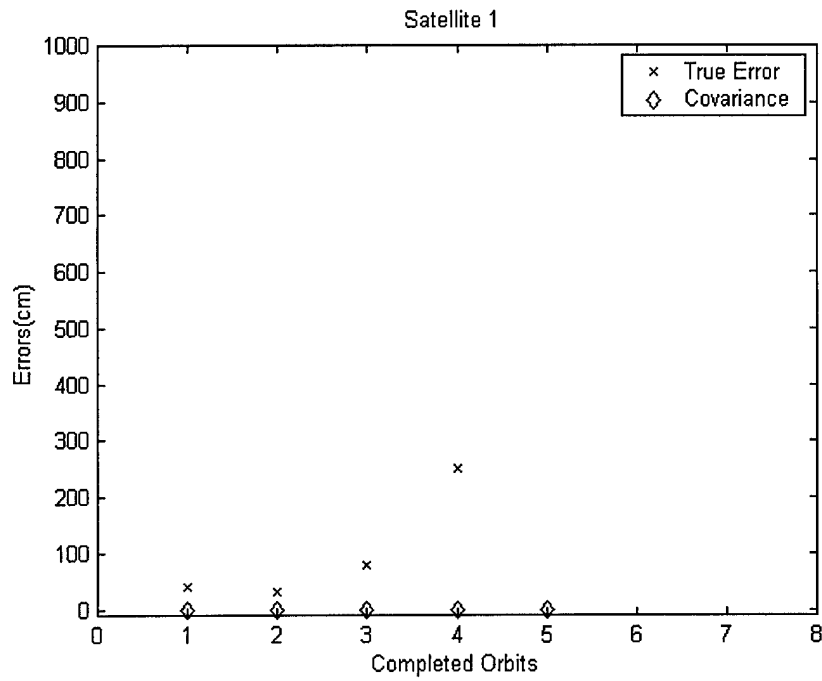
Figure 4. Comparison of $|e_i|$ and (σ_i) versus time for satellite 1.
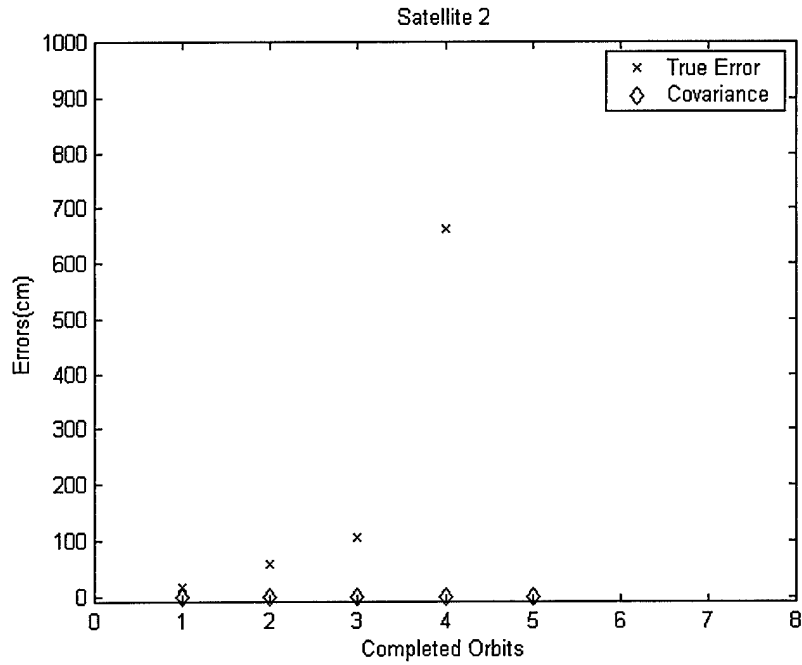
Figure 5. Comparison of $\left| e_i \right|$ and $\left(\sigma_i \right)$ versus time for satellite 2.

Figure 6. Satellite 1 position (x) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 1 meter for GPS and 3 centimeters for range data.

Figure 7.   Satellite 1 position (y) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 1 meter for GPS and 3 centimeters for range data.

Figure 8.   Satellite 1 position (z) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 1 meter for GPS and 3 centimeters for range data.

Figure 9.   Satellite 2 position (x) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 1 meter for GPS and 3 centimeters for range data.

Figure 10.   Satellite 2 position (y) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 1 meter for GPS and 3 centimeters for range data.

23

Figure 11. Satellite 2 position (z) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 1 meter for GPS and 3 centimeters for range data.

It appears that the Bayes filter may be experiencing problems similar to those that the Kalman filter has been known to experience when processing very accurate data. With the data in terms of DU's, estimates are striving to be accurate on the order of 10^{-7} to 10^{-9}, within meters to centimeters, respectively. The P values ranged from 10^{-11} to 10^{-16}, which not only represent P approaching zero but approach the limits of double precision capabilities as well. The problem arises when the initial residuals are small and P $\rightarrow$ 0, and then as the residuals increase the filter ignores them. As $P^{-1}(-)$ becomes very large, equation (36) shows that the $\sum_{i=1}^{N} T_i^T Q_i^{-1} T_i$ term becomes negligible and $P^{-1}(+) = P^{-1}(-)$.

Next, in equation (37), $P(+) \rightarrow 0$, thereby eliminating any correction to the previous state.

Prior to the failure of the filter, several other cases of interest were to be investigated as well. The cases of interest were when the accuracy provided by the range instruments weren't much better than the data provided by the GPS receiver, and the case where GPS estimates approached the accuracies obtained by post-processing, approximately 10 cm. Because each of these cases presented data within the same order of magnitude, they were run through the filter to see if the results were any different. Also, over short durations, maybe an orbit or so, the output still gives an indication of what accuracies might be obtainable if the filter was made to work. The results are shown in figures 12 - 21. As expected, the filter's performance resembles that of the previous data, although it doesn't diverge as quickly.

Figure 12. Satellite 1 position (x) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 3 meters for GPS and 1 meter for range data.
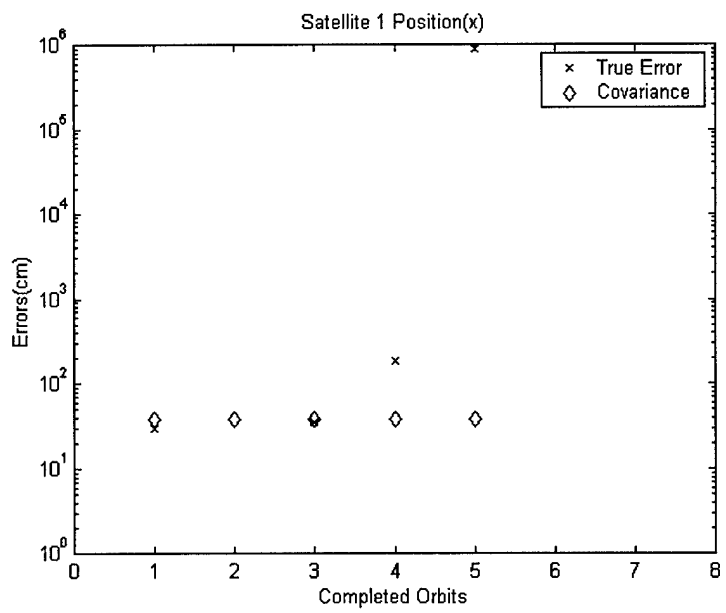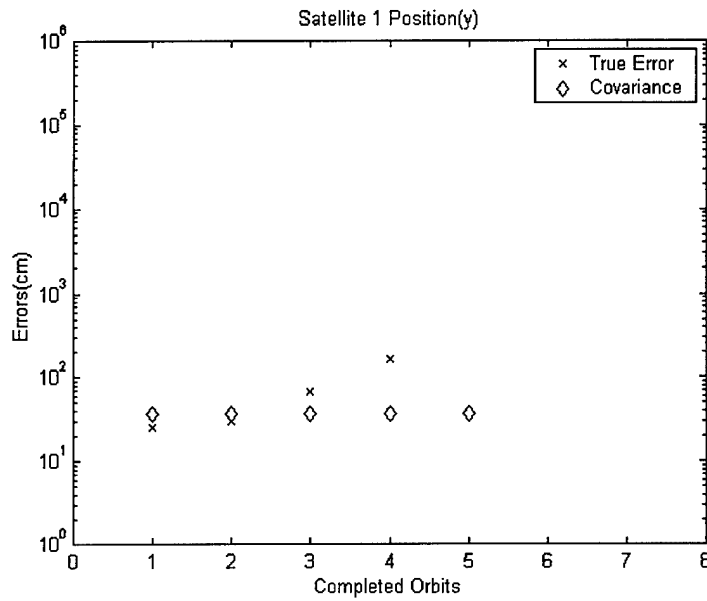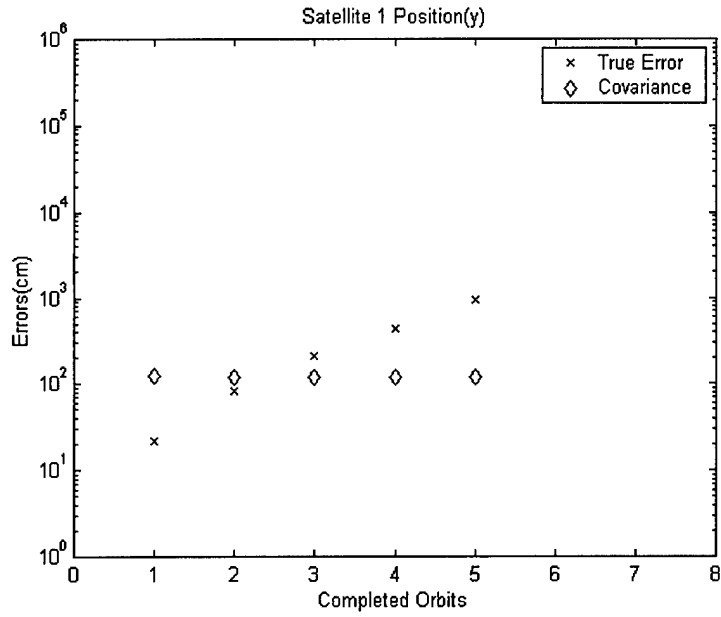
Figure 13.   Satellite 1 position (y) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 3 meters for GPS and 1 meter for range data.
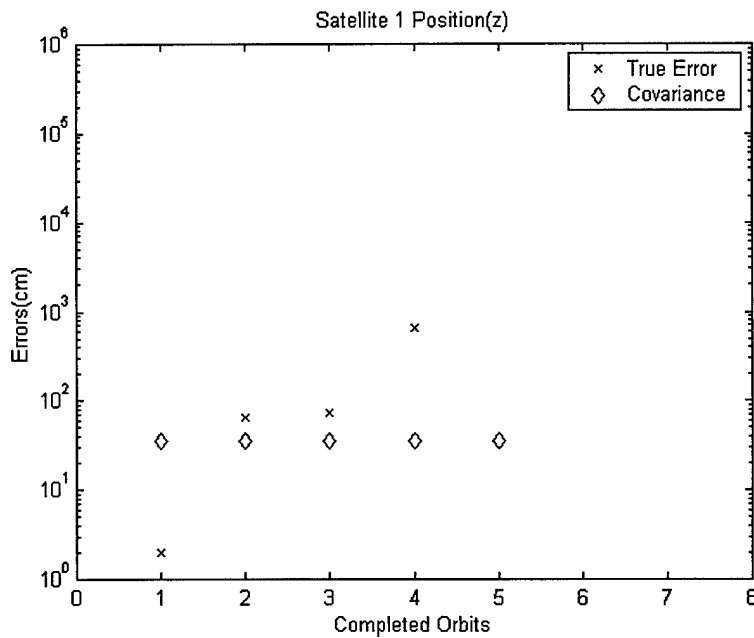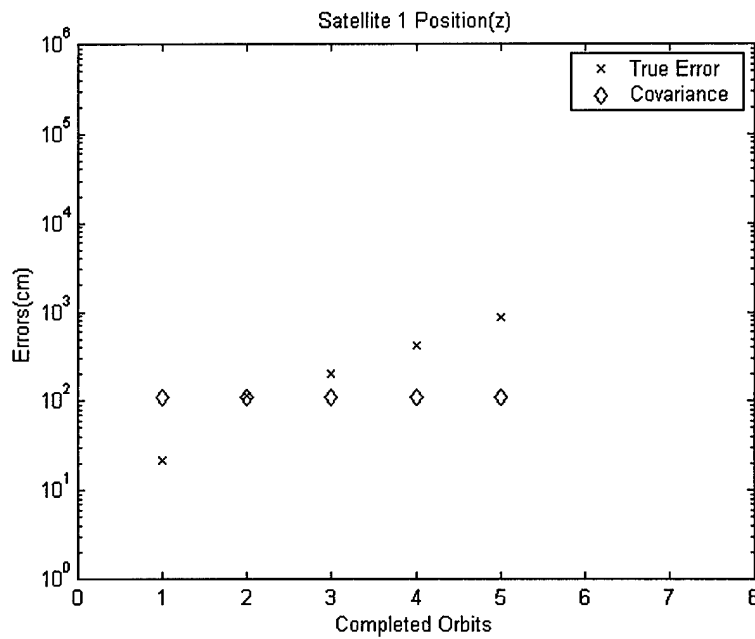

Figure 14.   Satellite 1 position (z) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 3 meters for GPS and 1 meter for range data.
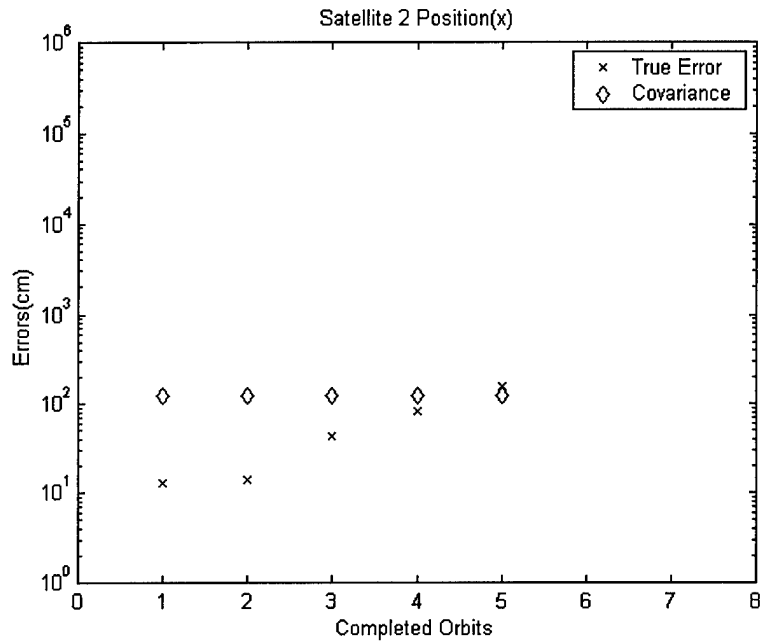
Figure 15. Satellite 2 position (x) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 3 meters for GPS and 1 meter for range data.
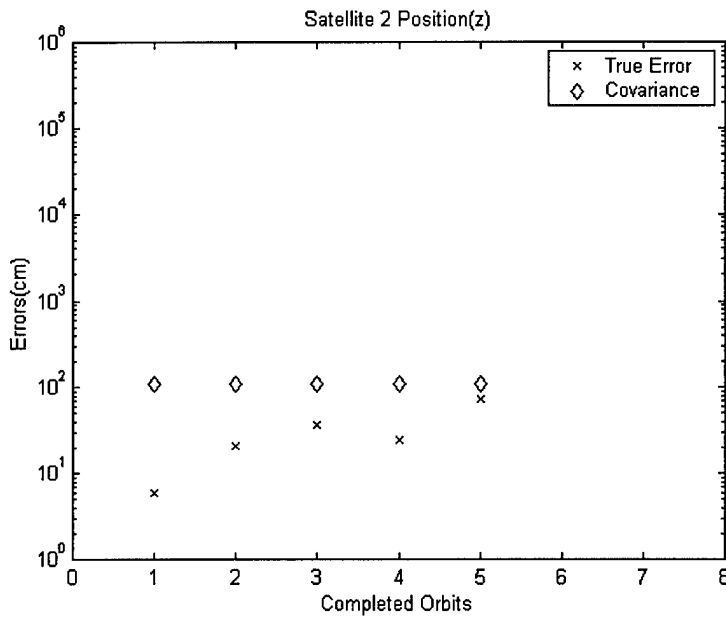
Figure 16. Satellite 2 position (z) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 3 meters for GPS and 1 meter for range data.
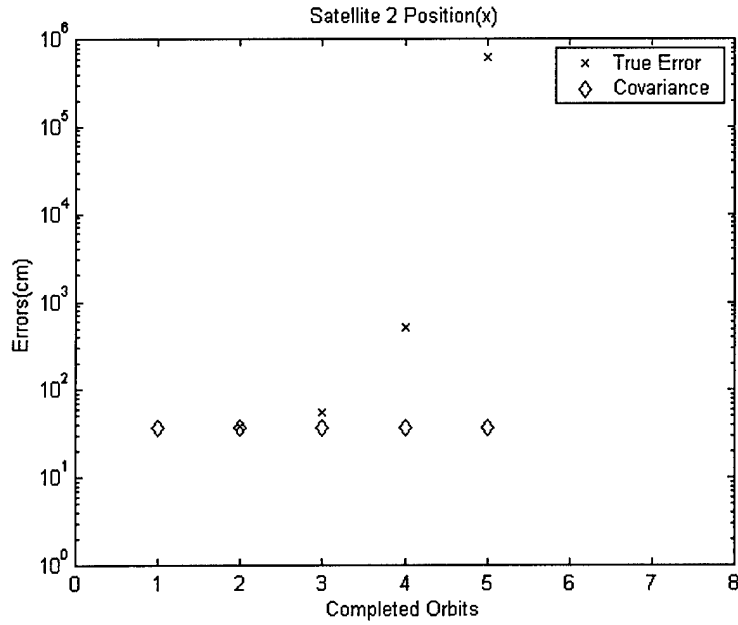
27
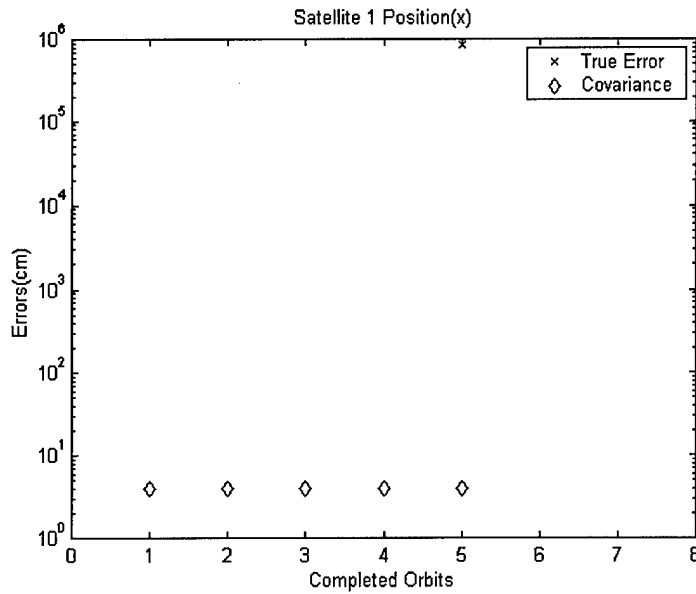
Figure 17.   Satellite 1 position (x) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 10 centimeters for GPS and 3 centimeters for range data.
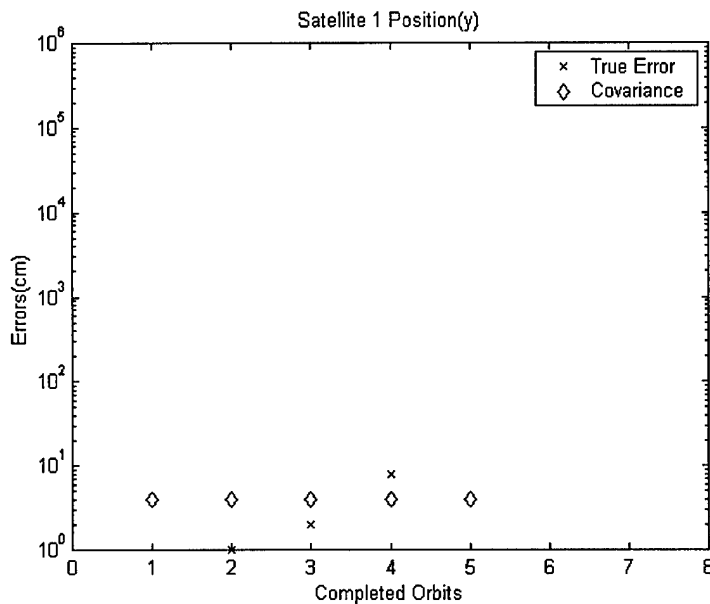
Figure 18.   Satellite 1 position (y) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 10 centimeters for GPS and 3 centimeters for range data.
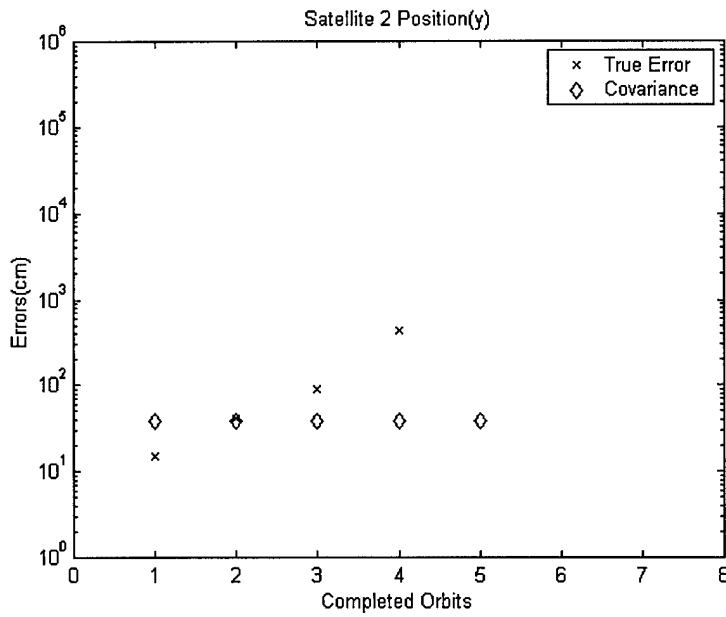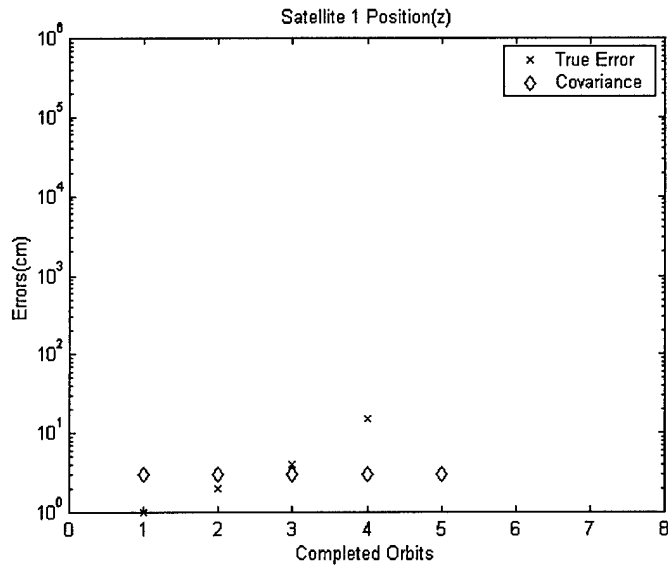
Figure 19. Satellite 1 position (z) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 10 centimeters for GPS and 3 centimeters for range data.
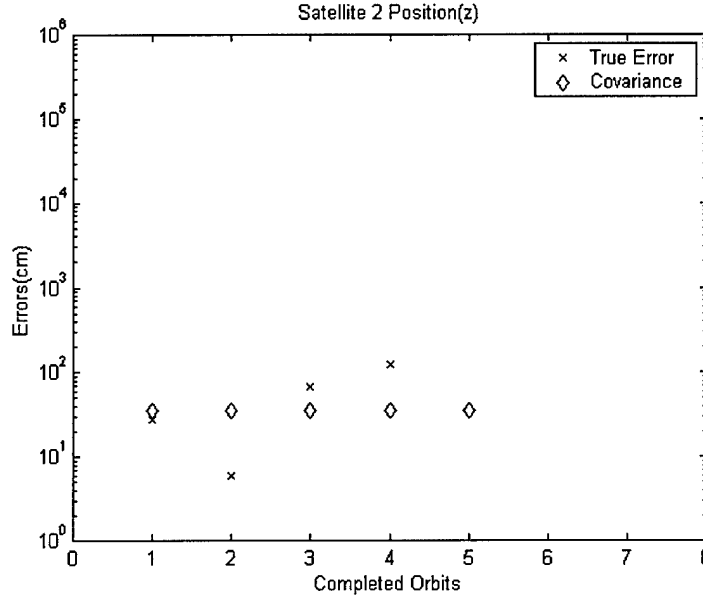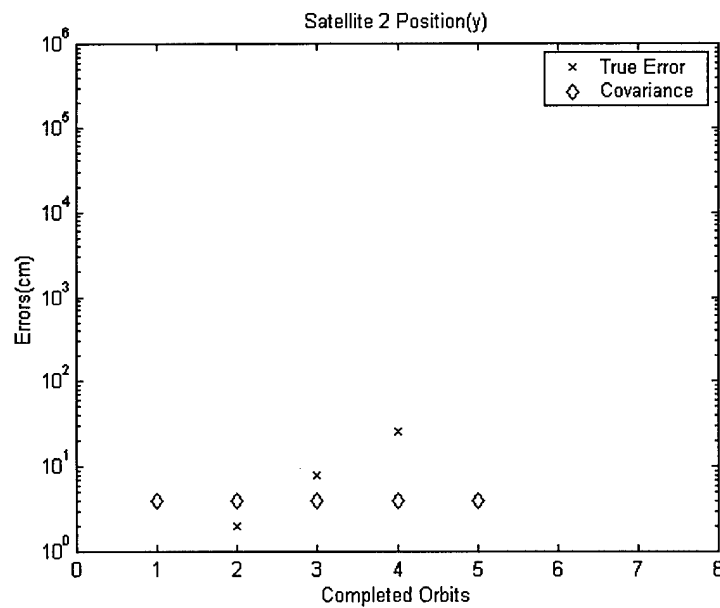

Figure 20. Satellite 2 position (y) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 10 centimeters for GPS and 3 centimeters for range data.
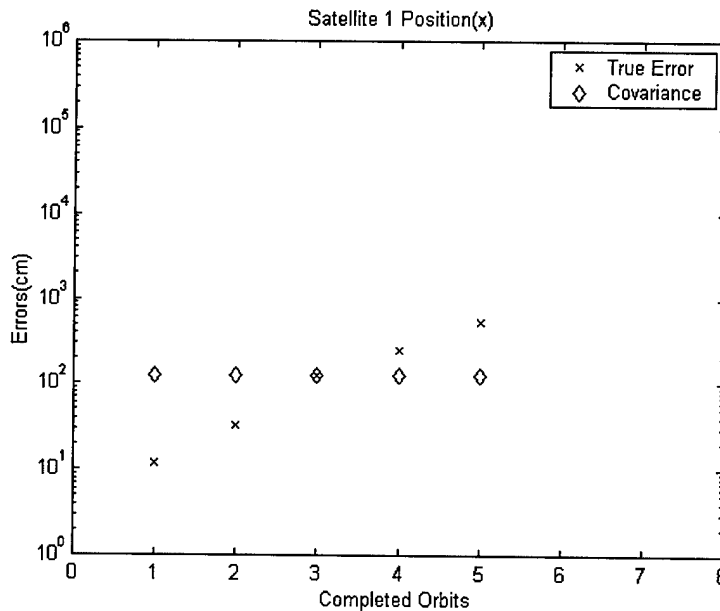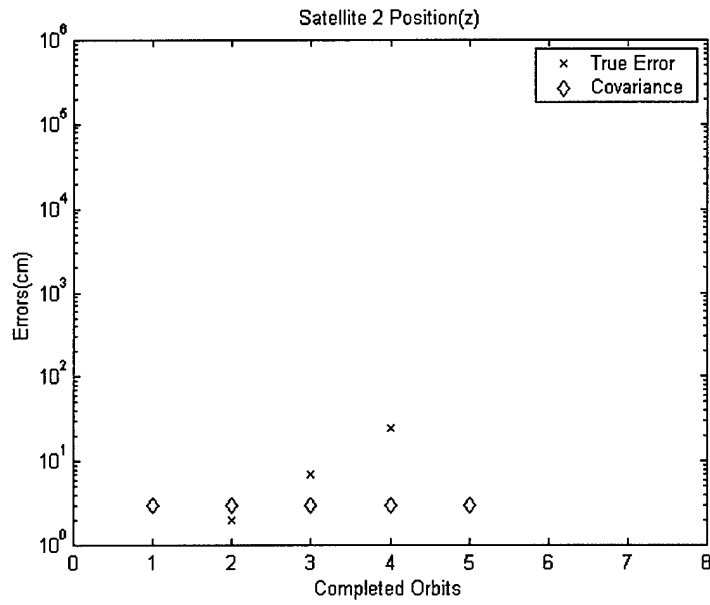
Figure 21. Satellite 2 position (z) comparison of true error (e_i) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 10 centimeters for GPS and 3 centimeters for range data.

IV. Conclusion

The Bayes algorithm as tested proved to be incapable of handling the accurate data simulated by the truth model. The covariances of the state were so small that the filter ignored any residuals, even as they increased in magnitude. Attempting to determine the absolute positions to such a small scale probably contributed to the failure. Trying to estimate positions to the accuracy of centimeters on such a large scale, ≈ 1850 km, not only tested the filter but stretched the limits of double precision computing as well.

Investigating methods used with the Kalman filter to account for similar limitations might be the next step. However, with respect to the double precision limits, it is recommended to look at the problem in terms of the satellite cluster's world. The problem might be split up between the absolute position of the cluster as a whole, and the relative position of the satellites within the cluster. For a cluster spread out over 1 - 2 km, centimeter accuracy would be to less significant digits. Also, depending on the application, the absolute position of the cluster would probably only require sub-kilometer accuracy.

Solid conclusions cannot be drawn about the ability of the filter to meet the required accuracies of a space-based radar. However, if the early orbits can be used as a measure, estimates in the tens of centimeters were obtained, which still fell short for some applications. Also, the data was for a single trial, and performing multiple trials is necessary to get statistical data for achievable accuracies.

Appendix A: Satellite Precision Requirement

This appendix describes the process used to determine the accuracy requirements for the Bayes filter. When functioning as a space-based radar, the requirements are based on many variables, most notably the type of application and the resolution desired. Several items to consider are described below, beginning with the methodology used in the previous thesis work.

In order to form a cohesive image, the relative position of each satellite needs to be known to at least one-quarter wavelength (7:443). However, at this limit fogginess and loss of contrast can prevent finer resolutions. Therefore precision requirements were reduced to one tenth of a wavelength. The maximum wavelength is a function of the size of the antenna and was determined to be approximately 216.23 meters for a cluster at an altitude of 1000km (4:57). Therefore, the accuracy required by the filter is 21.6 meters.

A pulsed Synthetic Aperture Radar (SAR) has much more stringent requirements. SAR is a technique that the cluster would implement for various applications, such as performing the role of Joint-STARS as discussed in the introduction. SAR requires that the positions and velocities between the elements are known such that the returns can be assigned with an accuracy that is better than the instrumental resolution for range and Doppler, typically 30 centimeters (8:16).

If the cluster is tasked to perform as a phased array radar, other concerns arise which can affect the required precision. Large corporate-fed phased array antennas can

experience decreased performance due to structural distortions caused by the thermal and natural radiation environments. Distortions to the plane of the array cause random phase errors, decreasing antenna gain, when the error correlation interval is large with respect to a wavelength. To prevent significant losses, the planar distortion of the elements must be held to less than one tenth of a wavelength (2:495). For a radar operating at a wavelength of 10 cm, the plane of the array must remain within 1 cm to prevent distortion. Although the elements in a satellite cluster aren't physically connected, this distortion may still apply when the *virtual plane* is in effect distorted by inaccurate position determination.

Obviously, there are many precision requirements dependent upon the application desired. In order to cover all bases, the precision requirements for this study will be set at 1 cm.

Appendix B: Computer Source Code

This appendix provides the Visual Basic source code used for the truth model and the Bayes filter. The framework for the routines was converted from FORTRAN source code provided by Dr. William Wiesel. Note, because the code was manipulated for different scenarios, such as during the tuning process, the code as printed may not produce all/any of the results presented in the analysis section.

Orbit Propagator Routine

```
'This program will propagate the orbit of a satellite

Public x As Double
Public x1 As Double
Public x2 As Double
Public y(42, 4) As Double
Public y1(42, 4) As Double
Public y2(42, 4) As Double
Public f(42, 4) As Double
Public f1(42, 4) As Double
Public f2(42, 4) As Double
Public errest(42) As Double
Public errest1(42) As Double
Public errest2(42) As Double
Public h As Double
Public n As Integer
Public mode As Integer
Public isw As Integer
Public isw1 As Integer
Public isw2 As Integer
Public jsw As Integer
Public jsw1 As Integer
Public jsw2 As Integer
Public q1(7, 7) As Double
Public zpred(7) As Double
Public zpredn(7) As Double
Public hm(7, 12) As Double

Sub Moveit()
    'Moveit is a simple dynamics propagator.
    'It takes input state vector y and propagates
    'it from t0 to tf.  The phi matrix is calculated
    'and output at tf.

    'Declare local variables
    Dim t0 As Double
    Dim tf As Double
```

```
Dim TU2min As Double
Dim phi1(6, 6) As Double
Dim phi2(6, 6) As Double
Dim n3 As Double

TU2min = 13.44686457  'conversion factor for TUs into minutes

'Read input

Open "C:\Test_move_in.txt" For Input As #1
Input #1, y1(1, 1), y1(2, 1), y1(3, 1)     'satellite 1 initial position
Input #1, y1(4, 1), y1(5, 1), y1(6, 1)      'satellite 1 initial velocity
Input #1, y2(1, 1), y2(2, 1), y2(3, 1)      'satellite 2 initial position
Input #1, y2(4, 1), y2(5, 1), y2(6, 1)      'satellite 2 initial velocity
Input #1, yr1, mon1, day1, hr1, min1, sec1      'start time
t0 = julday(yr1, mon1, day1, hr1, min1, sec1) 'convert to modified julian day
t0 = t0 * (1440# / TU2min) 'convert from julian day to TU
Input #1, yr2, mon2, day2, hr2, min2, sec2      'finish time
tf = julday(yr2, mon2, day2, hr2, min2, sec2) 'convert to modified julian day
tf = tf * (1440# / TU2min)
Input #1, mode, nstp    'mode and number of steps for integration

'Write input to output file

Open "C:\Test_move_out.txt" For Output As #2
Print #2, "initial time:"; Tab(5); t0
Print #2, "final time:"; Tab(5); tf
Print #2, "initial state vector for satellite 1:"
Print #2, y1(1, 1); Tab(5); y1(2, 1); Tab(5); y1(3, 1)
Print #2, y1(4, 1); Tab(5); y1(5, 1); Tab(5); y1(6, 1)
Print #2, "initial state vector for satellite 2:"
Print #2, y2(1, 1); Tab(5); y2(2, 1); Tab(5); y2(3, 1)
Print #2, y2(4, 1); Tab(5); y2(5, 1); Tab(5); y2(6, 1)

Open "C:\Observations.txt" For Output As #3
'Open "C:\Truedata.txt" For Output As #4

'Setup haming initialization

'number of ODEs

If mode = 1 Then
   n = 42
Else
   n = 6
End If

'initialize phi1(t0) & phi2(t0) if necessary
                          '

If mode = 1 Then
   For i = 7 To 42
      y1(i, 1) = 0#
      y2(i, 1) = 0#
```

```
      Next
      For i = 7 To 42 Step 7
        y1(i, 1) = 1#
        y2(i, 1) = 1#
      Next
End If

'Timestep setup  -  also prevents plotting more than
'100 points if plotting routine is added/called

If nstp <= 100 Then
    n1 = nstp
Else
    n1 = 100
End If

n2 = 1 + nstp / n1
n3 = n1 * n2
h = (tf - t0) / n3
x1 = t0
x2 = t0
nxt1 = 0
nxt2 = 0

'Print time interval (h) to observation file for use by filter

Print #3, h

'initialize haming

' setup haming for satellite 1
x = x1
nxt = nxt1
For i = 1 To 42
  For j = 1 To 4
     y(i, j) = y1(i, j)
  Next
Next

Call Haming(nxt)

'Retain settings for satellite 1
x1 = x
For i = 1 To 42
  For j = 1 To 4
     y1(i, j) = y(i, j)
  Next
Next
For i = 1 To 42
  For j = 1 To 4
     f1(i, j) = f(i, j)
  Next
Next
```

```
For i = 1 To 42
    errest1(i) = errest(i)
Next
nxt1 = nxt
isw1 = isw
jsw1 = jsw

'setup haming for satellite 2
x = x2
nxt = nxt2
For i = 1 To 42
  For j = 1 To 4
    y(i, j) = y2(i, j)
  Next
Next

Call Haming(nxt)

'Retain settings for satellite 2
x2 = x
For i = 1 To 42
  For j = 1 To 4
    y2(i, j) = y(i, j)
  Next
Next
For i = 1 To 42
  For j = 1 To 4
    f2(i, j) = f(i, j)
  Next
Next
For i = 1 To 42
    errest2(i) = errest(i)
Next
nxt2 = nxt
isw2 = isw
jsw2 = jsw


'Numerical Integration Loop - one timestep per call

For i = 1 To n1
  For j = 1 To n2
    ' setup haming for satellite 1
    x = x1
    nxt = nxt1
    isw = isw1
    jsw = jsw1
    For s = 1 To 42
      For t = 1 To 4
        y(s, t) = y1(s, t)
      Next
    Next
    For s = 1 To 42
```

```
   For t = 1 To 4
      f(s, t) = f1(s, t)
   Next
Next
For s = 1 To 42
   errest(s) = errest1(s)
Next

Call Haming(nxt)

'Retain settings for satellite 1
x1 = x
For s = 1 To 42
   For t = 1 To 4
      y1(s, t) = y(s, t)
   Next
Next
For s = 1 To 42
   For t = 1 To 4
      f1(s, t) = f(s, t)
   Next
Next
For s = 1 To 42
      errest1(s) = errest(s)
Next
nxt1 = nxt
isw1 = isw
jsw1 = jsw

'setup haming for satellite 2
x = x2
nxt = nxt2
isw = isw2
jsw = jsw2
For s = 1 To 42
   For t = 1 To 4
      y(s, t) = y2(s, t)
   Next
Next
For s = 1 To 42
   For t = 1 To 4
      f(s, t) = f2(s, t)
   Next
Next
For s = 1 To 42
   errest(s) = errest2(s)
Next

Call Haming(nxt)

'Retain settings for satellite 2
x2 = x
For s = 1 To 42
```

38

```
        For t = 1 To 4
           y2(s, t) = y(s, t)
        Next
      Next
      For s = 1 To 42
        For t = 1 To 4
           f2(s, t) = f(s, t)
        Next
      Next
      For s = 1 To 42
           errest2(s) = errest(s)
      Next
      nxt2 = nxt
      isw2 = isw
      jsw2 = jsw
      tob = x2
   Next
   Call Obser(nxt)

   'Introduce gaussian noise

   For iii = 1 To 7
      zpredn(iii) = zpred(iii) + randg / Sqr(q1(iii, iii))
   Next

   'Print observation data
   Print #3, tob; zpredn(1)
   Print #3, zpredn(2); zpredn(3); zpredn(4)
   Print #3, zpredn(5); zpredn(6); zpredn(7)

   'Print truth data
   'Print #4, tob; zpred(1)
   'Print #4, zpred(2); zpred(3); zpred(4)
   'Print #4, zpred(5); zpred(6); zpred(7)


Next



'Write final state vector to output file

Print #2, "Satellite 1 state vector at tf:"
Print #2, y1(1, 1); Tab(5); y1(2, 1); Tab(5); y1(3, 1)
Print #2, y1(4, 1); Tab(5); y1(5, 1); Tab(5); y1(6, 1)

'Do we print phi also??

If mode = 0 Then
   Exit Sub
Else
   For irow = 1 To 6
```

```
      For jcol = 1 To 6
         phi1(irow, jcol) = y1(6 * jcol + irow, nxt)
         Print #2, phi1(irow, jcol);
      Next
    Print #2, " "
    Next
End If
Print #2, "Satellite 2 state vector at tf:"
Print #2, y2(1, 1); Tab(5); y2(2, 1); Tab(5); y2(3, 1)
Print #2, y2(4, 1); Tab(5); y2(5, 1); Tab(5); y2(6, 1)

'Do we print phi also??

If mode = 0 Then
   Exit Sub
Else
   For irow = 1 To 6
     For jcol = 1 To 6
        phi2(irow, jcol) = y2(6 * jcol + irow, nxt)
        Print #2, phi2(irow, jcol);
     Next
   Print #2, " "
   Next
End If
Close #1
Close #2
Close #3
'Close #4
End Sub
```

Ordinary Differential Equations Integrator

```
Static Sub Haming(nxt)
    'Haming is an ordinary differential equations integrator.
    'It is a fourth order predictor-corrector algorithm,
    'which means that it carries along the last four
    'values of the state vector, and extrapolates these
    'values to obtain the next value (the prediction step).
    'It then evaluates the equations of motion at the predicted point
    'and then corrects the extrapolated value to find a
    'new value for the state vector(the correction step).
    '
    'The value nxt in the call specifies which of the 4 values
    'of the state vector is the "next" , or current, one.
    'nxt is updated by haming automatically, and must be zero on
    'the first call.
    '
    'The user must supply a main program
    'and the external routine rhs(nxt) which
    'evaluates the equations of motion.

    'Declare variables

    'Dim y(42, 4) As Double  'state vector (4 copies of it) with nxt pointing at next one
    'Dim f(42, 4) As Double  'equations of motion (4 copies)
    'Dim errest(42) As Double
    ' Dim x As Double         'independent variable (often time)
    Dim xo As Double
    ' Dim h As Double         'time step
    Dim hh As Double
    ' Dim mode As Integer     '0 for just EOM, 1 for EOM and EOV
    ' Dim n As Integer        'number of equations being integrated (6 or 42)


    tol = 0.000000000001

    'Check if this is the first call

    If nxt = 0 Then

        'This is a forwards Picard iteration (slow and expensive)
        'to step forwards in time three steps to get the 3 next points.
        'A successful startup returns nxt=1, and time has not been incremented.
        'If startup fails, nxt will be returned as zero.
        xo = x
        hh = h / 2#
        Call rhs(1)
        For l = 2 To 4
          x = x + hh
          For i = 1 To n
            y(i, l) = y(i, l - 1) + hh * f(i, l - 1)
          Next
          Call rhs(l)
```

```
    x = x + hh
    For i = 1 To n
      y(i, 1) = y(i, 1 - 1) + h * f(i, 1)
    Next
    Call rhs(1)
Next
jsw = -10
isw = 1
For i = 1 To n
  hh = y(i, 1) + h * (9# * f(i, 1) + 19# * f(i, 2) - 5# * f(i, 3) + f(i, 4)) / 24#
  If (Abs(hh - y(i, 2)) < tol) Then
    isw = isw
  Else
    isw = 0
  End If
  y(i, 2) = hh
  hh = y(i, 1) + h * (f(i, 1) + 4# * f(i, 2) + f(i, 3)) / 3#
  If (Abs(hh - y(i, 3)) < tol) Then
    isw = isw
  Else
    isw = 0
  End If
  y(i, 3) = hh
  hh = y(i, 1) + h * (3# * f(i, 1) + 9# * f(i, 2) + 9# * f(i, 3) + 3# * f(i, 4)) / 8#
  If (Abs(hh - y(i, 4)) < tol) Then
    isw = isw
  Else
    isw = 0
  End If
  y(i, 4) = hh
Next
x = xo
For l = 2 To 4
  x = x + h
  Call rhs(l)
Next

'If something was out of tolerance, perform more, up to 10, iterations

Do While isw <= 0 And jsw < 0
  jsw = jsw + 1
  isw = 1
  For i = 1 To n
    hh = y(i, 1) + h * (9# * f(i, 1) + 19# * f(i, 2) - 5# * f(i, 3) + f(i, 4)) / 24#
    If (Abs(hh - y(i, 2)) < tol) Then
      isw = isw
    Else
      isw = 0
    End If
    y(i, 2) = hh
    hh = y(i, 1) + h * (f(i, 1) + 4# * f(i, 2) + f(i, 3)) / 3#
    If (Abs(hh - y(i, 3)) < tol) Then
      isw = isw
```

```vb
      Else
        isw = 0
      End If
      y(i, 3) = hh
      hh = y(i, 1) + h * (3# * f(i, 1) + 9# * f(i, 2) + 9# * f(i, 3) + 3# * f(i, 4)) / 8#
      If (Abs(hh - y(i, 4)) < tol) Then
        isw = isw
      Else
        isw = 0
      End If
      y(i, 4) = hh
    Next
    x = xo
    For l = 2 To 4
      x = x + h
      Call rhs(l)
    Next
  Loop

  'If in tolerance, exit

  If isw <= 0 Then
    Exit Sub

  'Otherwise provide error estimate

  Else
    x = xo
    isw = 1
    jsw = 1
    For i = 1 To n
      errest(i) = 0#
    Next
    nxt = 1
    Exit Sub
  End If

'Normal propagation Loop

Else

'A call to haming with nxt=-nxt, after a successful startup, will
'will turn off the second evaluation of the equations of motion following the corrector step.
'This can save on run time

  If nxt < 1 Then
    jsw = 2
    nxt = Abs(nxt)
  Else
    jsw = 1
  End If

'This is the predictor corrector algorithm...
```

43

'First the indices are permuted

```
x = x + h
np1 = (nxt Mod 4) + 1
If isw <> 2 Then
  If nxt = 1 Then
    nxt = np1
    Exit Sub
  ElseIf nxt = 2 Then
    nxt = np1
    Exit Sub
  ElseIf nxt = 3 Then
    nxt = np1
    Exit Sub
  Else
    isw = 2
    nm2 = (np1 Mod 4) + 1
    nm1 = (nm2 Mod 4) + 1
    npo = (nm1 Mod 4) + 1

    '...then the predictor part is run to find an extrapolated value of
    'the state vector at the new time...

    For i = 1 To n
      f(i, nm2) = y(i, np1) + 4# * h * (2# * f(i, npo) - f(i, nm1) + 2# * f(i, nm2)) / 3#
      y(i, np1) = f(i, nm2) - 0.925619835 * errest(i)
    Next

    'the equations of motion are evaluated at the
    'extrapolated value of the state vector...

    Call rhs(np1)

    'and the corrector algorithm is used to add this
    'new information and obtain a better value of the
    'new state vector...

    For i = 1 To n
      y(i, np1) = (9# * y(i, npo) - y(i, nm2) + 3# * h * (f(i, np1) + 2# * f(i, npo) - f(i, nm1))) / 8#
      errest(i) = f(i, nm2) - y(i, np1)
      y(i, np1) = y(i, np1) + 0.0743801653 * errest(i)
    Next
    If jsw = 1 Then
      Call rhs(np1)
      nxt = np1
      Exit Sub
    ElseIf jsw = 2 Then
      nxt = np1
      Exit Sub
    End If
  End If
Else
  nm2 = (np1 Mod 4) + 1
```

```
nm1 = (nm2 Mod 4) + 1
npo = (nm1 Mod 4) + 1

'...then the predictor part is run to find an extrapolated value of
'the state vector at the new time...

For i = 1 To n
    f(i, nm2) = y(i, np1) + 4# * h * (2# * f(i, npo) - f(i, nm1) + 2# * f(i, nm2)) / 3#
    y(i, np1) = f(i, nm2) - 0.925619835 * errest(i)
Next

'the equations of motion are evaluated at the
'extrapolated value of the state vector...

Call rhs(np1)

'and the corrector algorithm is used to add this
'new information and obtain a better value of the
'new state vector...

For i = 1 To n
    y(i, np1) = (9# * y(i, npo) - y(i, nm2) + 3# * h * (f(i, np1) + 2# * f(i, npo) - f(i, nm1))) / 8#
    errest(i) = f(i, nm2) - y(i, np1)
    y(i, np1) = y(i, np1) + 0.0743801653 * errest(i)
Next

'finally, the equations of motion are reevaluated
'at the better value of the state vector...
'this can be suppressed

If jsw = 1 Then
    Call rhs(np1)
    nxt = np1
    Exit Sub
ElseIf jsw = 2 Then
    nxt = np1
    Exit Sub
End If
    End If
  End If
End Sub
```

Equations of Motion/Variation Calculator

```
Private Sub rhs(nxt)
    'rhs calculates equations of motion and/or not equations of variation
    'The state vector is split out as
    'y(1-3,nxt) are x,y,z components of position vector
    'y(4-6,nxt) are x,y,z components of velocity vector
    'y(7-42),nxt) is the state transition matrix, stored as
    'columns of phi end to end

    'Declaration of variables

    Dim a(6, 6) As Double
    Dim err(42) As Double
    Dim r1 As Double
    Dim r2 As Double
    Dim r3 As Double
    Dim J2 As Double


    r2 = (y(1, nxt) * y(1, nxt) + y(2, nxt) * y(2, nxt) + y(3, nxt) * y(3, nxt))
    r1 = r2 ^ 0.5
    r3 = r2 ^ 1.5

    'Constants
    mu = 1#    'DU^3/TU^2
    re = 1#    'DU
    J2 = 1082.64 * 10 ^ -6


    'Equations of motion

    'position dot = velocity vector

    f(1, nxt) = y(4, nxt)
    f(2, nxt) = y(5, nxt)
    f(3, nxt) = y(6, nxt)

    'velocity dot = gravity acceleration including J2 effects



    f(4, nxt) = (-mu * y(1, nxt) / r3) * (1# - J2 * 1.5 * ((re / r1) ^ 2#) * (5# * ((y(3, nxt) * y(3, nxt)) / r2) - 1#))
    f(5, nxt) = (-mu * y(2, nxt) / r3) * (1# - J2 * 1.5 * ((re / r1) ^ 2#) * (5# * ((y(3, nxt) * y(3, nxt)) / r2) - 1#))
    f(6, nxt) = (-mu * y(3, nxt) / r3) * (1# + J2 * 1.5 * ((re / r1) ^ 2#) * (3# - (5# * (y(3, nxt) * y(3, nxt) / r2))))


    'check to see if only interested in eom

    If mode = 0 Then
        Exit Sub
    Else
```

```
      mode = mode   'benign command to null loop
End If

'equations of variation
'calculate a matrix

For i = 1 To 6
   For j = 1 To 6
      a(i, j) = 0#      'Fill matrix with zeros
   Next
Next
a(1, 4) = 1#
a(2, 5) = 1#         'create identity matrix in upper right 3X3
a(3, 6) = 1#

'diagonal terms in lower left 3X3

a(4, 1) = (-mu * r3 + 3# * mu * y(1, nxt) * y(1, nxt) * r1) / (r2 ^ 3#) + (15# * J2 * mu * (re ^ 2#) * y(3,
nxt) * y(3, nxt) * (r2 ^ 3.5) - 105# * J2 * mu * (re ^ 2#) * y(3, nxt) * y(3, nxt) * y(1, nxt) * y(1, nxt) * (r2 ^
2.5)) / (2# * (r2 ^ 7#)) - (3# * J2 * mu * (re ^ 2#) * (r2 ^ 2.5) - 15# * J2 * mu * (re ^ 2#) * y(1, nxt) * y(1,
nxt) * r3) / (2# * (r2 ^ 5))

a(5, 2) = (-mu * r3 + 3# * mu * y(2, nxt) * y(2, nxt) * r1) / (r2 ^ 3#) + (15# * J2 * mu * (re ^ 2#) * y(3,
nxt) * y(3, nxt) * (r2 ^ 3.5) - 105# * J2 * mu * (re ^ 2#) * y(3, nxt) * y(3, nxt) * y(2, nxt) * y(2, nxt) * (r2 ^
2.5)) / (2# * (r2 ^ 7#)) - (3# * J2 * mu * (re ^ 2#) * (r2 ^ 2.5) - 15# * J2 * mu * (re ^ 2#) * y(2, nxt) * y(2,
nxt) * r3) / (2# * (r2 ^ 5))

a(6, 3) = (-mu * r3 + 3# * mu * y(3, nxt) * y(3, nxt) * r1) / (r2 ^ 3#) - (6# * mu * J2 * (re ^ 2#) * (r2 ^ 2#)
- 75# * mu * J2 * (re ^ 2#) * y(3, nxt) * y(3, nxt) * r2 + 105# * mu * J2 * (re ^ 2#) * y(3, nxt) * y(3, nxt) *
y(3, nxt) * y(3, nxt)) / (2# * (r2 ^ 4.5)) - (3# * J2 * mu * (re ^ 2#) * (r2 ^ 2.5) - 15# * J2 * mu * (re ^ 2#) *
y(3, nxt) * y(3, nxt) * r3) / (2# * (r2 ^ 5))


'off diagonal terms in lower left 3X3

a(4, 2) = (3# * mu * y(1, nxt) * y(2, nxt) * r1) / (r2 ^ 3#) - (105# * J2 * mu * (re ^ 2#) * y(3, nxt) * y(3,
nxt) * y(1, nxt) * y(2, nxt) * (r2 ^ 2.5)) / (2# * (r2 ^ 7#)) + (15# * J2 * mu * (re ^ 2#) * y(1, nxt) * y(2, nxt)
* r3) / (2# * (r2 ^ 5#))

a(5, 1) = a(4, 2)

a(4, 3) = (3# * mu * y(1, nxt) * y(3, nxt) * r1) / (r2 ^ 3#) + (30# * J2 * mu * (re ^ 2#) * y(1, nxt) * y(3,
nxt) * (r2 ^ 3.5) - 105# * J2 * mu * (re ^ 2#) * y(1, nxt) * y(3, nxt) * y(3, nxt) * y(3, nxt) * (r2 ^ 2.5)) / (2#
* (r2 ^ 7#)) + (15# * J2 * mu * (re ^ 2#) * y(1, nxt) * y(3, nxt) * r3) / (2# * (r2 ^ 5#))

a(6, 1) = a(4, 3)

a(5, 3) = (3# * mu * y(2, nxt) * y(3, nxt) * r1) / (r2 ^ 3#) + (30# * J2 * mu * (re ^ 2#) * y(2, nxt) * y(3,
nxt) * (r2 ^ 3.5) - 105# * J2 * mu * (re ^ 2#) * y(2, nxt) * y(3, nxt) * y(3, nxt) * y(3, nxt) * (r2 ^ 2.5)) / (2#
* (r2 ^ 7#)) + (15# * J2 * mu * (re ^ 2#) * y(2, nxt) * y(3, nxt) * r3) / (2# * (r2 ^ 5#))

a(6, 2) = a(5, 3)
```

47

```
'a matrix now calculated
'now calculate phi dot = a * phi and put into last
'32 slots of f matrix

For ii = 1 To 6
  For jj = 1 To 6
    ipos = 6 * ii + jj
    f(ipos, nxt) = 0#
    For kk = 1 To 6
      jpos = 6 * jj + kk
      f(ipos, nxt) = f(ipos, nxt) + a(ii, kk) * y(jpos, nxt)
    Next
  Next
Next

End Sub
```

Observations Relation Processor

```
Sub Obser(nxt)
  ' Observation relation processing
  ' Calculates predicted observation zpred,
  ' H Matrix H, and Q inverse matrix q1.

  'Public q1(4, 4) As Double
  'Public zpred(4) As Double
  'Public hm(4, 12) As Double

  'Declare local variables

  'Dim tob As Double  'Time of observation

  ' Q inverse matrix

  Sigmagps = 1 'm
  Sigmarange = 3 'cm

  q1(1, 1) = (1 / ((Sigmarange / 637813500) ^ 2)) 'DU^2
  q1(1, 2) = 0
  q1(1, 3) = 0
  q1(1, 4) = 0
  q1(1, 5) = 0
  q1(1, 6) = 0
  q1(1, 7) = 0
  q1(2, 1) = 0
  q1(2, 2) = (1 / ((Sigmagps / 6378135) ^ 2)) 'DU^2
  q1(2, 3) = 0
  q1(2, 4) = 0
  q1(2, 5) = 0
  q1(2, 6) = 0
  q1(2, 7) = 0
  q1(3, 1) = 0
  q1(3, 2) = 0
  q1(3, 3) = (1 / ((Sigmagps / 6378135) ^ 2)) 'DU^2
  q1(3, 4) = 0
  q1(3, 5) = 0
  q1(3, 6) = 0
  q1(3, 7) = 0
  q1(4, 1) = 0
  q1(4, 2) = 0
  q1(4, 3) = 0
  q1(4, 4) = (1 / ((Sigmagps / 6378135) ^ 2)) 'DU^2
  q1(4, 5) = 0
  q1(4, 6) = 0
  q1(4, 7) = 0
  q1(5, 1) = 0
  q1(5, 2) = 0
  q1(5, 3) = 0
  q1(5, 4) = 0
  q1(5, 5) = (1 / ((Sigmagps / 6378135) ^ 2)) 'DU^2
```

```
q1(5, 6) = 0
q1(5, 7) = 0
q1(6, 1) = 0
q1(6, 2) = 0
q1(6, 3) = 0
q1(6, 4) = 0
q1(6, 5) = 0
q1(6, 6) = (1 / ((Sigmagps / 6378135) ^ 2)) 'DU^2
q1(6, 7) = 0
q1(7, 1) = 0
q1(7, 2) = 0
q1(7, 3) = 0
q1(7, 4) = 0
q1(7, 5) = 0
q1(7, 6) = 0
q1(7, 7) = (1 / ((Sigmagps / 6378135) ^ 2)) 'DU^2


'predicted data vector
'Range part

zpred(1) = Sqr((y1(1, nxt) - y2(1, nxt)) ^ 2# + (y1(2, nxt) - y2(2, nxt)) ^ 2# + (y1(3, nxt) - y2(3, nxt)) ^ 2#)

'GPS position x,y,z for sat1 (y1) and sat2 (y2)

zpred(2) = y1(1, nxt)
zpred(3) = y1(2, nxt)
zpred(4) = y1(3, nxt)
zpred(5) = y2(1, nxt)
zpred(6) = y2(2, nxt)
zpred(7) = y2(3, nxt)

'H Matrix

hm(1, 1) = (y1(1, nxt) - y2(1, nxt)) / zpred(1)
hm(1, 2) = (y1(2, nxt) - y2(2, nxt)) / zpred(1)
hm(1, 3) = (y1(3, nxt) - y2(3, nxt)) / zpred(1)
hm(1, 4) = 0
hm(1, 5) = 0
hm(1, 6) = 0
hm(1, 7) = (y2(1, nxt) - y1(1, nxt)) / zpred(1)
hm(1, 8) = (y2(2, nxt) - y1(2, nxt)) / zpred(1)
hm(1, 9) = (y2(3, nxt) - y1(3, nxt)) / zpred(1)
hm(1, 10) = 0
hm(1, 11) = 0
hm(1, 12) = 0
hm(2, 1) = 1
hm(2, 2) = 0
hm(2, 3) = 0
hm(2, 4) = 0
hm(2, 5) = 0
hm(2, 6) = 0
hm(2, 7) = 0
```

hm(2, 8) = 0
hm(2, 9) = 0
hm(2, 10) = 0
hm(2, 11) = 0
hm(2, 12) = 0
hm(3, 1) = 0
hm(3, 2) = 1
hm(3, 3) = 0
hm(3, 4) = 0
hm(3, 5) = 0
hm(3, 6) = 0
hm(3, 7) = 0
hm(3, 8) = 0
hm(3, 9) = 0
hm(3, 10) = 0
hm(3, 11) = 0
hm(3, 12) = 0
hm(4, 1) = 0
hm(4, 2) = 0
hm(4, 3) = 1
hm(4, 4) = 0
hm(4, 5) = 0
hm(4, 6) = 0
hm(4, 7) = 0
hm(4, 8) = 0
hm(4, 9) = 0
hm(4, 10) = 0
hm(4, 12) = 0
hm(5, 1) = 0
hm(5, 2) = 0
hm(5, 3) = 0
hm(5, 4) = 0
hm(5, 5) = 0
hm(5, 6) = 0
hm(5, 7) = 1
hm(5, 8) = 0
hm(5, 9) = 0
hm(5, 10) = 0
hm(5, 11) = 0
hm(5, 12) = 0
hm(6, 1) = 0
hm(6, 2) = 0
hm(6, 3) = 0
hm(6, 4) = 0
hm(6, 5) = 0
hm(6, 6) = 0
hm(6, 7) = 0
hm(6, 8) = 1
hm(6, 9) = 0
hm(6, 10) = 0
hm(6, 11) = 0
hm(6, 12) = 0
hm(7, 1) = 0

```
hm(7, 2) = 0
hm(7, 3) = 0
hm(7, 4) = 0
hm(7, 5) = 0
hm(7, 6) = 0
hm(7, 7) = 0
hm(7, 8) = 0
hm(7, 9) = 1
hm(7, 10) = 0
hm(7, 12) = 0

End Sub
```

Random Number Generator

Private Function randg()

'Gaussian pseudo random number generator
'unit variance, zero mean
'Uses central limit theorem with 10 iterates
'emperical constant for sigma

```
r = 0
Randomize  'Initialize random number generator
For i = 1 To 10
  r = r + Rnd
Next

randg = 1 * (r - 5#)

End Function
```

Bayes Filter

'Non-linear Bayes Algorithm

```
Public x As Double
Public x1 As Double
Public x2 As Double
Public y(42, 4) As Double
Public y1(42, 4) As Double
Public y2(42, 4) As Double
Public f(42, 4) As Double
Public f1(42, 4) As Double
Public f2(42, 4) As Double
Public errest(42) As Double
Public errest1(42) As Double
Public errest2(42) As Double
Public h As Double
Public n As Integer
Public mode As Integer
Public isw As Integer
Public isw1 As Integer
Public isw2 As Integer
Public jsw As Integer
Public jsw1 As Integer
Public jsw2 As Integer
Public q1(7, 7) As Double
Public zpred(7) As Double
Public hm(7, 12) As Double


Sub BayesRoutine()

'Observation Storage Buffers
Dim dt As Double                 'time interval for haming
Dim timeob(3000) As Double        'times of observations
Dim allobs(7, 3000) As Double     'observations

'Internal Buffers

Dim tmat(7, 12) As Double          'observation matrix T
Dim z(7) As Double                'current observation
Dim dx(12) As Double              'State Estimate
Dim r(7) As Double                'residuals vector
Dim tob As Double                 'current observation time
Dim yminus(12) As Double            'previous estimate of state vectors
Dim pminus(12, 12) As Double         'previous inverse covariance matrix
Dim yref(12) As Double             'reference vector(equal to previous guess)
Dim htq1(12, 7) As Double           'matrix product T transpose Q inverse
Dim htq1r(12) As Double            'matrix product
Dim pinv(12, 12) As Double          'state inverse covariance at epoch
Dim p(12, 12) As Double           'state covariance at epoch
Dim tepoch As Double              'initial time of state vector
Dim phi(12, 12) As Double           'phi matrix for both satellites
Dim phip(12, 12) As Double          'phi * p
```

54

'Matrix inverter variables

Dim pnorm As Double
Dim pmax As Double
Dim toler As Double
Dim pp As Double
Dim xx(12) As Double
Dim irr1(12) As Double
Dim ir As Integer
Dim iss As Integer
Dim id As Integer
Dim ier As Integer

'READ IN PREVIOUS ESTIMATE

Open "C:\Bayes_in.txt" For Input As #1
Input #1, tepoch
Input #1, yminus(1), yminus(2), yminus(3)   'satellite 1 initial position
Input #1, yminus(4), yminus(5), yminus(6)   'satellite 1 initial velocity
Input #1, yminus(7), yminus(8), yminus(9)   'satellite 2 initial position
Input #1, yminus(10), yminus(11), yminus(12) 'satellite 2 initial velocity
Input #1, maxit   'max allowed iterations
Input #1, reject   'residual rejection criteria (# sigmas)
Input #1, nstp
For i = 1 To 12
  For j = 1 To 12
    Input #1, pminus(i, j)
  Next
Next

'FIRST GUESS IS PREVIOUS ESTIMATE

For i = 1 To 12
  yref(i) = yminus(i)
Next

'WRITE INPUT TO OUTPUT FILE

Open "C:\Bayes_Output.txt" For Output As #2
Print #2, "Epoch Time:"; Spc(5); tepoch
Print #2, "Previous Estimated State Vector(Sat 1 Position, Velocity; Sat 2 Position, Velocity)"
Print #2, yminus(1); Spc(5); yminus(2); Spc(5); yminus(3)
Print #2, yminus(4); Spc(5); yminus(5); Spc(5); yminus(6)
Print #2, yminus(7); Spc(5); yminus(8); Spc(5); yminus(9)
Print #2, yminus(10); Spc(5); yminus(11); Spc(5); yminus(12)
Print #2, "Reject if sigma greater than:"; Spc(5); reject
Print #2, "Maximum Iterations:"; Spc(5); maxit

'READ IN OBSERVATIONS

Open "C:\Observations.txt" For Input As #3
Input #3, dt

```
iob = 1
Do Until ((EOF(3)) Or (iob = 3000))
  Input #3, timeob(iob), allobs(1, iob)
  Input #3, allobs(2, iob), allobs(3, iob), allobs(4, iob)
  Input #3, allobs(5, iob), allobs(6, iob), allobs(7, iob)
  iob = iob + 1
Loop


'SET LAST PASS FLAG --- LAST ITERATION; AND # OBSERVATIONS
nob = iob - 1
idone = 0

'BEGIN ITERATION LOOP

For iter = 1 To maxit

  'Prepare true data file for later comparison with estimate
  Open "C:\Estimates.txt" For Output As #4

  'REINITIALIZE NUMERICAL INTEGRATION VARIABLES
  t = tepoch
  mode = 1
  n = 42

  'BREAKOUT SATELLITE 1 & 2 FOR CALL TO HAMING
  'ics ARE NEW REFERENCE ORBIT GUESS

  For i = 1 To 6
    y1(i, 1) = yref(i)
    y2(i, 1) = yref(i + 6)
  Next
  For i = 7 To 42
    y1(i, 1) = 0#
    y2(i, 1) = 0#
  Next
  For i = 7 To 42 Step 7
    y1(i, 1) = 1#
    y2(i, 1) = 1#
  Next

  'Timestep setup

  If nstp <= 100 Then
    n1 = nstp
  Else
    n1 = 100
  End If

  n2 = 1 + nstp / n1
  n3 = n1 * n2   'not needed ?
  h = dt  'time interval needs to be same as truth model
  x1 = t
```

```
x2 = t
nxt1 = 0
nxt2 = 0

'initialize haming

'setup haming for satellite 1
x = x1
nxt = nxt1
For i = 1 To 42
  For j = 1 To 4
    y(i, j) = y1(i, j)
  Next
Next

Call Haming(nxt)

'Retain settings for satellite 1
x1 = x
For i = 1 To 42
  For j = 1 To 4
    y1(i, j) = y(i, j)
  Next
Next
For i = 1 To 42
  For j = 1 To 4
    f1(i, j) = f(i, j)
  Next
Next
For i = 1 To 42
    errest1(i) = errest(i)
Next
nxt1 = nxt
isw1 = isw
jsw1 = jsw

'setup haming for satellite 2
x = x2
nxt = nxt2
For i = 1 To 42
  For j = 1 To 4
    y(i, j) = y2(i, j)
  Next
Next

Call Haming(nxt)

'Retain settings for satellite 2
x2 = x
For i = 1 To 42
  For j = 1 To 4
    y2(i, j) = y(i, j)
  Next
```

```
Next
For i = 1 To 42
  For j = 1 To 4
    f2(i, j) = f(i, j)
  Next
Next
For i = 1 To 42
    errest2(i) = errest(i)
Next
nxt2 = nxt
isw2 = isw
jsw2 = jsw
```

'INITIALIZE BUFFERS FOR MATRIX PRODUCT ACCUMULATION

```
'initialize htq1r to pminus*(yminus-yref)
'initialize pinv to pminus

For i = 1 To 12
  htq1r(i) = 0#
  For j = 1 To 12
    pinv(i, j) = pminus(i, j)
    htq1r(i) = htq1r(i) + pminus(i, j) * (yminus(j) - yref(j))
  Next
Next
```

'PRINT FIRST OR LAST PASS RESIDUAL HEADERS WHEN NECESSARY

```
If (iter = 1) Then
  Print #2, "First Pass Residuals:"
End If
If (idone = 1) Then
  Print #2, "Last Pass Residuals:"
End If
If ((idone = 1) Or (iter = 1)) Then
  Print #2, "Time TU"; Spc(5); "Range DU"; Spc(5); "Position Data"
End If
```

'OBSERVATION PROCESSING LOOP

```
For iob = 1 To nob          'extract this observation
  tob = timeob(iob)
  For i = 1 To 7
    z(i) = allobs(i, iob)
  Next
```

'NUMERICALLY INTEGRATE STATE AND PHI TO OB TIME

'Numerical Integration Loop - one timestep per call

```
For jj = 1 To n2
  ' setup haming for satellite 1
  x = x1
  nxt = nxt1
  isw = isw1
  jsw = jsw1
  For s = 1 To 42
    For t = 1 To 4
      y(s, t) = y1(s, t)
    Next
  Next
  For s = 1 To 42
    For t = 1 To 4
      f(s, t) = f1(s, t)
    Next
  Next
  For s = 1 To 42
    errest(s) = errest1(s)
  Next

  Call Haming(nxt)

  'Retain settings for satellite 1
  x1 = x
  For s = 1 To 42
    For t = 1 To 4
      y1(s, t) = y(s, t)
    Next
  Next
  For s = 1 To 42
    For t = 1 To 4
      f1(s, t) = f(s, t)
    Next
  Next
  For s = 1 To 42
    errest1(s) = errest(s)
  Next
  nxt1 = nxt
  isw1 = isw
  jsw1 = jsw

  'setup haming for satellite 2
  x = x2
  nxt = nxt2
  isw = isw2
  jsw = jsw2
  For s = 1 To 42
    For t = 1 To 4
      y(s, t) = y2(s, t)
    Next
  Next
  For s = 1 To 42
    For t = 1 To 4
```

```
        f(s, t) = f2(s, t)
      Next
    Next
    For s = 1 To 42
      errest(s) = errest2(s)
    Next

    Call Haming(nxt)

    'Retain settings for satellite 2
    x2 = x
    For s = 1 To 42
      For t = 1 To 4
        y2(s, t) = y(s, t)
      Next
    Next
    For s = 1 To 42
      For t = 1 To 4
        f2(s, t) = f(s, t)
      Next
    Next
    For s = 1 To 42
      errest2(s) = errest(s)
    Next
    nxt2 = nxt
    isw2 = isw
    jsw2 = jsw
  Next

'OBTAIN MATRICES FOR THIS OBSERVATION

Call Obser(nxt)

'MATRIX CALCULATIONS --- THIS OBSERVATION

'Form residual vector, test for rejection

irej = 0
For i = 1 To 7
  r(i) = z(i) - zpred(i)
  If (Abs(r(i)) > (reject / Sqr(q1(i, i)))) Then
    irej = 1
  End If
Next

'PRINT FIRST PASS & LAST PASS RESIDUALS ONLY

If ((iter = 1) Or (idone = 1)) Then
  If (irej = 0) Then
    Print #2, tob; Spc(2); r(1); Spc(2); r(2); Spc(2); r(3); Spc(2); r(4); Spc(2); r(5); Spc(2); r(6);
Spc(2); r(7)
    Print #2, " "
  Else
```

```
          Print #2, "REJECTED"; Spc(2); tob; Spc(2); r(1); Spc(2); r(2); Spc(2); r(3); Spc(2); r(4); Spc(2);
r(5); Spc(2); r(6); Spc(2); r(7)
          Print #2, " "
      End If
    End If

    'IF THIS IS THE LAST PASS, CONVERGENCE ACHIEVED, SKIP MATRIX CALCULATIONS
    'ALSO, IF THIS OBSERVATION WAS REJECTED, CAN SKIP MATRIX CALCULATIONS

    If ((idone <> 1) Or (irej = 0)) Then   'otherwise perform calculations

      'initialize phi matrix(12X12), then extract in normal form

      For i = 1 To 12
        For j = 1 To 12
          phi(i, j) = 0#
        Next
      Next

      'upper left 6X6 portion contains Sat 1 phi matrix

      For i = 1 To 6
        For j = 1 To 6
          phi(i, j) = y1(6 * j + i, nxt)
        Next
      Next

      'lower right 6X6 portion contains Sat 2 phi matrix

      For i = 1 To 6
        For j = 1 To 6
          phi(i + 6, j + 6) = y2(6 * j + i, nxt)
        Next
      Next

      'form matrix product tmat

      For irow = 1 To 7
        For jcol = 1 To 12
          tmat(irow, jcol) = 0#
          For k = 1 To 12
            tmat(irow, jcol) = tmat(irow, jcol) + hm(irow, k) * phi(k, jcol)
          Next
        Next
      Next

      'form matrix product T transpose Q inverse

      For irow = 1 To 12
        For jcol = 1 To 7
          htq1(irow, jcol) = 0#
          For k = 1 To 7
            htq1(irow, jcol) = htq1(irow, jcol) + tmat(k, irow) * q1(k, jcol)
```

```
          Next
        Next
      Next

      'form product T transpose Q inverse T, adding result to whats already there...

      For i = 1 To 12
        For j = 1 To 12
          For k = 1 To 7
            pinv(i, j) = pinv(i, j) + htq1(i, k) * tmat(k, j)
          Next
        Next
      Next

      'form product T transpose Q inverse r, adding result to whats already there...

      For i = 1 To 12
        For j = 1 To 7
          htq1r(i) = htq1r(i) + htq1(i, j) * r(j)
        Next
      Next

      'END OF MATRIX CALCULATIONS FOR THIS OBSERVATION
    End If

    Print #4, x2
    Print #4, y1(1, nxt); Spc(3); y1(2, nxt); Spc(3); y1(3, nxt)
    Print #4, y2(1, nxt); Spc(3); y2(2, nxt); Spc(3); y2(3, nxt)

    'CONTINUE TO LOOP BACK FOR MORE DATA
  Next

  'IF LAST PASS RESIDUALS HAVE PRINTED, FINISHED WITH ITERATIONS

  If (idone = 1) Then
    iter = maxit + 1    'end iteration loop
  Else
    'data is processed...improve estimate
      'invert matrix H transpose Q inverse H to find covariance p

      'BEGIN MATRIX INVERTER CODE
      'load identity matrix in p

      For i = 1 To 12
        For j = 1 To 12
          p(i, j) = 0#
        Next
      Next
      For i = 1 To 12
        p(i, i) = 1#
      Next

      'calculate inverse
```

```
'find max norm of p

pnorm = 0#
For i = 1 To 12
  For j = 1 To 12
    If (Abs(pinv(i, j)) > pnorm) Then
      pnorm = Abs(pinv(i, j))
    End If
  Next
Next

'set tolerance = 2^(- number of binary digits in mantissa)

toler = 1 ^ -12#
ier = 0
id = 1
For i = 1 To 12
  irr1(i) = 0
Next

Do While (id <= 12)
  ir = 1
  iss = 1
  pmax = 0#

  'find max pivot

  For i = 1 To 12
    If (irr1(i) = 0) Then
      For j = 1 To 12
        pp = Abs(pinv(i, j))
        If ((pp - pmax) > 0) Then
          ir = i
          iss = j
          pmax = pp
        End If
      Next
    End If
  Next

  'singularity test

  'If ((pmax / pnorm) <= toler) Then
  '   ier = 129
  'End If

  'forward elimination

  irr1(ir) = iss
  For i = 1 To 12
    If (i <> ir) Then
      pp = pinv(i, iss) / pinv(ir, iss)
```

```
          For j = 1 To 12
             pinv(i, j) = pinv(i, j) - pp * pinv(ir, j)
          Next
          pinv(i, iss) = 0#
          For j = 1 To 12
             p(i, j) = p(i, j) - pp * p(ir, j)
          Next
        End If
    Next
    id = id + 1
Loop

'back substitution

For j = 1 To 12
  For i = 1 To 12
    ir = irr1(i)
    xx(ir) = p(i, j) / pinv(i, ir)
  Next
  For i = 1 To 12
    p(i, j) = xx(i)
  Next
Next

'END MATRIX INVERTER CODE

'multiply p by T transpose Q inverse r to get correction to state

For i = 1 To 12
  dx(i) = 0#
  For j = 1 To 12
    dx(i) = dx(i) + p(i, j) * htq1r(j)
  Next
Next

'CHECK CONVERGENCE

ifail = 0
For i = 1 To 12
  If (Abs(dx(i)) < Sqr(Abs(p(i, i)))) Then
    ifail = 1
  End If
Next

'print iteration

Print #2, " "
Print #2, "Iteration:"; Spc(15); iter
Print #2, "State Corrections:"
Print #2, dx(1); Spc(2); dx(2); Spc(2); dx(3); Spc(2); dx(4)
Print #2, dx(5); Spc(2); dx(6); Spc(2); dx(7); Spc(2); dx(8)
Print #2, dx(9); Spc(2); dx(10); Spc(2); dx(11); Spc(2); dx(12)
```

```
'add in state corrections

For i = 1 To 12
   yref(i) = yref(i) + dx(i)
Next

'print current best guess

Print #2, " "
Print #2, "Current state vector:"
Print #2, yref(1); Spc(5); yref(2); Spc(5); yref(3)
Print #2, yref(4); Spc(5); yref(5); Spc(5); yref(6)
Print #2, yref(7); Spc(5); yref(8); Spc(5); yref(9)
Print #2, yref(10); Spc(5); yref(11); Spc(5); yref(12)

'convergence achieved?

If (ifail = 0) Then
   idone = 1
End If
If (iter = 1) Then   'ensure residuals get computed in the event convergence
   idone = 0        'on the first iteration
End If
If (idone = 1) Then
   Print #2, " "
   Print #2, "CONVERGENCE ACHIEVED"
End If
End If
Close #4
Next

'Failure to converge....max iterations exceeded

If (idone = 0) Then
   Print #2, " "
   Print #2, "MAXIMUM ITERATION LIMIT EXCEEDED WITHOUT CONVERGENCE"
   Exit Sub
End If

'CONVERGENCE ACHIEVED.....SUCCESS

'Print covariance matrix

Print #2, " "
Print #2, "Covariance Matrix:"
For i = 1 To 12
  For j = 1 To 12
     Print #2, p(i, j)
  Next
Next

'Print state at time of last observation
```

```
Print #2, " "
Print #2, "State at Time:"; Spc(5); x2
Print #2, y1(1, nxt); Spc(3); y1(2, nxt); Spc(3); y1(3, nxt)
Print #2, y1(4, nxt); Spc(3); y1(5, nxt); Spc(3); y1(6, nxt)
Print #2, y2(1, nxt); Spc(3); y2(2, nxt); Spc(3); y2(3, nxt)
Print #2, y2(4, nxt); Spc(3); y2(5, nxt); Spc(3); y2(6, nxt)

'Calculate and print covariance at last observation time

'Extract phi matrix at last observation time
'upper left 6X6 portion contains Sat 1 phi matrix

For i = 1 To 6
  For j = 1 To 6
     phi(i, j) = y1(6 * j + i, nxt)
  Next
Next

'lower right 6X6 portion contains Sat 2 phi matrix

For i = 1 To 6
  For j = 1 To 6
     phi(i + 6, j + 6) = y2(6 * j + i, nxt)
  Next
Next

'Propagate covariance to last observation time phi*p*phi transpose

'matrix product phi*p

For irow = 1 To 12
  For jcol = 1 To 12
    phip(irow, jcol) = 0#
    For k = 1 To 12
      phip(irow, jcol) = phip(irow, jcol) + phi(irow, k) * p(k, jcol)
    Next
  Next
Next

'matrix product p(tf)  = phi*p*phi transpose

For irow = 1 To 12
  For jcol = 1 To 12
    p(irow, jcol) = 0#
    For k = 1 To 12
      p(irow, jcol) = p(irow, jcol) + phip(irow, k) * phi(jcol, k)
    Next
  Next
Next

'Print covariance at last observation time

Print #2, " "
```

66

```
Print #2, "Covariance at last observation time:"
Print #2, " "
For i = 1 To 12
  For j = 1 To 12
    Print #2, p(i, j)
  Next
Next

Close #1
Close #2
Close #3
End Sub
```

Appendix C: Selected Raw Data Products

Truth Model Output for Orbit 2

```
initial time:
     8461.2110018206
final time:
     8461.2970036412
initial state vector for satellite 1:
-0.807449485606805
     0.917352165803769
     0.413919205101727
-0.396178531841325
    -0.585459860822974
     0.524732402555635
initial state vector for satellite 2:
-0.835243767960277
     0.873683715226178
     0.451538231365086
-0.368379283527986
    -0.615769355460676
     0.510086148766103
Satellite 1 state vector at tf: 8461.2970036414
-0.808838486343198
     0.912786114275322
     0.421234624201552
-0.393570770973196
    -0.589633865125067
     0.522009673332486
 480.577373282127   257.921262697018  -128.964356675411  -68.3403511238056
-340.401792105648  -87.8013019349703
-205.878643672641   422.620069929954   179.365909051149  -141.790630738411
205.614110369458  -28.9586823055503
  14.3019515555513 -223.654655449614   480.428285208729  -148.633081605652
143.454326037708  -204.81193572584
 303.261544988212  -115.82343649243     24.0568071736197  164.332281736282
112.860621845208   -5.40085740426988
 209.646061983969   353.300685295701  -201.564617994986  -111.067422883437
355.292284057802    48.4275189548983
 -76.2559976897607   81.3723624228136   331.080367841402   63.4949512022829
-146.349941978495   188.159442871016
Satellite 2 state vector at tf: 8461.2970036414
-0.836249729543309
     0.869145346443746
     0.458380675246369
-0.365932506957397
    -0.619570857669331
     0.507231445003974
 477.132118640815   261.328304488368  -127.150898523712  -53.4514602983594
-344.905007576405  -99.2776966243231
-202.299298106043   426.017193507838   175.187861081162  -146.401900992092
183.954113742627  -18.2612068837798
  16.0629255887217 -227.705314791095   480.002331850661  -160.091524258117
154.096334500806  -198.02322145518
 299.279066920866  -117.702054211957    28.0005693757757  165.176799314882
104.425374448143   -3.60520161721907
```

68

```
  207.656956898063   360.307857445641  -202.029635125231  -119.431949262868
357.786363536246   54.8965302024613
-72.2826318580823   80.8253005927596   327.704954406332   65.2745017162923
-139.821652578574   184.545186620933
```

Bayes Output Given Perfect Data Orbit 2

Epoch Time: 8461.2110018206
Previous Estimated State Vector(Sat 1 Position, Velocity; Sat 2
Position, Velocity)
-0.807449485606819   0.917352165803733   0.413919205101737
-0.396178531841321   -0.585459860822997   0.52473240255562
-0.835243767960266   0.873683715226187   0.451538231365091
-0.368379283527978   -0.615769355460671   0.510086148766113
Reject if sigma greater than: 3000000000000
Maximum Iterations: 20
Iteration: 1
State Corrections:
 2.87285732712223E-13   1.00430511833367E-13   3.28892913792534E-13
4.17020449632066E-13
 1.45704230493225E-13   4.82864976887204E-13   -2.58341951034324E-13
-9.04683211939617E-14
-2.9741683800975E-13   -3.74977313758039E-13   -1.29744090399336E-13   -
4.37082523673406E-13

Current state vector:
-0.807449485606532   0.917352165803834   0.413919205102066
-0.396178531840904   -0.585459860822851   0.524732402556103
-0.835243767960524   0.873683715226097   0.451538231364794
-0.368379283528353   -0.615769355460801   0.510086148765676

Iteration: 2
State Corrections:
-3.28409049240736E-13   -1.1668328394017E-13   -3.78347144825244E-13   -
4.81703529017439E-13
-1.70887778537404E-13   -5.5261827297025E-13   2.93831069796362E-13
1.04133538781725E-13
 3.36628729508717E-13   4.29768869635027E-13   1.5409660132409E-13
4.92278896813915E-13

Current state vector:
-0.80744948560686   0.917352165803717   0.413919205101688
-0.396178531841386   -0.585459860823022   0.52473240255555
-0.83524376796023   0.873683715226201   0.45153823136513
-0.368379283527923   -0.615769355460647   0.510086148766168

CONVERGENCE ACHIEVED

State at Time: 8461.2970036414
-0.80883848634329   0.912786114275213   0.421234624201559
-0.393570770973226   -0.589633865125153   0.522009673332384
-0.836249729543363   0.869145346443611   0.458380675246549
-0.365932506957233   -0.619570857669404   0.507231445003982

Covariance at last observation time:

 3.55961295646376E-15
-6.04190990689082E-16
-4.37635987738259E-16
-2.32321171255428E-14
-1.05559629217239E-14
-2.49451458682546E-14
 1.26020620515404E-15

```
 3.42177391073885E-16
-9.96039519924639E-16
 9.4770142221766E-14
 3.26729638615292E-14
 1.10941037008875E-13
-6.04190990677784E-16
 3.50167136275538E-15
 6.01412685396743E-16
 1.21980060330065E-13
 4.56747311293194E-14
 1.44744104782683E-13
 2.52038968393451E-16
 1.70688291754669E-15
-8.94265412584334E-16
-7.24067541576481E-14
-2.52654383168722E-14
-7.92728603016871E-14
-4.37635987734037E-16
 6.01412685413672E-16
 3.31477675142907E-15
 1.03307335722966E-13
 3.81290712204025E-14
 1.16109305619094E-13
-9.60974927933848E-16
-9.53374585847039E-16
 1.08971686251968E-15
-1.66621908270316E-14
-5.4136029582959E-15
-2.21708026036415E-14
-2.32321171202714E-14
 1.21980060334784E-13
 1.03307335716847E-13
 5.7911961711063E-11
 2.06436997008439E-11
 6.66971107426677E-11
 9.92062430767437E-14
-6.41362279631304E-14
-2.2596178643069E-14
 2.05517465989558E-11
 7.19758037363317E-12
 2.3590807208132E-11
-1.05559629198485E-14
 4.56747311310157E-14
 3.81290712182452E-14
 2.06436997008516E-11
 7.36663782695102E-12
 2.37689510121156E-11
 3.4389488633077E-14
-2.22372884766814E-14
-7.66166128014696E-15
 7.20269010081148E-12
 2.52836945902903E-12
 8.26144057132013E-12
-2.49451458621858E-14
 1.44744104788078E-13
 1.16109305612014E-13
 6.66971107426435E-11
 2.37689510120981E-11
 7.68391343936988E-11
```

```
 1.15829705179862E-13
-6.94616796723774E-14
-2.89669547805619E-14
 2.35612798565654E-11
 8.2437194320819E-12
 2.70667384491708E-11
 1.26020620516359E-15
 2.5203896839827E-16
-9.60974927944771E-16
 9.92062430742856E-14
 3.4389488632191E-14
 1.1582970517707E-13
 3.52821724473578E-15
-6.53039948379394E-16
-3.81110243230369E-16
-1.58482990273088E-14
-8.01156122194304E-15
-1.71521777354177E-14
 3.42177391077851E-16
 1.70688291752262E-15
-9.533745858605E-16
-6.41362279721556E-14
-2.22372884798946E-14
-6.94616796827986E-14
-6.53039948392177E-16
 3.61714550896765E-15
 5.75785526577777E-16
 1.05316283211550E-13
 3.95866553445845E-14
 1.26184145982678E-13
-9.60395199920156E-16
-8.94265412585229E-16
 1.08971686252128E-15
-2.25961786398108E-14
-7.66166127899457E-15
-2.89669547768184E-14
-3.8111024321846E-16
 5.75785526574117E-16
 3.2860285548987E-15
 1.08104205889536E-13
 3.99321048521931E-14
 1.21983408703971E-13
 9.47701422259559E-14
-7.24067541577124E-14
-1.66621908324485E-14
 2.05517465981921E-11
 7.20269010053483E-12
 2.3561279855696E-11
-1.58482990267038E-14
 1.05316283215556E-13
 1.08104205886204E-13
 6.0292684138035E-11
 2.14784131535652E-11
 6.9404503126605E-11
 3.26729638630173E-14
-2.52654383168954E-14
-5.41360296020299E-15
 7.19758037336669E-12
 2.52836945893248E-12
```

```
 8.24371943177858E-12
-8.01156122172119E-15
 3.95866553459834E-14
 3.99321048510136E-14
 2.1478413153567E-11
 7.65987086628976E-12
 2.47179441019669E-11
 1.10941037013697E-13
-7.92728603018071E-14
-2.21708026099069E-14
 2.35908072072292E-11
 8.26144057099322E-12
 2.70667384481427E-11
-1.71521777347361E-14
 1.26184145987293E-13
 1.21983408700155E-13
 6.94045031265938E-11
 2.4717944101961E-11
 7.99169220971994E-11
```

Nodal regression and argument of perigee rotation checks.

Method	Nodal Regression	Argument of Perigee Rotation
Integated Result	1.82333045601071	0.290799017174425
Non-integrated Result	1.82308895252	0.292121126222

Bibliography

1. Bate, Roger R. , Donald D. Mueller, and Jerry E. White. <u>Fundamentals of Astrodynamics</u>. New York: Dover Publications, 1971.

2. Cantafio, Leopold J. <u>Space-Based Radar Handbook</u>. Norwood MA: Artech House, Inc., 1989.

3. Covault, Craig. "Space-Based Radars Drive Advanced Sensor Technologies," <u>Aviation Week & Space Technology, 150</u>: 49-50 (April 5, 1999).

4. Filer, Sherrie N. <u>Investigation of the Observability of a Satellite Cluster in a Near Circular Orbit</u>. MS Thesis, AFIT/GA/ENY/89D-2, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1989.

5. Green, G.B. and P.Axelrad. "Space Applications of GPS," <u>Journal of The Institute of Navigation,36</u>: 239-251 (Fall 1989).

6. Hale, Francis J. <u>Introduction to Space Flight</u>. Englewood Cliffs NJ: Prentice Hall, 1994.

7. Jacobs, Donald H. <u>Fundamentals of Optical Engineering</u>. New York: McGraw-Hill Book Company, 1943

8. Johnson, William T.K. <u>Final Report of the SAR/GPS Task</u>. Jet Propulsion Laboratory and California Institute of Technology, July 1998.

9. Johnston, Stephen C. <u>Autonomous Navigation of a Satellite Cluster</u>. MS Thesis, AFIT/GA/ENY/90D-9, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1990.

10. Kaplan, Elliot D. <u>Understanding GPS: Principles and Applications</u>. Norwood MA: Artech House, Inc., 1996.

11. Scott, William B. "Testbeds Wring Out Technologies," <u>Aviation Week & Space Technology, 150</u>: 52-53 (April 5, 1999).

12. Sellers, Jerry J. <u>Understanding Space: An Introduction to Astronautics</u>. New York: McGraw-Hill, 1994.

13. "Techsat 21," Web page for Techsat 21 program, n. pag. hhtp://quark.plk.af.mil/vsd/techsat21, 6 October 99.

Bibliography

14. Ward, Captain Michael L.P. <u>Estimated Satellite Cluster Elements in Near Circular Orbit</u>. MS Thesis, AFIT/GA/AA/88D-13, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1988.

15. Wiesel, William B. Class handout distributed in MECH 731, <u>Modern Methods of Orbit Determination</u>. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, June 1998.

16. -----. <u>Spaceflight Dynamics</u>. New York: McGraw-Hill, 1997.

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE March 2000	3. REPORT TYPE AND DATES COVERED Master's Thesis	

4. TITLE AND SUBTITLE NAVIGATION OF SATELLITE CLUSTERS	5. FUNDING NUMBERS 00-402

6. AUTHOR(S)

Jeffrey S. Davis, Captain, USAF

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 P Street, Building 640 WPAFB OH 45433-7765	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GSO/ENY/00M-01

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/VAAD Attn: Andrew G. Sparks 2210 8th Street WPAFB OH 45433-7531 Phone: (937) 255-8450 DSN: 785-8450	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

Advisor: Dr. William Wiesel, ENY, Phone: (937) 255-3636 x4312 DSN: 785-3636x4312, William.Wiesel@afit.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

ABSTRACT (Maximum 200 Words)
The relative position determination of a cluster of satellites in near circular orbit was investigated in previous thesis work. The purpose of this thesis is to extend the concept to cover absolute position determination. A Bayes filter is used for the estimator with dynamics based on the two-body problem extended to account for J_2 perturbations. Measurements consist of combining Global Positioning System (GPS) data for each satellite and range data between the satellites. Simulations were conducted investigating the accuracy obtainable when combining the measurements for input into the filter. Performance results consist of comparing the magnitude of the true error to the filter covariance as a function of time. True errors are also compared to minimum accuracy requirements for a space-based radar. The filter encountered numerical difficulties due to the extreme accuracy requirements and proved unsuccessful in providing usable estimates. The results suggest separating the absolute and relative problems.

14. SUBJECT TERMS Satellite Clusters, Bayes Filter, Global Positioning System	15. NUMBER OF PAGES 87
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL