NAVIGATION OF SATELLITE CLUSTERS

THESIS

Jeffrey S. Davis, Captain, USAF

AFIT/GSO/ENY/00M-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

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

Approved for public release; distribution unlimited
NAVIGATION OF SATELLITE CLUSTERS

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

Approved:

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

Captain Gregory Agnes, Ph. D.

Dr. Steven Tragesser, Ph. D.
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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgments</td>
<td>iv</td>
</tr>
<tr>
<td>Abstract</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>I. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Background</td>
<td>1</td>
</tr>
<tr>
<td>Objectives</td>
<td>3</td>
</tr>
<tr>
<td>II. Methodology</td>
<td>5</td>
</tr>
<tr>
<td>Satellite Orbits Selection</td>
<td>5</td>
</tr>
<tr>
<td>Propagating the Orbits</td>
<td>6</td>
</tr>
<tr>
<td>Truth Model</td>
<td>12</td>
</tr>
<tr>
<td>Bayes Filter</td>
<td>13</td>
</tr>
<tr>
<td>III. Performance Analysis</td>
<td>17</td>
</tr>
<tr>
<td>IV. Conclusions</td>
<td>31</td>
</tr>
<tr>
<td>Appendix A. Satellite Precision Requirement</td>
<td>32</td>
</tr>
<tr>
<td>Appendix B. Computer Program Source Code</td>
<td>34</td>
</tr>
<tr>
<td>Appendix C. Selected Raw Data Products</td>
<td>68</td>
</tr>
<tr>
<td>Bibliography</td>
<td>75</td>
</tr>
<tr>
<td>Vita</td>
<td>77</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Techsat 21 Missions</td>
<td>2</td>
</tr>
<tr>
<td>2.</td>
<td>Spacing Criteria for Different Modes</td>
<td>5</td>
</tr>
<tr>
<td>3.</td>
<td>Comparison of true error (e_i) and covariance (\sqrt{P_i}) versus time,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>given perfect data</td>
<td>18</td>
</tr>
<tr>
<td>4.</td>
<td>Comparison of (</td>
<td>e_i</td>
</tr>
<tr>
<td>5.</td>
<td>Comparison of (</td>
<td>e_i</td>
</tr>
<tr>
<td>6.</td>
<td>Satellite 1 position (x) comparison of true error (e_i) and covariance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\sqrt{P_i}) versus time, with noise values of 1 meter for GPS and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 centimeters for range data</td>
<td>21</td>
</tr>
<tr>
<td>7.</td>
<td>Satellite 1 position (y) comparison of true error (e_i) and covariance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\sqrt{P_i}) versus time, with noise values of 1 meter for GPS and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 centimeters for range data</td>
<td>22</td>
</tr>
<tr>
<td>8.</td>
<td>Satellite 1 position (z) comparison of true error (e_i) and covariance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\sqrt{P_i}) versus time, with noise values of 1 meter for GPS and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 centimeters for range data</td>
<td>22</td>
</tr>
<tr>
<td>9.</td>
<td>Satellite 2 position (x) comparison of true error (e_i) and covariance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\sqrt{P_i}) versus time, with noise values of 1 meter for GPS and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 centimeters for range data</td>
<td>23</td>
</tr>
<tr>
<td>10.</td>
<td>Satellite 2 position (y) comparison of true error (e_i) and covariance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\sqrt{P_i}) versus time, with noise values of 1 meter for GPS and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 centimeters for range data</td>
<td>23</td>
</tr>
</tbody>
</table>
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 .............................................. 24

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 ..................................................... 25

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 ..................................................... 26

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 ..................................................... 26

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 ..................................................... 27

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

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 ............................................. 28

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 ............................................. 28
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. .............................................29

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. .............................................29

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. .............................................30
NAVIGATION OF SATELLITE CLUSTERS

1. 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.
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
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
plane, it is not a concern for this study. The velocities of the satellites were calculated using

\[ v_{cir} = \sqrt{\frac{\mu}{r}} \]  

(1)

where

\[ v_{cir} = \text{orbital velocity (circular)} \]

\[ \mu = 3.986005 \times 10^{14} \text{ km}^3/\text{s}^2 \text{ (the earth's gravitational parameter)} \]

\[ r = \text{orbit radius (km)} \]

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
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,

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

(3)
The equations of motion were derived from the gravitational potential function (1:419):

\[ V = -\frac{\mu}{r} + \frac{\mu r_e^2 J_2}{2 r^3} \left( \frac{z^2}{r^2} - 1 \right) \]  \hspace{1cm} (4)

where

\[ \mu=3.986005\times10^{14} \text{ km}^3/\text{s}^2 \text{ (the earth's gravitational parameter)} \]

\[ r_e=6378.137 \text{ km (the earth's radius)} \]

\[ J_2=0.00108263 \]

The acceleration terms were found by setting the following equalities:

\[ \ddot{x} = -\frac{\partial V}{\partial x} \] \hspace{1cm} (5)

\[ \ddot{y} = -\frac{\partial V}{\partial y} \] \hspace{1cm} (6)

\[ \ddot{z} = -\frac{\partial V}{\partial z} \] \hspace{1cm} (7)

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

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

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

\[ \ddot{z} = -\frac{\mu z}{r^3} \left[ 1 + J_2 \frac{3}{2} \left( \frac{r_e}{r} \right)^2 \left( 3 - \frac{z^2}{r^2} \right) \right] \] \hspace{1cm} (10)
where

\[ \vec{r} = \sqrt{x^2 + y^2 + z^2} \]

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

\[ \Phi(t,t_0) = A(t) \Phi(t,t_0) \]  \hspace{1cm} (11)

where \( A(t) = \frac{\partial \dot{X}}{\partial X} \), 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

\[
\frac{\partial \dot{X}}{\partial X} = \begin{bmatrix}
\frac{\partial V_x}{\partial x} & \frac{\partial V_x}{\partial y} & \frac{\partial V_x}{\partial z} & \frac{\partial V_y}{\partial x} & \frac{\partial V_y}{\partial y} & \frac{\partial V_y}{\partial z} & \frac{\partial V_z}{\partial x} & \frac{\partial V_z}{\partial y} & \frac{\partial V_z}{\partial z} \\
\frac{\partial V_y}{\partial x} & \frac{\partial V_y}{\partial y} & \frac{\partial V_y}{\partial z} & \frac{\partial V_z}{\partial x} & \frac{\partial V_z}{\partial y} & \frac{\partial V_z}{\partial z} & \frac{\partial \dot{V}_x}{\partial x} & \frac{\partial \dot{V}_x}{\partial y} & \frac{\partial \dot{V}_x}{\partial z} \\
\frac{\partial V_z}{\partial x} & \frac{\partial V_z}{\partial y} & \frac{\partial V_z}{\partial z} & \frac{\partial \dot{V}_x}{\partial x} & \frac{\partial \dot{V}_x}{\partial y} & \frac{\partial \dot{V}_x}{\partial z} & \frac{\partial \dot{V}_y}{\partial x} & \frac{\partial \dot{V}_y}{\partial y} & \frac{\partial \dot{V}_y}{\partial z} \\
\frac{\partial \dot{V}_x}{\partial x} & \frac{\partial \dot{V}_x}{\partial y} & \frac{\partial \dot{V}_x}{\partial z} & \frac{\partial \dot{V}_y}{\partial x} & \frac{\partial \dot{V}_y}{\partial y} & \frac{\partial \dot{V}_y}{\partial z} & \frac{\partial \dot{V}_z}{\partial x} & \frac{\partial \dot{V}_z}{\partial y} & \frac{\partial \dot{V}_z}{\partial z} \\
\frac{\partial \dot{V}_y}{\partial x} & \frac{\partial \dot{V}_y}{\partial y} & \frac{\partial \dot{V}_y}{\partial z} & \frac{\partial \dot{V}_z}{\partial x} & \frac{\partial \dot{V}_z}{\partial y} & \frac{\partial \dot{V}_z}{\partial z} & \end{bmatrix} \]  \hspace{1cm} (12)
Calculating the partial derivatives will show $A(t)$ to have the following form (15:78):

$$A(t) = \begin{pmatrix} \phi & I \\ A_{21} & \phi \end{pmatrix}$$  \hspace{1cm} (13)

where

$\phi = 3 \times 3$ null matrix

$I = $ identity matrix

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

$$\frac{\partial V_x}{\partial x} = \mu \left[ 3 \frac{x^2 - \vec{r}^2}{\vec{r}^5} \right] + \frac{3}{2} J_2 \mu r^2 \left[ 5 \frac{\vec{r}^2 z^2 - 3 \frac{x^2 z^2}{\vec{r}^6}}{\vec{r}^6} - \frac{\vec{r}^2 - 5 x^2}{\vec{r}^7} \right]$$  \hspace{1cm} (14)

$$\frac{\partial V_x}{\partial y} = \frac{3 \mu xy}{\vec{r}^5} + \frac{15}{2} J_2 \mu r^2 \left[ - \frac{7 xy z^2}{\vec{r}^6} + \frac{xy}{\vec{r}^7} \right]$$  \hspace{1cm} (15)

$$\frac{\partial V_x}{\partial z} = \frac{3 \mu xz}{\vec{r}^5} + \frac{15}{2} J_2 \mu r^2 \left[ \frac{2 xz \vec{r}^2 - 7 x z^3}{\vec{r}^6} + \frac{xz}{\vec{r}^7} \right]$$  \hspace{1cm} (16)

$$\frac{\partial V_y}{\partial x} = \frac{3 \mu xy}{\vec{r}^5} + \frac{15}{2} J_2 \mu r^2 \left[ - \frac{7 xy z^2}{\vec{r}^6} + \frac{xy}{\vec{r}^7} \right]$$  \hspace{1cm} (17)

$$\frac{\partial V_y}{\partial y} = \mu \left[ 3 \frac{y^2 - \vec{r}^2}{\vec{r}^5} \right] + \frac{3}{2} J_2 \mu r^2 \left[ 5 \frac{z^2 \vec{r}^2 - 3 \frac{y^2 z^2}{\vec{r}^6}}{\vec{r}^6} - \frac{\vec{r}^2 - 5 y^2}{\vec{r}^7} \right]$$  \hspace{1cm} (18)

$$\frac{\partial V_y}{\partial z} = \frac{3 \mu yz}{\vec{r}^5} + \frac{15}{2} J_2 \mu r^2 \left[ \frac{2 yz \vec{r}^2 - 7 y z^3}{\vec{r}^6} + \frac{yz}{\vec{r}^7} \right]$$  \hspace{1cm} (19)
\[
\frac{\partial V_z}{\partial x} = \frac{3\mu xz}{r^5} + \frac{15}{2} J_2 \mu r^2 \left[ \frac{2xz\dot{r}^2 - 7xz^3}{\dot{r}^9} + \frac{xz}{\dot{r}^7} \right] 
\]

(20)

\[
\frac{\partial V_z}{\partial y} = \frac{3\mu yz}{r^5} + \frac{15}{2} J_2 \mu r^2 \left[ \frac{2yz\dot{r}^2 - 7yz^3}{\dot{r}^9} + \frac{yz}{\dot{r}^7} \right] 
\]

(21)

\[
\frac{\partial V_z}{\partial z} = \mu \left[ \frac{3z^2 - \dot{r}^2}{r^5} \right] + \frac{1}{2} J_2 \mu r^2 \left[ \frac{105z^4 - 75z^2\dot{r}^2 - 6r^4}{\dot{r}^9} - \frac{3r^2 - 15z^2}{\dot{r}^7} \right] 
\]

(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, $\Phi$, 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):

\[ \Omega = \Omega_0 - \frac{3J_2}{2a_0^{7/2}} \sqrt{\frac{\mu}{1-e_0^2}} \cos i_0 (t - T_0) \] \hspace{1cm} (23)

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

\[ \text{range}(\rho) = \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2 + (z_1-z_2)^2} \] \hspace{1cm} (25)
The measurement data consists of the position vectors of the two satellites along with the range between them. The resulting observation function, \( G(x,t) \), takes the form

\[
\begin{bmatrix}
\rho \\
x_{\text{sat}1} \\
y_{\text{sat}1} \\
z_{\text{sat}1} \\
x_{\text{sat}2} \\
y_{\text{sat}2} \\
z_{\text{sat}2}
\end{bmatrix} = \begin{bmatrix}
\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \\
x_1 \\
y_1 \\
z_1 \\
x_2 \\
y_2 \\
z_2
\end{bmatrix} \quad \text{(26)}
\]

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

\[
z_{\text{corrupt}} = z_{\text{perfect}} + \sigma_{\text{noise}} \quad \text{(27)}
\]

The \( z_{\text{corrupt}} \) data is used to form the inputs to the estimator. The \( \sigma_{\text{noise}} \) 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, \( z_t \), is output for each observation time-step \( t \).

**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

\[
x_{\text{ref}} = \bar{x}(-)
\]

(28)

for each new observation time \( t_i \). Next, the state vectors and \( \Phi \) matrices are propagated to time \( t_i \). The residual vector, \( r_i \), linearization matrix, \( H_i \), and the observation matrix, \( T_i \), are calculated and

\[
r_i = z_i - G
\]

(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 \( G(x,t) \) with respect to the partials of the state \( X \), where \( X \) is now

\[
X^r = (x_1, y_1, z_1, \dot{x}_1, \dot{y}_1, \ddot{x}_1, x_2, y_2, z_2, \dot{x}_2, \dot{y}_2, \ddot{x}_2)
\]

(30)
The H matrix is

\[
H = \begin{bmatrix}
\frac{x_1 - x_2}{\rho} & \frac{y_1 - y_2}{\rho} & \frac{z_1 - z_2}{\rho} & 0 & 0 & 0 & \frac{x_2 - x_1}{\rho} & \frac{y_2 - y_1}{\rho} & \frac{z_2 - z_1}{\rho} & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]  

The observation matrix is found by setting

\[
T_i = H_i \Phi(t_i, t_o)
\]  

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

\[
Q = \begin{bmatrix}
\sigma_{range}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \sigma_{GPS}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \sigma_{GPS}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \sigma_{GPS}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \sigma_{GPS}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \sigma_{GPS}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \sigma_{GPS}^2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{GPS}^2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{GPS}^2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{GPS}^2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{GPS}^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{GPS}^2
\end{bmatrix}
\]

Continuing with the algorithm, sum additional terms

\[
\sum_i T_i^T Q_i^{-1} T_i
\]
\[ \sum_{i} T_i^T Q_i^{-1} \bar{r}_i \]  

(35)

Next, obtain the new covariance of the state correction:

\[ P^{-1}(+) = P^{-1}(-) + \sum_{i=1}^{N} T_i^T Q_i^{-1} T_i \]  

(36)

Also obtain the estimate of the state correction:

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

(37)

Finally, correct the reference solution and check for convergence:

\[ x_{ref+1}(t_0) = x_{ref}(t_0) + \delta \bar{x}(t_0) \]  

(38)

If the algorithm fails to converge, the process repeats itself starting with the propagation of the state and \( \Phi \) 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 $|e_t|$ 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

$$ e_{true} = \sqrt{e_x^2 + e_y^2 + e_z^2} $$  \hspace{1cm} (39)

where $e_t$ is the difference between the $\Delta x$, $\Delta y$, and $\Delta z$ 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

$$ \sigma = \sqrt{\text{eigenvalue}_x^2 + \text{eigenvalue}_y^2 + \text{eigenvalue}_z^2} $$  \hspace{1cm} (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 $P^{-1}(\lambda) = 0$. This eliminates the need for initial run of least squares to provide an input estimate (15:106). Next, the filter was fed perfect
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

\[ \Delta_{dc} < 0.01 \sqrt{P_i} \]  

(41)

where \( \Delta_{dc} \) represented the difference between a given component of the estimate and the observed component and \( \sqrt{P_i} \) 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 \( \sqrt{P_i} \). In order to get an output for what the filter was doing with the estimates for subsequent orbits, the converging criteria was changed to

\[ \Delta_{dc} < 12 \sqrt{P_i} \]  

(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
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 $|e|$ and $(\sigma)$ versus time for satellite 1.
Figure 5. Comparison of $|e_i|$ and $(\sigma_i)$ versus time for satellite 2.

Figure 6. Satellite 1 position ($\times$) 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 ($e_y$) and covariance $\sqrt{P_{yy}}$ 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 ($e_z$) and covariance $\sqrt{P_{zz}}$ 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.
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.

![Satellite 1 Position Chart](image)

**Figure 12.** Satellite 1 position (x) comparison of true error (e) 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 ($e_y$) and covariance $\sqrt{P_{yy}}$ 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 ($e_z$) and covariance $\sqrt{P_{zz}}$ 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) 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) and covariance $\sqrt{P_{ii}}$ versus time, with noise values of 3 meters for GPS and 1 meter for range data.
Figure 17. Satellite 1 position (x) comparison of true error (e_x) and covariance $\sqrt{P_{xx}}$ 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_y) and covariance $\sqrt{P_{yy}}$ 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,) and covariance $\sqrt{P_{zz}}$ 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,) and covariance $\sqrt{P_{yy}}$ 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 foggingness 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
Input #1, yr2, mon2, day2, hr2, min2, sec2 ' finish time
Input #1, mode, stp ' mode and number of steps for integration

'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, stp ' 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 i
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
    errrest1(i) = errrest(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
    errrest2(i) = errrest(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
  $f(s, t) = f_1(s, t)$
Next
Next
For $s = 1$ To 42
  $errest(s) = errest_1(s)$
Next

Call Haming(nxt)

' Retain settings for satellite 1
$x_1 = x$
For $s = 1$ To 42
  For $t = 1$ To 4
    $y_1(s, t) = y(s, t)$
  Next
Next
For $s = 1$ To 42
  For $t = 1$ To 4
    $f_1(s, t) = f(s, t)$
  Next
Next
For $s = 1$ To 42
  $errest_1(s) = errest(s)$
Next
nxt1 = nxt
isw1 = isw
jsw1 = jsw

' Setup Haming for satellite 2
$x = x_2$
nxt = nxt2
isw = isw2
jsw = jsw2
For $s = 1$ To 42
  For $t = 1$ To 4
    $y(s, t) = y_2(s, t)$
  Next
Next
For $s = 1$ To 42
  For $t = 1$ To 4
    $f_2(s, t) = f_2(s, t)$
  Next
Next
For $s = 1$ To 42
  $errest(s) = errest_2(s)$
Next

Call Haming(nxt)

' Retain settings for satellite 2
$x_2 = x$
For $s = 1$ To 42
For t = 1 To 4
    y2(s, t) = y(s, t)
Next
For s = 1 To 42
    f2(s, t) = f(s, t)
Next
For s = 1 To 42
    errest2(s) = errest(s)
Next
nxt2 = nx
isw2 = is
jsw2 = js
job = 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 i = 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(1)
\[
\begin{align*}
x &= x + hh \\
\text{For } i = 1 \text{ To } n \\
y(i, 1) &= y(i, 1 - 1) + h \cdot f(i, 1) \\
\text{Next} \\
\text{Call rhs}(i) \\
\text{Next} \\
\text{jsw} &= -10 \\
\text{isw} &= 1 \\
\text{For } i = 1 \text{ To } n \\
hh &= y(i, 1) + h \cdot (9\# \cdot f(i, 1) + 19\# \cdot f(i, 2) - 5\# \cdot f(i, 3) + f(i, 4)) / 24\# \\
\text{If } (\text{Abs}(hh - y(i, 2)) < \text{tol}) \text{ Then} \\
&\quad \text{isw} = \text{isw} \\
\text{Else} \\
&\quad \text{isw} = 0 \\
\text{End If} \\
y(i, 2) &= hh \\
hh &= y(i, 1) + h \cdot (f(i, 1) + 4\# \cdot f(i, 2) + f(i, 3)) / 3\# \\
\text{If } (\text{Abs}(hh - y(i, 3)) < \text{tol}) \text{ Then} \\
&\quad \text{isw} = \text{isw} \\
\text{Else} \\
&\quad \text{isw} = 0 \\
\text{End If} \\
y(i, 3) &= hh \\
hh &= y(i, 1) + h \cdot (3\# \cdot f(i, 1) + 9\# \cdot f(i, 2) + 9\# \cdot f(i, 3) + 3\# \cdot f(i, 4)) / 8\# \\
\text{If } (\text{Abs}(hh - y(i, 4)) < \text{tol}) \text{ Then} \\
&\quad \text{isw} = \text{isw} \\
\text{Else} \\
&\quad \text{isw} = 0 \\
\text{End If} \\
y(i, 4) &= hh \\
\text{Next} \\
x &= xo \\
\text{For } l = 2 \text{ To } 4 \\
x &= x + h \\
\text{Call rhs}(l) \\
\text{Next}
\end{align*}
\]

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

\[
\begin{align*}
\text{Do While } \text{isw} <= 0 \text{ And } \text{jsw} < 0 \\
\text{jsw} &= \text{jsw} + 1 \\
\text{isw} &= 1 \\
\text{For } i = 1 \text{ To } n \\
hh &= y(i, 1) + h \cdot (9\# \cdot f(i, 1) + 19\# \cdot f(i, 2) - 5\# \cdot f(i, 3) + f(i, 4)) / 24\# \\
\text{If } (\text{Abs}(hh - y(i, 2)) < \text{tol}) \text{ Then} \\
&\quad \text{isw} = \text{isw} \\
\text{Else} \\
&\quad \text{isw} = 0 \\
\text{End If} \\
y(i, 2) &= hh \\
hh &= y(i, 1) + h \cdot (f(i, 1) + 4\# \cdot f(i, 2) + f(i, 3)) / 3\# \\
\text{If } (\text{Abs}(hh - y(i, 3)) < \text{tol}) \text{ Then} \\
&\quad \text{isw} = \text{isw}
\end{align*}
\]
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 rhs(l)
Loop

'If in tolerance, exit

If isw <= 0 Then
    Exit Sub
End If

'Otherwise provide error estimate

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

'Normal propagation Loop

Else

'A call to hamiing 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
    jsr = 2
    nxt = Abs(nxt)
Else
    jsr = 1
End If

'This is the predictor corrector algorithm...

43
'First the indices are permuted

\[ x = x + h \]
\[ np1 = (\text{nxt \ Mod \ 4} + 1 \]

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# 'create identity matrix in upper right 3X3
a(2, 5) = 1# 
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 * (r2 ^ 2#)) * y(3, nxt) * y(3, nxt) * (r2 ^ 3.5) - 105# * J2 * mu * (r2 ^ 2#) * y(3, nxt) * y(3, nxt) * y(1, nxt) * y(1, nxt) * (r2 ^ 2.5) / (2# * (r2 ^ 7#)) - (3# * J2 * mu * (r2 ^ 2#) * (r2 ^ 2.5) - 15# * J2 * mu * (r2 ^ 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 * (r2 ^ 2#)) * y(3, nxt) * y(3, nxt) * (r2 ^ 3.5) - 105# * J2 * mu * (r2 ^ 2#) * y(3, nxt) * y(3, nxt) * y(2, nxt) * y(2, nxt) * (r2 ^ 2.5) / (2# * (r2 ^ 7#)) - (3# * J2 * mu * (r2 ^ 2#) * (r2 ^ 2.5) - 15# * J2 * mu * (r2 ^ 2#) * y(2, nxt) * y(2, nxt) * y(3, nxt) * y(3, nxt) * r3) / (2# * (r2 ^ 5#))

a(6, 3) = (-mu * r3 + 3# * mu * y(3, nxt) * y(3, nxt) * r1) / (r2 ^ 3#) + (6# * mu * J2 * (r2 ^ 2#)) * (r2 ^ 2#) - 75# * mu * J2 * (r2 ^ 2#) * y(3, nxt) * y(3, nxt) * r2 + 105# * mu * J2 * (r2 ^ 2#) * y(3, nxt) * y(3, nxt) * y(3, nxt) * y(3, nxt) / (2# * (r2 ^ 4.5)) - (3# * J2 * mu * (r2 ^ 2#) * (r2 ^ 2.5) - 15# * J2 * mu * (r2 ^ 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 * (r2 ^ 2#)) * y(3, nxt) * y(3, nxt) * y(1, nxt) * y(2, nxt) * (r2 ^ 2.5) / (2# * (r2 ^ 7#)) + (15# * J2 * mu * (r2 ^ 2#)) * y(1, nxt) * y(2, nxt) * y(3, 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 * (r2 ^ 2#)) * y(1, nxt) * y(3, nxt) * (r2 ^ 3.5) - 105# * J2 * mu * (r2 ^ 2#) * y(1, nxt) * y(3, nxt) * y(3, nxt) * y(3, nxt) * (r2 ^ 2.5) / (2# * (r2 ^ 7#)) + (15# * J2 * mu * (r2 ^ 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 * (r2 ^ 2#)) * y(2, nxt) * y(3, nxt) * (r2 ^ 3.5) - 105# * J2 * mu * (r2 ^ 2#) * y(2, nxt) * y(3, nxt) * y(3, nxt) * y(3, nxt) * (r2 ^ 2.5) / (2# * (r2 ^ 7#)) + (15# * J2 * mu * (r2 ^ 2#)) * y(2, nxt) * y(3, nxt) * r3) / (2# * (r2 ^ 5#))

a(6, 2) = a(5, 3)
' 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
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
\[ q1(5, 6) = 0 \]
\[ q1(5, 7) = 0 \]
\[ q1(6, 1) = 0 \]
\[ q1(6, 2) = 0 \]
\[ q1(6, 3) = 0 \]
\[ q1(6, 4) = 0 \]
\[ q1(6, 5) = 0 \]
\[ q1(6, 6) = (1 / ((Sigmagps / 6378135) ^ 2)) \cdot DU^2 \]
\[ q1(6, 7) = 0 \]
\[ q1(7, 1) = 0 \]
\[ q1(7, 2) = 0 \]
\[ q1(7, 3) = 0 \]
\[ q1(7, 4) = 0 \]
\[ q1(7, 5) = 0 \]
\[ q1(7, 6) = 0 \]
\[ q1(7, 7) = (1 / ((Sigmagps / 6378135) ^ 2)) \cdot DU^2 \]

'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
' empirical 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 h1q(12, 7) As Double 'matrix product T transpose Q inverse
Dim h1qr(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 phi(12, 12) As Double 'phi * p
'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
iof = 1
Do Until ((EOF(3)) Or (iof = 3000))
    Input #3, timeo(iof), allobs(1, iоф)
    Input #3, allobs(2, iоф), allobs(3, iоф), allobs(4, iоф)
    Input #3, allobs(5, iоф), allobs(6, iоф), allobs(7, iоф)
    iоф = iоф + 1
Loop

'SET LAST PASS FLAG --- LAST ITERATION; AND # OBSERVATIONS
nоб = iоф - 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

    'Time step setup
    If ntp <= 100 Then
        n1 = ntp
    Else
        n1 = 100
    End If

    n2 = 1 + ntp / 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
        y1(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
    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
    errorest2(i) = errorest(i)
Next
nxt2 = nxt
isw2 = isw
jsw2 = jsw

'THREE DIMENSIONAL CODE

'INITIALIZE BUFFERS FOR MATRIX PRODUCT ACCUMULATION

'initialize htlr to pminus*(yminus-yref)
'initialize pinv to pminus

For i = 1 To 12
    htlr(i) = 0#
For j = 1 To 12
    pinv(i, j) = pminus(i, j)
    htlr(i) = htlr(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
\[
f(s, t) = f_2(s, t) \\
Next \\
Next \\
For s = 1 To 42 \\
  errst(s) = errst2(s) \\
Next \\
Call Haming(nxt) \\
\]

' Retain settings for satellite 2 
\[
x_2 = \text{x} \\
For s = 1 To 42 \\
  For t = 1 To 4 \\
    y_2(s, t) = y(s, t) \\
  Next \\
Next \\
For s = 1 To 42 \\
  For t = 1 To 4 \\
    f_2(s, t) = f(s, t) \\
  Next \\
Next \\
For s = 1 To 42 \\
  errst2(s) = errst(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; Spe(2); r(1); Spe(2); r(2); Spe(2); r(3); Spe(2); r(4); Spe(2); r(5); Spe(2); r(6); Spe(2); r(7) 
  Print #2, " " 
Else 

60
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#
    Next
For k = 1 To 12
    tmat(irow, jcol) = tmat(irow, jcol) + hm(irow, k) * phi(k, jcol)
Next
Next

'form matrix product T transpose Q inverse

For irow = 1 To 12
    For jcol = 1 To 7
        htl1(irow, jcol) = 0#
    Next
For k = 1 To 7
    htl1(irow, jcol) = htl1(irow, jcol) + tmat(k, irow) * q1(k, jcol)
Next  
Next  
Next  

'form product T transpose Q inverse T, adding result to what's 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 what's 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  

62
'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
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
Next
End If

'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)
    End If
Next
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#
Next
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
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
  0.45138231365086
-0.368379283527986
  -0.615769355460676
  0.510086148766103
Satellite 1 state vector at tf: 8461.2970036414
-0.808838463643198
  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  -28.958682305503
14.3019515555513  -223.654655449614  480.428285208729 -148.633081605652
143.454326037708  -204.81193572584
303.26154988212  -115.82343649243  24.0568071736197  164.332281736282
112.860621845208  -5.40085740426988
209.646061983969  353.300685295701 -201.564617994986 -111.06742283437
355.292284057802  48.4275189548883
-76.255976897607  81.372364228136  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
-344.90507576405  -99.2776966243231
-202.299298106043  426.017193507838  175.187861081162 -146.40190099209
183.954113742627  -18.2612068837798
16.0629255887217  -227.705341791095  480.002331850661 -160.09152458117
154.096334500806  -198.02322145518
299.279066920866  -117.702054211957  28.0005693757757  165.176799314882
104.42537448143  -3.6502161721907
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>207.656956898063</td>
<td>360.307857445641</td>
<td>-202.029635125231</td>
<td>-119.431949262868</td>
<td></td>
</tr>
<tr>
<td>357.786363536246</td>
<td>54.8965302024613</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-72.2826318580823</td>
<td>80.8253005927596</td>
<td>327.704954406332</td>
<td>65.2745017162923</td>
<td></td>
</tr>
<tr>
<td>-139.821652578574</td>
<td>184.545186620933</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
Current state vector:
-0.807449485606532 0.917352165803834 0.413919205102066
-0.396178531840904 -0.585459860822851 0.524732402556103
-0.835243767960524 0.873683715226097 0.451538231364794
-0.368379283528353 -0.615769355460681 0.510086148765676
Iteration: 2
State Corrections:
4.81703529017439E-13
-1.70887785374004E-13 -5.5261827297025E-13 2.93831069796362E-13
1.04133538781725E-13
3.36628729508717E-13 4.29768869635027E-13 1.5409660132409E-13
4.9227898619151E-13
Current state vector:
-0.80744948560686 -0.917352165803834 0.413919205101688
-0.396178531841386 -0.585459860823022 0.52473240255555
-0.835243767960256 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 0.522009673323284
-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
-1.0559629217239E-14
-2.49451458682546E-14
1.2602620515404E-15
<table>
<thead>
<tr>
<th>8.24371943177858E-12</th>
<th>-8.01156122172119E-15</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.95866553459834E-14</td>
<td>3.99321048510136E-14</td>
</tr>
<tr>
<td>2.1478413153567E-11</td>
<td>7.65987086628976E-12</td>
</tr>
<tr>
<td>2.47179441019669E-11</td>
<td>1.10941037013697E-13</td>
</tr>
<tr>
<td>-7.92728603018071E-14</td>
<td>-2.21708026099069E-14</td>
</tr>
<tr>
<td>2.35908072072292E-11</td>
<td>8.26144057099322E-12</td>
</tr>
<tr>
<td>2.7067384481427E-11</td>
<td>-1.7152177347361E-14</td>
</tr>
<tr>
<td>-1.26184145987293E-13</td>
<td>1.21983408700155E-13</td>
</tr>
<tr>
<td>6.94045031265938E-11</td>
<td>2.47179441019661E-11</td>
</tr>
<tr>
<td>7.99169220971994E-11</td>
<td>-7.92728603018071E-14</td>
</tr>
</tbody>
</table>
Nodal regression and argument of perigee rotation checks.

<table>
<thead>
<tr>
<th>Method</th>
<th>Nodal Regression</th>
<th>Argument of Perigee Rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrated Result</td>
<td>1.82333045601071</td>
<td>0.290799017174425</td>
</tr>
<tr>
<td>Non-integrated Result</td>
<td>1.82308895252</td>
<td>0.292121126222</td>
</tr>
</tbody>
</table>
Bibliography


Bibliography


Vita

Captain Jeffrey S. Davis was born on 14 August 1962 in Sumner, Iowa. He graduated from Eldora High School in Eldora, Iowa in 1980. He enlisted in the Air Force in 1983. He graduated from McMurry University in Abilene, Texas with a Bachelor of Science degree in Mathematics-Computer Science and was accepted to Officer Training School. He was commissioned on 22 September 1993. After a tour as a Communications Officer he attended Undergraduate Space and Missile Training at Vandenberg AFB, California. From there he was assigned to the 5th Space Operations Squadron located at Onizuka AS in Sunnyvale, California. He supported launch and early orbit operations for the Defense Satellite Communications System (DSCS) Flight 26 mission and the SKYNET 4D mission. He also provided Air Force Satellite Control Network (AFSCN) support for Space Shuttle missions STS-82 and STS-91 as well as numerous Delta II missions. He departed as a NATO/SKYNET Systems Instructor/Evaluator in August 1998 to attend the School of Engineering at the Air Force Institute of Technology. Upon graduation, he will be assigned to Head Quarters Air Force Space Command at Peterson AFB, Colorado.

Permanent Address: 506 N. Washington
Eldora, IA 50627
**Title and Subtitle**

NAVIGATION OF SATELLITE CLUSTERS

**Author(s)**

Jeffrey S. Davis, Captain, USAF

**Performing Organization Name(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

**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

**Supplementary Notes**

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

**DISTRIBUTION / AVAILABILITY STATEMENT**

Approved for public release; distribution unlimited.

**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.

**Subject Terms**

Satellite Clusters, Bayes Filter, Global Positioning System

**CLASSIFICATION OF REPORT**

UNCLASSIFIED

**CLASSIFICATION OF THIS PAGE**

UNCLASSIFIED

**CLASSIFICATION OF ABSTRACT**

UNCLASSIFIED

**LIMITATION OF ABSTRACT**

UL

**NUMBER OF PAGES**

97

**PRICE CODE**

298-102