Star pattern recognition and spacecraft attitude determination

Thomas E. Strikwerda
John L. Junkins

Virginia Polytechnic Institute
and State University
Blacksburg, Virginia 24061

Prepared for
U.S. ARMY CORPS OF ENGINEERS
ENGINEER TOPOGRAPHIC LABORATORIES
FORT BELVOIR, VIRGINIA 22060

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED
Destroy this report when no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The citation in this report of trade names of commercially available products does not constitute official endorsement or approval of the use of such products.
A method for real-time on-board spacecraft attitude determination is presented. The method is suitable for processing CCD or CID stellar camera and rate gyro output to determine orientation to better than five arc seconds. The system is self-calibrating; uncertain parameters such as interlock angles and gyro biases can be included in the estimation algorithms. The system is implemented in a microcomputer system which conclusively establishes that the system is compatible with on-board computation constraints.
PREFACE

This document is the final report of Contract DAAK70-78-C-0038 for the U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia.

The authors appreciate the capable guidance of Mr. L. A. Gambino, Director of the Computer Science Laboratory (USAETL), who served as Technical Monitor for this effort.
SUMMARY

The primary results of the research efforts are the following:

1. Development of an approach for real time on-board estimation of spacecraft orientation with sub five arc-second precision.

2. Detailed formulation of an efficient and reliable star pattern recognition strategy appropriate for use with charged-coupled-device (CCD) array-type star sensors.

3. Formulation of a motion integration/Kalman filter algorithm to integrate gyro measured angular rates and (by sequential processing of the discrete orientation information available from the star sensing, identification, and attitude determination process) provide optimal real time estimates of spacecraft orientation and angular velocity.

4. Development of truth models to generate realistic input data for the star pattern recognition and Kalman filter strategies.

5. Formulation of algorithms using Euler parameters to define orientation.

6. Implementation and validation of the approach in a laboratory microcomputer - the objective being to assess the problems associated with a real-time, on-board version of this system.

These results are discussed in detail herein. This report is organized in such a fashion that the key features and results of the work are discussed in the main body of the text; the more involved and technical details are documented in the ten appendices.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>i</td>
</tr>
<tr>
<td>Summary</td>
<td>ii</td>
</tr>
<tr>
<td>1.0 Introduction and System Overview</td>
<td>1</td>
</tr>
<tr>
<td>2.0 Orientation Parameters and Coordinate Frames</td>
<td>10</td>
</tr>
<tr>
<td>3.0 Process B</td>
<td>16</td>
</tr>
<tr>
<td>4.0 Process C</td>
<td>22</td>
</tr>
<tr>
<td>5.0 Truth Model and Simulation Tests</td>
<td>26</td>
</tr>
<tr>
<td>6.0 Conclusions</td>
<td>49</td>
</tr>
<tr>
<td>7.0 References</td>
<td>51</td>
</tr>
<tr>
<td>Appendix</td>
<td></td>
</tr>
<tr>
<td>1  CCD Star Tracker</td>
<td>52</td>
</tr>
<tr>
<td>2  CCD Instrument Response and Stellar Magnitude Conversion</td>
<td>54</td>
</tr>
<tr>
<td>3  Star Data Base and Mission Catalog Creation</td>
<td>61</td>
</tr>
<tr>
<td>4  Inter-Star Cosine Calculations</td>
<td>65</td>
</tr>
<tr>
<td>5  Stellar Aberration</td>
<td>67</td>
</tr>
<tr>
<td>6  Least-Squares Correction Techniques</td>
<td>71</td>
</tr>
<tr>
<td>7  Orientation of the Vehicle Frame</td>
<td>76</td>
</tr>
<tr>
<td>8  Riccati Equation Covariance Propagation</td>
<td>81</td>
</tr>
<tr>
<td>9  Rate Gyro Readout Data Generation</td>
<td>87</td>
</tr>
<tr>
<td>10 Software Documentation</td>
<td>89</td>
</tr>
</tbody>
</table>
1.0 Introduction

This document reports the findings of a three year research project to develop a method for on-board satellite attitude determination. We believe this method can achieve sub-five-arcsecond accuracy when applied to data obtained with a new general purpose star tracker (typical of several existing configurations).

The primary motivation for the research is the exploitation of recently developed light sensitive Charge-Coupled-Devices (CCD) arrays, placed in the focal plane of a tracker lens, to act as a "film" for imaging starlight. Satellite attitude can be determined by identifying the stars detected by the CCD. The star image data, output from the CCD, can be either telemetered to ground for later analysis or, as described in this report, analyzed on-board (via computers configured in parallel) to determine satellite attitude autonomously in near real-time.

The basic system we propose consists of 2 or 3 CCD star trackers and 3 microcomputers, each with a dedicated function. The function of each of the 4 sub-systems is outlined below, with reference to Figures 1.1, 1.2, and 1.3.

1.1 System Overview

(1) CCD Star Sensors and Associated Electronics

Although the development of CCD sensors and trackers is not part of this research, there are several CCD star tracker designs proposed by various organizations involved in hardware development. The purpose of our work has been exploitation of the CCD star tracker technology; we have chosen a particular set of parameters.
Figure 1.1 UVASTAR An electro-optical/software system capable of real time readout of digitized star coordinates, and ultimately, autonomous, near-real time star pattern recognition and attitude determination.
CCD ARRAY (488 x 390 Pixels)  
("Defocused" Negative Image)

Pixel Response to Defocused Starlight

Star Centroid:

\[
\begin{align*}
X_{c_j} &= \frac{I_{x_j} P_{x_j}}{Z_{x_j}} \\
Y_{c_j} &= \frac{I_{y_j} P_{y_j}}{Z_{y_j}}
\end{align*}
\]

Figure 1.2 Formation of Image on the CCD Array.
Figure 1.3

STAR PATTERN RECOGNITION/SPACECRAFT ATTITUDE ESTIMATION

PROCESS A
Digital Image Processing
- Determine image centroids and magnitudes.
- Delete spurious images.
- Apply calibration corrections.

PROCESS B
Discrete Attitude Determination
- Identify subsets of measured stars associated with specific stars.
- Determine the orientation and position of each subset which produces the best agreement between measured and predicted image coordinates.

PROCESS C
Near-Real Time Attitude Estimation
- Motion integration between successiveattitude estimates.
- Kalman filter of attitude at time t.

OUTPUT
- Best estimates of attitude and time derivatives.
- 1000 readouts per minute.
- Measure orthogonal components of angular velocity.
- A/D converter.

60 readouts per min
5 to 10 readouts per min
1 to 2 readouts per min
Several readouts per min

FOV A

FOV B

Pixel response

Pixel response
but it should be kept in mind that these are nominal, achievable values without further CCD/star tracker technology advances. Each of the two (or three) trackers is identical and their boresights are assumed separated by the nominal interlock angle of $90^\circ$. We assume a lens focal length of 70 mm and a CCD array size such that a $7^\circ \times 9^\circ$ field of view (FOV) is imaged onto the array. The arrays are assumed to be Fairchild's 11.4 mm x 8.8 mm matrix consisting of 488 x 380 silicon pixels, each pixel accurately imbedded (to 1 part in 10,000) in a microcircuit chip. Starlight is defocused slightly on the CCD in order to spread typical images over 9 to 16 neighboring pixels. This permits accurate "centroiding" of the image to determine image coordinates accurate to about 10% of a pixel (10% is a conservative estimate). The processing of a data frame consists of a rapid sequential readout of the voltage response of all pixels and an analog to digital (A/D) conversion only of certain pixels (based upon response above an analog threshold level or prior selection). The scans of each field of view are controlled by a common clock and are assumed to represent 2 (or 3) frames (1 from each sensor) taken at the same instant. This assumption is valid for all but very rapidly spinning satellites, since the CCDs can be scanned 10 times per second. (Refer to Appendix I for more star tracker information.)

(2) Microcomputer A

Program Process A is performed by a Microcomputer A with either one computer per sensor or sequential treatment of data for 2 or 3 sensors. Again, Process A is not part of this research program but since the functions to be performed are straightforward, we simply replace Process A by calculating synthetic output data
whose availability is clock controlled. Process A takes as input data the digitized pixel voltages and pixel coordinates for up to 10 stars in each FOV and the associated time. Image centroids are calculated for each image and corrections for lens distortion and other known error sources are applied; a relative magnitude or intensity is also calculated. As output, Process A delivers the focal plane coordinates for each star image. Since Process A calculations for one data frame can be performed in near real time and many times faster than the attitude can be determined, it may be possible and desirable to perform additional editing of the star data. For example, images with rapidly varying image intensity from frame to frame could be eliminated or images whose successive positions are inconsistent with the overall motion caused by vehicle motion (such as images of space debris) could be deleted immediately from consideration (failure to detect and delete all spurious images does not prove fatal, but does slow the pattern recognition logic of Process B). Process A would be expected to output image coordinate and magnitude data at the rate of about 5 frames per sensor each minute and simply overwrite old data. The microcomputer of Process A is considered as an integral part of the star tracker itself, making it a "smart sensor".

Since Process A controls the scan of the CCD and its electronics, it is possible to track only those stars desired (those whose pixel response lies within specified bounds). Thus, even though the CCD array contains thousands of pixels, only a small fraction of their response values need be subjected to A/D conversion and stored at any one time. It is this data compaction feature, along with the
high dimensional stability of CCD arrays that make them so attractive for this application. Also significant is the high speed readout of the CCD which allows one to assume, for most cases, that the star images visible in a given frame have been imaged simultaneously. Therefore, stellar resection (geometric) methods can be used for attitude determination and the vehicle motion can be ignored for analysis of a single frame of data.

(3) Microcomputer B

Data from Process A (and Process C) are analyzed by program Process B; again by means of a dedicated microcomputer. As input, Process B accepts:

- star image coordinates and magnitude data; one set per FOV (from Process A);
- a-priori attitude estimates and covariance, (from Process C); and
- a-priori estimates and covariance of interlock angles between the sensors image planes (from previous analysis of Process B data).

The sequence of calculations/logical decisions divides into two primary functions:

- identify measured stars in each FOV as specific stars contained in an on-board star catalog (containing, in the general case, the direction cosines and instrument magnitudes of the 5000 brightest stars) and
- determine the spacecraft orientation and field of view interlock angles which cause the simulated images of identified catalog stars to overlay the corresponding measured images in a least-squares sense.
These two tasks will be discussed in detail in Section 3 and appendices. The expected output rate for Process B is two or more attitude updates per minute of elapsed time. The old attitude and covariance, output to Process C, are overwritten by each new attitude and covariance.

(4) Microcomputer C

The attitude determined by Process B for a discrete time is further processed by program Process C in microcomputer C. Input to this program consists of the attitude and covariance from Process B and A/D converted gyro rate measurements of angular velocity. The kinematic differential equations governing the spacecraft attitude are integrated forward from the attitude determined from the previous pass through Process C (using the gyro rate measurements). This yields an attitude estimate at the time associated with the next set of image coordinates from Process A. After Process B has determined the discrete attitude it is combined with the integrated attitude in a Kalman Filter calculation to give a best estimate of attitude at the time associated with the star tracker data. Further forward integration gives an estimated attitude and covariance at real-time.

1.2 Focus of This Study

Our primary tasks were to develop the algorithms for Process B (star pattern recognition and attitude determination), and Process C (state integration and Kalman filter routines). This, of course, required some study of CCD arrays and star tracker design (Process A). A governing principle was that this system be suitable for a general purpose satellite; that is, we did not design it with a particular mission in
mind. We have required a slowly rotating satellite, however, in order to insure that star images do not cause streaks in the star camera and that our rate integration be valid (i.e., the vehicle not undergo rapid maneuvers).

The algorithms we devised can run on a large memory (64,000 bytes), general purpose microcomputer. To demonstrate this, we have programmed the algorithms on a Hewlett-Packard 9845S microcomputer equipped with a high level BASIC interpreter language package. Although the processing time with this language is significantly slower than a compiler type of system or machine code program its use permitted programming ease, which was essential for development work. Our tests show the present system will produce updated attitude estimates every 60 seconds (with rate integrated attitude available several times per second) in a steady state mode; when the programs are implemented in a form suitable for a satellite computer they should execute much faster.

We have organized this report to include most of the detail and mathematical developments in appendices in order to keep the body descriptive and concise. Section 2 discusses the coordinate frames and orientation variables used in this study. Processes B and C are described in Sections 3 and 4, respectively. We discuss our truth model and simulation tests of our algorithms in Section 5 and present conclusions in Section 6. The reader is referred to references 1-3 for a discussion of intermediate results of this project.
2.0 Orientation Parameters and Coordinate Frames

In order to describe the orientation of a spacecraft we need to specify some coordinate frame fixed in the vehicle and another fixed in inertial space. In addition, we need a parameter set to describe the relative orientation of these two frames.

Euler angles provide an easily understood description of relative orientation of two frames. The three angles specify a sequence of rotations about three successive coordinate axes of a rotated frame. However, although they are descriptive, Euler angles are not very suitable for our purposes for several reasons. Any of the twelve possible rotation sequences possesses two singularities. In addition, the differential equations describing the rotational motion of a vehicle involve trigonometric nonlinearity when expressed in terms of Euler angles. The same is true of the least-squares equations used in the star pattern recognition algorithms. Extensive use of trigonometric functions will significantly increase the computation time.

2.1 Euler Parameters

These problems have been circumvented by using a set of four variables called Euler parameters instead of Euler angles. Euler parameters have the advantages that (1) they do not have a geometric singularity, (2) they rigorously satisfy linear differential equations, and (3) no evaluation of trigonometric functions need be done in any application discussed herein. One disadvantage is the four parameters must sum-square to unity; we have found methods to include this constraint in our estimation algorithms.

Euler parameters ($\beta_0, \beta_1, \beta_2, \beta_3$) can be interpreted geometrically in terms of Euler's theorem: A completely general angular displacement
of a rigid body can be accomplished by a single rotation (the principal angle, $\phi$) about a line (the principal line, $\hat{\ell}$) which is fixed relative to both arbitrary body-fixed axes $\{b\}$ and reference axes $\{n\}$. If $\{n\}$ is initially coincident with $\{\hat{b}\}$, then the direction cosines $(\ell_1, \ell_2, \ell_3)$ of $\hat{\ell}$ with respect to $\{\hat{n}\}$ and $\{b\}$ are identical.

The Euler parameters are then related to the principal rotation parameters as follows:

$$\beta_0 = \cos \frac{\phi}{2}$$

$$\beta_i = \ell_i \sin \frac{\phi}{2}, \quad i = 1, 2, 3.$$  

Note that Euler parameters satisfy the constraint:

$$\sum_{i=0}^{3} \beta_i^2 = 1.$$  

The rotation matrix $[C]$ characterizing the relationship between a body fixed frame $\{b\}$ and a reference frame $\{n\}$ by: $\{b\} = [C]\{n\}$ can be written in terms of Euler parameters as:

$$[C] = \begin{bmatrix}
\beta_3^2 + \beta_1^2 - \beta_2^2 - \beta_3^2 & 2(\beta_1 \beta_2 + \beta_0 \beta_3) & 2(\beta_1 \beta_3 - \beta_0 \beta_2) \\
2(\beta_1 \beta_2 - \beta_0 \beta_3) & \beta_1^2 - \beta_2^2 + \beta_3^2 & 2(\beta_2 \beta_3 + \beta_0 \beta_1) \\
2(\beta_1 \beta_3 + \beta_0 \beta_2) & 2(\beta_2 \beta_3 - \beta_0 \beta_1) & \beta_1^2 - \beta_2^2 - \beta_3^2 + \beta_3^2
\end{bmatrix}$$

2.2 Coordinate Frames

For convenience we have used several coordinate frames for Process B and C algorithms. They are: the inertial frame, $N$, the gyroscope frame, $G$, the vehicle frame, $V$, and two camera frames, $A$ and $B$ (Table 2.1).

The inertial frame is our primary reference frame and is defined, essentially, by star positions. The locations of all the stars in the
Table 2.1

COORDINATE FRAMES

Inertial Frame (N): Primary reference frame. Used for star positions and vehicle velocity components.

Gyroscope Frame (G): Defined by orientation of three orthogonal gyroscopes. Rotation rate of the vehicle is measured in this frame.

Camera "A" Frame (A): Defined by orientation of camera boresight and focal plane.

Camera "B" Frame (B): Defined by orientation of camera boresight and focal plane.

Vehicle Frame (V): Defined by boresight unit vectors of the "A" and "B" frames. Orientation of this frame with respect to the "N" frame is determined by Process B.

RELATIONSHIPS BETWEEN FRAMES

G - N: Changes as vehicle rotates.

V - N: Changes as vehicle rotates.

V - G: Assume this varies slowly with time. Gyro bias terms compensate for small, slow variations.

B - A: Assume this varies slowly with time. Interlock parameters are monitored by Process B.
onboard catalog are specified in this frame, as are the vehicle velocity components (used for aberration corrections). The gyro frame is defined by the axes of three orthogonal gyroscopes, fixed in the vehicle. Each gyroscope gives a measure of the vehicle rotation rate about that axis (these are the rates integrated by Process C). Our simulation studies have been configured for a nominally earth pointing spacecraft. Accordingly, we have specified that the unit vectors \{g_i\} along the gyro axes be oriented such that \(g_3\) is along the radius vector, \(g_2\) is perpendicular to the orbit plane and then \(g_1 = g_2 \times g_3\) (nominally along the velocity vector).

The two camera frames, A and B, are assumed fixed to the vehicle and, therefore, maintain a fixed orientation with respect to the gyro frame. We have specified that \(a_3\) and \(b_3\) coincide with the camera boresights and point 45° from the direction of vehicle motion, above and below the orbit plane. Unit vectors \(a_1\) and \(b_1\) lie along the \(x\) axis of the CCD of each camera and lie in the orbit plane while \(a_2\) and \(b_2\) form the \(y\) axis of each CCD.

The \(V\) frame has been defined by the boresight vectors, \(a_3\) and \(b_3\) (see Figure 2.2):

\[
\begin{align*}
v_1 &= \frac{(a_3 + b_3)}{|a_3 + b_3|} \\
v_2 &= v_3 \times v_1 \\
v_3 &= \frac{(a_3 \times b_3)}{|a_3 \times b_3|} .
\end{align*}
\]

Both Processes B and C have been formulated to employ the Euler parameters which orient this vehicle frame with respect to the inertial frame.

There are several advantages to this definition of the \(V\) frame. First, the boresight vector of each frame is well determined compared...
Figure 2.1. Relationship of the vehicle frame to FOV(A) and FOV(B)
with the rotation about the boresight vector. Thus, the V frame orientation is not affected by the poorly known quantities. Second, by using this definition we weight frames A and B equally.

Although frames G, A and B are nominally fixed with respect to the vehicle, in reality these interlock relative orientations will vary due to thermal cycling, vehicle vibrations, etc. Therefore, we have included techniques in Processes B and C, to be discussed in later sections, to monitor and/or partially correct for these interlock variation effects. A by-product is the attractive feature that the system becomes fully self-calibrating.
3.0 Process B

Attitude determination by processing image coordinates obtained from Process A depends upon the ability to describe mathematically the location of a star image on the CCD image plane, given its direction in space and the orientation of the star tracker. This mapping, a function of the Euler parameters discussed in Section 2, is described by the stellar colinearity equations which, for frame A, have the form:

\[
\begin{align*}
x &= f \frac{L_1 A_{11} + L_2 A_{12} + L_3 A_{13}}{L_1 A_{31} + L_2 A_{32} + L_3 A_{33}} + x_0 \\
y &= f \frac{L_1 A_{21} + L_2 A_{22} + L_3 A_{23}}{L_1 A_{31} + L_2 A_{32} + L_3 A_{33}} + y_0
\end{align*}
\]

where \( f \) = lens focal length, assumed to be constant,

\( A_{ij} \) = elements of the coordinate frame rotation matrix \( A_N \), in turn a function of Euler parameters, and

\( L_i \) = star direction cosines for the particular star as measured in the \( N \) frame

\((x_0, y_0)\) = principal point offsets.

If there are several stars in a single field of view (FOV) we seek to minimize the sum of the squares of the residuals between measured star images and predicted coordinates for the same stars. This is accomplished by adjusting the Euler parameters, which orient the star tracker frame, using a least-square differential correction scheme.

Before outlining the least-square procedures, we describe the process of equating particular catalog stars with measured stars. To start Process B we need an estimate of the camera orientation. This can be provided by either the results of a previous pass through Processes B and C or from some indirect method such as horizon sensors. This...
estimate is needed to acquire a "sub-catalog" from the mission catalog and must be sufficiently accurate so that the subcatalog contains the measured stars.

3.1 Star Catalog

As part of our work on Process B, we have converted the visual magnitude of over 5,000 stars to a standard infra-red (I) magnitude. For simplicity, we assumed the instrument magnitude is identical to the I magnitude (which could be arranged by using an I filter). In specific applications instrument magnitude would probably be based upon laboratory calibration. We have not precessed the star positions nor corrected for proper motion since these tasks would best await an actual flight test of the system. Details of magnitude conversions are discussed in Appendix 2.

In addition, we developed a star catalog format for easy access. The celestial sphere is divided into cells or segments in an orderly pattern so that any cell can be accessed easily to obtain the positions of stars contained within. It is important to keep in mind that our catalog segmentation and access logic were designed for a general mission. Simplified catalogs could be designed for specific missions. See Appendix 3 for more details on the cell structure and access logic.

3.2 Star Pairing

Associating catalog and measured stars begins by sorting catalog stars by angular distance off the apriori estimated FOV boresight direction. By computing the vector dot product of each star with the boresight vector we have a suitable measure of angular distance and can sort stars according to this parameter (and thereby avoid repetative
angle calculations from inverse trigonometric functions). The next step is to compute and store in a table, the cosine of the interstar angle for all possible pairs of measured stars. We then pair catalog stars, beginning with stars nearest the estimated boresight, compute the cosine of the interstar angle, and then compare this value with each value for the measured pairs (refer to Appendix 4). This process is repeated until either a match is found to within some tolerance or the list of catalog stars is exhausted. In the latter case we start over with fresh data from Process A and a new estimate of orientation. However, if a match is found, we tentatively assume the catalog pair is the same as the measured pair. We are now ready to adjust the estimated orientation parameters via least-squares correction to get the two projected catalog stars to overlay the two measured stars. Because of the relatively high probability of finding an invalid star pair match, attitude confirmation requires additional star matches, as discussed below. We must also account for the effect of stellar aberration on the star direction cosines (refer to Appendix 5).

3.3 Least-Squares Correction

The non-linear relationship between the Euler parameters and star image coordinates requires an iterative least-squares correction procedure to find the best estimate of vehicle orientation. Basically, at each iteration we require the Euler parameter corrections to minimize:

\[(\Delta X - A \Delta \theta)^T W (\Delta X - A \Delta \theta)\]  \hspace{1cm} (3.2)

where \(\Delta X\) is a column vector of x and y coordinate residuals between the measured and predicted images (using the current values for the orientation variables), A is a matrix of partial derivatives of star positions.
(the stellar colinearity equations) with respect to current Euler parameters, \( \Delta \beta \) is the correction vector to be added to the current parameters and \( W \) is a weight matrix. The derivation of this equation is found in Appendix 6.

Since the Euler parameters must satisfy a constraint equation, it is necessary to guarantee that the corrected Euler parameters also satisfy this constraint. If we express the constraint equation as

\[
\beta^T \beta = 1,
\]  

then, after correcting the parameters, the corrections \( \Delta \beta \) must satisfy:

\[
(\beta + \Delta \beta)^T (\beta + \Delta \beta) = 1.
\]  

Expanding to first order we have:

\[
\beta^T \beta + 2 \beta^T \Delta \beta = 1 + \text{residual}
\]  

and by writing this as

\[
(1 - \beta^T \beta) - 2 \beta^T \Delta \beta = \text{residual}
\]  

we can append \( 1 - \beta^T \beta \) to the \( \Delta X \) vector, \( 2 \beta^T \) to the \( A \) matrix and \( \Delta \beta \) is again the correction vector. In solving Eq. (3.2) we assign a large weight to this constraint equation in order to insure that it is satisfied (i.e., the residual will be essentially zero).

After the vector of Euler parameters has been found by iteration, it is necessary to confirm whether or not the catalog pair is indeed the measured pair (i.e., whether we have the correct orientation). Each catalog star is mathematically projected onto the focal plane and tested to see if it lies near a measured star. A match of three or more stars is considered a positive outcome; a match of only two stars (most likely the initial pair) or fewer constitutes failure and we continue with star
pair matching to find another pair. In the present software version we accept up to 5 catalog stars which match measured stars in one FOV.

The star pair matching and confirmation calculations described above are performed separately for each FOV. If the outcome for each FOV is positive, we have up to 5 measured stars from each FOV with their corresponding catalog positions. All of these stars are used to correct the orientation again and, in addition, to correct the Euler parameters defining the interlock relationship between the two FOV. We again minimize:

$$(\Delta X - A \Delta B)^T W (\Delta X - A \Delta B).$$

Now, $\Delta X$ contains the residuals for all images, the $A$ matrix contains partial derivatives with respect to both the Euler parameters orienting the vehicle frame and those orienting frame $B$ with respect to $A$, and $\Delta B$ contains corrections to these same Euler parameters. As before, we append two constraint equations, one for each set of Euler parameters, to the matrix equation. (Refer to Appendix 7 for details of this procedure).

This method yields an accurate $\beta_{VN}$ vector compared with $\beta_{BA}$. The $\beta_{VN}$ describe the orientation of the vehicle frame which, in turn, is determined by the FOV boresight vectors, both usually well determined. On the other hand, the $\beta_{BA}$ are effected by the relatively poorer determination of the roll angle about the boresight vector of each FOV. Therefore, we have found it desirable to further process $\beta_{BA}$. We assume the true $\beta_{BA}$ vary slowly (due to such things as thermal cycling) and write:

$$\dot{\beta}_{BA} = 0 \tag{3.7}$$

and then combine the apriori or predicted values of $\beta_{BA}$ (obtained from a previous analysis) with the calculated values of $\beta_{BA}$ obtained via
least-squares. The two vectors of $\beta_{BA}$ are combined using a discrete Kalman filter (see Appendix 6 for details). This method can be used, with proper tuning, to monitor the interlock variations and give the system "memory" of past interlock determinations.

We note that the least-squares method for two FOV and the Kalman filter calculations involve considerable mathematics, such as matrix multiplication and matrix inversion, which adversely affects execution time. However, it is important, we feel, to provide the option to calibrate (as often as necessary) the interlocks between camera frames. By monitoring these variations we can make the system self-calibrating and can tolerate modest lack of mechanical stability in the various interlocks.
4.0 Process C

Process C software has two primary functions: (1) integrate the kinematic differential equations describing the satellite motion over a short time interval in order to provide Process B with a new attitude estimate, and (2) combine this integrated orientation with the orientation determined by least-squares in Process B. The second function is performed via a discrete Kalman filter to yield an optimal estimate of the orientation at a particular time.

4.1 Kinematic Equations

The differential equations describing the kinematics of a rotating coordinate frame with respect to a fixed frame, expressed in terms of Euler parameters, are:

\[
\begin{bmatrix}
\dot{\beta}_0 \\
\dot{\beta}_1 \\
\dot{\beta}_2 \\
\dot{\beta}_3
\end{bmatrix} = \frac{1}{2} \begin{bmatrix}
\beta_1 & -\beta_2 & \beta_3 \\
\beta_2 & \beta_0 & -\beta_3 \\
\beta_3 & \beta_1 & \beta_0
\end{bmatrix} \begin{bmatrix}
\omega_1 \\
\omega_2 \\
\omega_3
\end{bmatrix} = \begin{bmatrix}
[\beta]\{\omega\}
\end{bmatrix} \tag{4.1}
\]

where \{\beta\} are the Euler parameters orienting the frame and \{\omega\} are the gyro rates measured in that frame, along the 3 orthogonal axes.

In our model we prefer to use the Euler parameters orienting the vehicle frame, V, with respect to the inertial frame, N. Therefore, the
gyro rates, measured in the G frame, must be transformed into the V frame via matrix \( V_G \), which we assume to be constant. In addition, the measured gyro rates, \( \{\ddot{\omega}\} \), contain noise terms and other effects such as errors due to nonorthogonality of the gyroscopes, variations in the V-G interlocks, and gravity or magnetic effects. We account for these effects, to first order, by absorbing all except gyro noise into bias terms, \( \{b\} \), one per axis. The equations become (using letter subscripts to denote the appropriate coordinate frame relationships):

\[
\{\dot{\beta}_{VN}\} = [\beta_{VN}][V_G](\ddot{\omega}_{GN} - b_{GN})
\]

\[
= [\ddot{\omega}_{VN}]{\beta}_{VN} - [\beta_{VN}][V_G]{b}_{GN}
\]

where \( \{\ddot{\omega}_{GN}\} = \{\omega_{GN}\}_{\text{true}} + \{b_{GN}\} + \{\text{noise}\} \). The gyro biases are assumed to be slowly varying; this allows us to write:

\[
\begin{bmatrix}
\dot{b}_1 \\
\dot{b}_2 \\
\dot{b}_3
\end{bmatrix} = 0
\]

which is valid over short time intervals. The set of seven differential equations (for the four Euler parameters and three biases) can be integrated via Runge-Kutta methods to yield a new orientation estimate for Process B.

4.2 Kalman Filter Equations

The two estimates of vehicle attitude, one from integration of the kinematic equations and the second from Process B attitude estimation, are combined to give a best estimate of the attitude. We have adopted the discrete Kalman filter equations for a linear system:
\[
\hat{x}_{k+1}(k+1) = \bar{x}_k(k+1) + K(k+1)(\hat{y}(k+1) - \bar{y}(k+1))
\]

\[
P_k(k+1) = P_k(k) + \int_{t_k}^{t_{k+1}} \beta \, dt \tag{4.5a-d}
\]

\[
K(k+1) = P_k(k+1)H^T(k+1)\left[L_{V_{k+1}V_{k+1}} + H(k+1)P_k(k+1)H^T(k+1)\right]^{-1}
\]

\[
P_{k+1}(k+1) = [I - K(k+1)H(k+1)]P_k(k+1)
\]

where

\(\hat{x}_{k+1}(k+1)\) = optimal estimate of the state \(X\) at time \(t_{k+1}\) based on \(k + 1\) data sets,

\(\bar{x}_k(k+1)\) = state \(X\) at time \(t_{k+1}\) based on \(k\) data sets, and calculated from forward integration of kinematic equations,

\(K(k+1)\) = Kalman gain matrix for time \(t_{k+1}\),

\(\hat{y}(k+1)\) = \(N\) = Values of \(\beta_{VN}\) from Process B and bias values from previous iteration,

\(\bar{y}(k+1)\) = \(\begin{bmatrix} \beta_{VN} \\ \beta_{GN} \end{bmatrix}\) = Values of \(\beta_{VN}\) from integration and bias values from previous iteration,

\(P_i(j)\) = 7 \times 7 covariance matrix at time \(t_j\) based on \(i\) data sets,

\(H(k+1)\) = \(\frac{\partial \hat{y}}{\partial x} \bigg|_{t_{k+1}}\) = I (for our case),

\(L_{V_{k+1}V_{k+1}}\) = covariance matrix associated with measurement of the state (or observations) from Process B (the upper left 4 \times 4) and covariance for the biases (the lower right 3 \times 3), and

\(\int_{t_k}^{t_{k+1}} \beta \, dt\) = integration of the matrix Riccati equation for covariance propagation (see Appendix 8).
Notice that we have included the biases as observables in our Kalman filter equations. By choosing the appropriate covariance values in matrix L and P, we can control the corrections to the biases. We have done this because we hypothesize that the biases vary slowly—or at least the effects which we are most interested in monitoring vary slowly. Our simulation tests indicate that with this formulation we can follow the bias terms added to the rate gyro data and absorb variations in the interlock matrix VG into the bias terms with little degradation in the optimal state estimate.
5.0 Truth Model and Simulation Tests

The algorithms for Process B and C were tested by processing data produced by a simulation program ("truth model") and then comparing the results with the "true" model. Each test consisted of processing a series of 29 data frames separated by 30 seconds of satellite motion. The most important input data are the image coordinates and intensities of the stars in each field of view from Process A and the most important output data are the calculated orientation from Processes B and C. The simulation program was written to include variations in several important parameters such as Euler parameters describing the relative orientation of the two camera frames, Euler parameters for the rotation from the gyro to vehicle frames, and gyro bias terms. To illustrate the performance of our algorithms for this report each series included the parameter variations of the previous model plus only one additional parameter variation.

5.1 Simulation Program

We first describe briefly the creation of simulated data. The first step is to choose an appropriate satellite orbit, specified by its semimajor axis, orbital period and inclination. To facilitate the calculation of satellite position and velocity, we make use of Herrick's two body solution (see Ref. 6, p. 155). To use this method we specify the initial position and velocity components, expressed in the inertial frame, and the associated time. All later positions and velocities can be determined by specifying the desired time and solving several equations. This same method is used to determine the earth's position and velocity at each time step. Velocity data are needed to calculate the aberration of starlight which affects the apparent star directions.
All of our tests have assumed a circular satellite orbit and a nominally earth pointing vehicle. To reflect this choice we initially orient the gyroscope frame so that the $g_2$ axis is perpendicular to the orbit plane and the primary vehicle rotation is about that axis. The $g_3$ axis is initially along the orbit radius vector, $r$, and $g_1$ is given by $g_2 \times g_3$. Since our primary orientation variables are $\beta_{VN}$, as discussed in Section 2, we obtain their initial values as follows: specify the initial values for $\beta_{VG}$ and calculate the rotation matrix $V_G$, then use the gyro unit vectors $\{\mathbf{g}\}$ to fill matrix $G_N$ and calculate $V_N = V_G \cdot G_N$; $\beta_{VN}$ can then be recovered from $V_N$. All subsequent values of $\beta_{VN}$ are obtained by integrating the kinematic differential equations forward in time. The gyroscope rate history, needed for the integration, is given for the G frame; therefore, the rates are rotated into the V frame by matrix $V_G$, a function of $\beta_{VG}$, which can be either constant or time varying. See Appendix 9 for details of rate gyro data simulation.

At each time step we calculate the $V_N$ matrix from $\beta_{VN}$. Matrix $B_A$ is computed from $\beta_{BA}$ (again, constant or time varying parameters) and from $B_A$ we compute $A_V$ (see Appendix 7). The last row of $A_N = A_V \cdot V_N$ is the FOV(A) camera boresight unit vector, needed to access the star catalog for a subcatalog of stars. After adding the effects of aberration, the stars are projected onto the CCD image plane via the stellar collinearity equations. Stars seen by the second camera are obtained in the same manner, after first computing $B_N = B_A \cdot A_N$ to get the boresight unit vector.

The image coordinates obtained by the above methods are assumed to represent the "true" state. In an actual system Process A will not,
of course, produce the true image coordinates. We have assumed that the
centroiding of an image can be performed to an accuracy of 10% of a
pixel (1-sigma error) and that systematic errors such as image distortion
can be accounted for and removed. Therefore, we perturb the true image
coordinates with Gaussian noise.

Various data are stored on tape or disk for later analysis by
Process B and C. Space is left at the end of each record (one record
per frame) for data computed by Processes B and C; these are later
analyzed for accuracy and displayed.

5.2 Simulation Tests

A set of eight models was used to test our algorithms. All models
followed the same orbital path and rotation history. Of the 29 data
frames, each consisting of image coordinates in a pair of FOV and
separated by 30 seconds of flight time, only once does a FOV contain
2 stars (the case at 8 minutes from the start). In that case, the
least-squares solution used only the stars from one FOV; the orientation
errors for this case are relatively large in all models. For display
purposes, we have plotted the root-mean-square of the angular errors
between the calculated and true vehicle frame (using the 1-2-3 Euler
angle set). There are actually three calculated frames: the result
of Process B least-squares, the integrated state, and the optimal
estimate from the Kalman filter. Each series started with an estimate
about 2 degrees in error. Thus, several frames must be processed for
the system to reach steady state.

No noise or parameter variations were included in the first
model in order to verify that the software could indeed recover the
true state (Figure 5.1). Gaussian noise added to the rate gyro data (1 sigma = 1 arcsecond/second) causes the integrated state of our second model to deviate from the true state (Figure 5.2). It is evident that with this level of noise, the state could be integrated several minutes, at least, before the accumulated error would place the estimated state too far from the true state. Thus, Process C provides adequate backup for failures of Process B. Notice also that the optimal estimate nearly matches the Process B state; this is due to the significantly smaller covariance associated with the Process B result.

Our third test included noise in the image coordinates (1 sigma = 0.0034 mm) corresponding to approximately 10 arcsecond error in determining a star's direction (lens focal length = 70 mm). Once again the optimal estimate is nearly the Process B result (Figure 5.3). This and all following simulation models show that the most important factor affecting the vehicle attitude determination accuracy is the accuracy with which individual star centroids can be determined. A reduction of centroid errors will produce a proportional reduction in orientation errors. Since the least-squares result nearly matches the optimal estimate, improving the star tracker performance will yield the most improvement in attitude estimation. Our choice of 10% pixel centroiding error for each star, yielding about 5 arcsecond vehicle pointing error, is considered conservative. Indications are that 5% pixel error can be obtained routinely, with perhaps even smaller errors for brighter stars (considerable research is presently under way to determine optimal "tuning" of the sensor and centroiding process—clearly an appropriate scale factor can be applied to our results to reflect other centroiding error models).
The fourth model included variations in the Euler parameters describing the interlock between star tracker frames. As expected, this does not seriously degrade the orientation of the vehicle (V) frame since its orientation is defined by the boresight unit vectors (Figure 5.4a). It will be recalled that Process B algorithms estimate these interlock parameters. Figures 5.4b-d display the estimates obtained for the three interlock angles. Each figure shows the deviations, from the nominal interlock angle, of the true angle, the angle calculated from least-squares and the best estimate of the interlock angle. This series used a value 5 arcseconds for the variance of process noise matrix in the Kalman filter calculations. The fifth series was identical to the forth series except we used a value of 10 arcseconds. Results of Figure 5.5a indicate little effect on the vehicle frame orientation while Figures 5.5b-d show improved interlock recovery compared with Figure 5.4b-d. The value used for the process noise should be influenced by the size of the expected variations in interlocks. A strict value (small noise) prohibits the algorithm from following a true variation while a large value leads to large fluctuations in the interlocks and no meaningful self-calibration.

The next several simulations concern Process C performance. First we added a time varying bias term to each gyro axis in order to test how well Process C algorithms recover and follow each bias. Figure 5.6a indicates the biases degrade the integrated state only slightly once the biases have been recovered (after several minutes). Figure 5.6b shows the true and calculated bias values. By choosing a different value for the bias variance (see Section 4) we can control the fluctuations in the recovered biases. To demonstrate this, in our seventh
series we increased the variance square root from 0.5 to 1.0 arcsecond/second. Figure 5.7a shows no effect on the vehicle frame determination while Figure 5.7b indicates a faster bias recovery but somewhat larger bias fluctuations compared with Figure 5.6b.

Our final simulation test included the effects of time-varying and off-set Euler parameters, \( \beta_{VG} \), describing the relationship between the vehicle and gyro frames. Process C algorithms assume this relationship is fixed (in our case rotation matrix \( VG \) is the identity matrix) so any deviation will appear, over the short interval, as simply an additional bias term in the gyroscopes. Thus, we see little effect in the V frame errors (Figure 5.8a) but notice the recovered bias values are displaced somewhat from their previous tracks (Figure 5.8b, variance square-root is 0.5 arcsecond/second). We assume that other slowly varying or constant effects will be accounted for in like manner.

The choice for the bias variance should be influenced by the expected variations in the gyro biases as well as an estimate of variations in other elements. As with the FOV interlock weight, a strict value (small variance) restricts the tracking of a true variation while a liberal value negates the self-calibrating nature of the algorithms.
Figure 5.1: Orientation errors of vehicle frame for simulation test with no rate gyro errors and no image centroid errors.
Figure 5.2: Orientation errors of vehicle frame for simulation test with noise added to rate gyro data. ($\sigma = 1$ arc sec/second).
Figure 5.3: Orientation errors of vehicle frame for simulation test with image centroid errors. (σ = 0.0034 mm).
Figure 5.4a: Orientation errors of vehicle frame for simulation test with time varying interlock angles between camera frames. (process noise = 5 arcseconds)
Figure 5.4b: Deviations of the first Euler angle (3-1-3 set) from the nominal value (O) -- true angle, + = least-squares estimate, * = Kalman filter estimate.
Figure 5.4c: Deviation of the second Euler angle (3-1-3 set) from the nominal value (90°): - = true angle, + = least-squares estimate, * = Kalman filter estimate.
Figure 5.4d: Deviation of the third Euler angle (3-1-3 set) from the nominal value (0°): - = true angle, + = least-squares estimate, * = Kalman filter estimate.
Figure 5.5a: Orientation errors of vehicle frame for simulation test with time varying interlock angles between camera frames. (process noise = 10 arc seconds)
Figure 5.5b: Deviations of the first Euler angle (3-1-3 set) from the nominal value. (°): = true angle, + = least-squares estimate, * = Kalman filter estimate.
Figure 5.5c: Deviation of the second Euler angle (3-1-3 set) from the nominal value (90°): = true angle, + = least-squares estimate, * = Kalman filter estimate.
Figure 5.5d: Deviation of the third Euler angle (3-1-3 set) from the nominal value (0°): - = true angle, + = least-squares estimate, * = Kalman filter estimate.
Figure 5.60: Orientation errors of vehicle frame for simulation test with time varying gyro bias terms.
Figure 5.6b: True and calculated bias terms for each gyro axis (bias process noise standard deviation = 0.5 arc sec/second).
Figure 5.7a: Orientation errors of vehicle frame for simulation test with time varying gyro bias terms.
Figure 5.7b: True and calculated bias terms for each gyro axis (bias process noise standard deviation = 1.0 arc sec/second).
ANGULAR ERRORS IN ORIENTATION

+ = Integrated state - True state
* = Process B state - True state
~ = Optimal state - True state

Figure 5.8a: Orientation errors of vehicle frame for simulation test with time varying interlock angles between the gyro and vehicle frame.
Figure 5.8b: True and calculated bias terms for each gyro axis (bias process noise standard deviation = 0.5 arc sec/second).
6.0 Conclusions

The simulations discussed in Section 5 illustrate that our algorithms can routinely yield 5 arcsecond accuracy for the assumed star tracker configuration and using up to 5 stars in each field of view. There will be occasional errors greater than 5 arcseconds because of too few stars in one or both FOV. Therefore, we consider our quoted accuracy to represent a one sigma error.

We have not demonstrated explicitly that the necessary calculations can be carried out rapidly enough to yield a new attitude estimate every 30 seconds, as planned. However, this can be accomplished, we believe, simply by converting our algorithms from an interpreter to compiler type of computer language. Such a change would probably reduce computer time by a factor of 5 to 10 (from roughly 60 seconds per frame to less than 12 seconds).

There are several features of our algorithms which need special emphasis. First, it is important to keep in mind that our algorithms assume a slowly rotating satellite. Our algorithms are designed to determine the vehicle attitude provided there is an attitude estimate which is within, say, 5-10 degrees of the truth. In a steady-state mode, we can integrate rate-gyro data between successive frames and thereby provide sequential estimates. However, there must be some system such as horizon sensors to provide a rough attitude estimate either in case of start-up, after vehicle maneuvers, and perhaps after successive failures of Process B.

Also, several parameters must be chosen after a real system is designed or assembled. Two of these are the variance for the Kalman
filter interlock estimation and the gyro bias variance for the Kalman filter bias estimation; both of these are important for the self-calibrating features of our algorithms. We have not attempted to provide the extensive error checking capability needed on a flight system since it is impossible to foresee the many types of failures or errors encountered in a real-life situation.

There are, of course, many possible modifications and additions which could be made. One obvious modification is to reduce the number of stars used in the least-squares correction from the current 5 to perhaps 3 or 4 per FOV. Obviously, this will reduce the attitude accuracy but would have the advantage of reducing computing time and memory requirements. Perhaps with some additional logic the 3 or 4 stars most widely distributed over the FOV could be selected and thereby lessen the impact of fewer stars. An improvement in attitude estimation could be obtained by using a longer focal length lens for each star tracker. This reduces the pointing error to each star due to centroiding errors and therefore improves orientation estimates. However, such a change reduces the field of view and the number of stars detected (unless a larger lens is used to detect fainter stars, requiring a larger catalog as well).
7.0 REFERENCES


Appendix 1: CCD Star Tracker

An important element of the attitude determination system discussed in this report is the CCD star tracker. In this appendix we discuss some of the key features of such a star tracker.

A CCD array has a high degree of dimensional stability and is relatively immune to magnetic effects. These features make it very attractive for a star tracker. For a general purpose tracker the field size is typically 5° to 10° wide and with CCD array sizes currently available, the star images would be a fraction of a pixel in diameter. By defocusing the camera lens slightly so a typical image covers a 3 x 3 array of pixels an image centroid can be computed with at least 10% pixel accuracy. By applying a stored correction function, this error can be reduced still further.

Two other factors affecting centroid accuracy are pixel response variations and photon noise. Response variations can be corrected via a-priori calibration. However, due to time and computer memory limitations only the most severe variations would be corrected in practice. Cooling the CCD reduces thermal noise but the photon noise is always present, affecting fainter images more than brighter images. It is expected that a star tracker with sophisticated software could determine centroids with an accuracy of 5% of a pixel (1o) for minimum brightness stars and perhaps 2-3% error for the brightest stars.

To prevent the star images from smearing on the CCD due to vehicle motion, the exposure time or integration time must be kept short. On the other hand, the analog voltage response from star illuminated pixels must be accurately converted to a digital value, a relatively slow process. Several techniques can be employed to improve read out speed.
The first is to trigger the analog to digital (A/D) conversion only for preselected pixels or those that exceed a minimum threshold. Preselection can be done after a set of stars has been located from the previous frames, while the triggering method can be instituted in a search mode. A second speed gain can be achieved by line-skipping—skipping the read-out of rows of pixel responses after they have been transferred from vertical registers into the horizontal register. This technique can be employed in the track mode once a set of stars has been located.

We anticipate that Process A will be able to received multiple frames of data from the star trackers before Process B is ready to accept new data. This may allow Process A time to edit the data such as predicting, crudely, where the stars may appear in the next frame and/or providing Process B with some average position for each star in a frame. This latter technique could improve the projected accuracy by averaging out some random position errors.
Appendix 2: CCD Instrument Response and Stellar Magnitude Conversion

As mentioned in Section 1, the outputs of Process A are the interpolated centroids and instrument magnitudes for each valid star image. The purpose of this Appendix is the discussion of the approximate techniques utilized in the synthesis of these two outputs.

A2.1 The Star Centroids

The centroid location \((x_c, y_c)\) is given by

\[
x_c = \frac{\sum_{i,j} x_i R_{ij}}{\sum_{i,j} R_{ij}}
\]

(A2.1)

and

\[
y_c = \frac{\sum_{i,j} y_j R_{ij}}{\sum_{i,j} R_{ij}}
\]

(A2.2)

where \(R_{ij}\) is the A/D converted response level of the pixel located at \((x_i, y_i)\) and the summations are over the square array of pixels illuminated by the defocused star image (9 to 16 pixels).

Typical cell size for a CCD is approximately 0.030 mm on a side. A CCD placed behind a 70 mm focal length lens (proposed for one CCD star-sensor) gives a resolution of approximately 1.5 arc-minutes for a focused image. When spread over a 3 x 3 or 4 x 4 cell pattern, the resolution with which the centroid can be located has been found to be \(\leq 6\) arc-sec. For double stars, Process A would produce image coordinates for a single star but with poorly determined image coordinates (a weighted mean of the two stars).

Detections of double stars should not be used in Process B since they would result in poor orientations. One solution to this problem
is to delete from the mission catalog all star pairs with separations less than some tolerance (~6 arc-min. in this case). There are sufficient stars in the catalog that this deletion should not seriously degrade performance. Some additional time in Process B will be used trying unsuccessfully to pair measured double stars with catalog stars, but since detection of double stars will be a relatively rare event, this time penalty should not be a significant practical problem.

A2.2 CCD Magnitude Response

A2.2.1 Magnitude Conversion

Due to both the different spectral qualities of various stars and the peculiarities in the unfiltered CCD response (the primary sensitivity is to red or near infra-red radiation) two stars of the same visual (V) magnitude (for example) may cause different CCD response. Hence none of the cataloged star magnitudes may be used directly. Rather, the magnitudes of the stars must be properly transformed (using the spectral properties contained in the master star catalog SKYMAP) prior to insertion in the mission catalog. Simply stated, the mission catalog must contain an "instrument magnitude" for each star.

It has been decided to convert the V magnitudes of SKYMAP to I magnitudes and utilize an I filter placed over the CCD array. The following points support this decision:

(1) Given the information in SKYMAP, one could, in principle, perform a magnitude conversion from V to a CCD magnitude. However, this would require the choice of a particular CCD detector in order to determine its response characteristics in the laboratory.
(2) The response functions of typical CCD's are quite broad, a fact which makes a rigorous conversion to a CCD magnitude difficult in light of the complicated stellar spectral features in the blue wavelength region. A detailed description of the spectra would be required. The I filter, on the other hand, is confined to the red wavelengths where the star spectra are relatively smooth.

(3) The I filter response peak is near that of typical CCD's. In addition, it overlaps the main peaks of the commonly accepted "typical CCD response" (Ref. 6). Hence, an I filter placed over the CCD array would serve to limit the wings of the CCD response and still provide adequate through-put for sensitivity.

(4) Information exists for converting V magnitudes to I magnitudes. The transformation requires only spectral type and luminosity class - both readily available from SKYMAP.

In the ideal case, the set of detectable stars exactly matches the catalog. Since this is not possible, it is desirable to maximize the completeness of the catalog to some rather faint magnitude to insure that most detectable stars are contained. It is important to note that stars which are faint in V may be relatively brighter at red wavelengths. As will be demonstrated in the next sections, a limit of magnitude 5 in I seems to be a reasonable limit for the CCD configuration assumed for the present study. The 8th magnitude limit of SKYMAP assures that the mission catalog listing will be sufficient.

A2.2.2 The I Magnitude Conversion Method

Two external items of information are needed for the magnitude conversion. Values of $V_{\text{abs}} - I_{\text{abs}}$ were obtained from Johnson [7], whose
table contains listings for three luminosity classes (I, III & V - super giants, giants and main sequence) and extends over most spectral types. The second value needed is $a_I/a_V$, the ratios of absorption in I to absorption in V, expressed in magnitude and as a function of spectral type by

$$a_I = \frac{\log_{10}[(\int_0^\infty I(\lambda)E(\lambda)(1 - I(\lambda)d\lambda))/\int_0^\infty I(\lambda)E(\lambda)d\lambda)]}{a_V = \log_{10}[(\int_0^\infty V(\lambda)E(\lambda)(1 - I(\lambda)d\lambda))/\int_0^\infty V(\lambda)E(\lambda)d\lambda)]}$$

Where $I(\lambda)$ and $V(\lambda)$ are the filter responses, $E(\lambda)$ is the star energy function and $I(\lambda)$ is the relative absorption function and $\lambda$ is the wavelength. (See SKYMAP description [5] for details). As pointed out in SKYMAP, this ratio is nearly constant over spectral type for narrow or intermediate band filters. To calculate this ratio the Planck energy function was used to model the stellar flux. Although this is not precisely valid, forming the ratio should lead to quite accurate results. The temperatures used were those given by Johnson [7] and no distinction was made by luminosity class. The values for absorption were taken from Fig. 3.2 in the SKYMAP description by assuming absorption in magnitude is a linear function of wavelength over the range of interest (4800 Å - 10000 Å),

$$a(\lambda) = 1.77 \times 10^{-4} \times \lambda(\text{Å}) + 1.77,$$

Where $a(\lambda)$ is absorption in magnitude at wavelength $\lambda$. The results of these calculations were that $a_I/a_V$ varied from 0.25 to 0.32. The same value of $a_I/a_V$ was used for I, III and V luminosity class stars at a given spectral type.

A2.2.3 SKYMAP DATA

The data from SKYMAP needed for the magnitude conversion consists of apparent visual magnitude, spectral type, luminosity class and
absorption in V. These data, with the exception of $a_V$, are given for most stars. It was necessary to collapse categories of luminosity class and some spectral types since the table from Johnson is limited. Collapsing is justified in most cases because either the category contains few stars and/or the properties are similar to those of listed star types. The following combinations were used:

- **R**
  - Luminosity class III, spectral type M
  - (all are variants of M III)
- **N**
- **C**
- **S**
- **WR**
- **WC**
  - Lum class III, spectral type 09
  - (all are hot giant stars similar to 09)
- **WN**
- **IV**
- **III**
- **II**
- **V**
- **VI**
- **I**
- **I_a',ab',b**

No Luminosity class $\rightarrow$ V (most stars are V stars)
No subinterval in spectral type $\rightarrow$ 5 (i.e., A becomes A5)
No absorption given $\rightarrow$ set to 0
No spectral type $\rightarrow$ exclude

Given the spectral type and luminosity class, the value of $V_{abs} - I_{abs}$ and $a_I/a_V$ were found by interpolation in the table. Then:
\[ m_I = m_V - (V_{\text{abs}} - I_{\text{abs}}) - a_V(1 - a_I/a_V), \]  
(A2.5)

where \( m_V \) and \( a_V \) are from SKYMAP.

SKYMAP contains approximately 45,000 stars. Of these, 37 were not processed due to missing spectral type or visual magnitude.

A2.2.4 Magnitude Limit of CCD Sensor

In order to establish a reasonable magnitude limit for a CCD sensor we 1) determine the flux in the filter bandpass for some standard star at the earth's atmosphere, and 2) multiply by appropriate factors dictated by the sensor.

The most direct way to obtain a flux estimate would be to observe known stars from space with the CCD sensor. Barring this, ground-based observations of stars with varying zenith angles could yield flux estimates outside the earth's atmosphere.

Our approximate method was to numerically integrate the surface flux distribution of a K7 V star model atmosphere over the I bandpass. The absolute I magnitude of such a star is approximately 6.2 (\( V = 8.1, V - I = 1.92 \)). The radius is given by \( \log R^*/R_{\text{sun}} = -0.11 \) where \( R_{\text{sun}} = 6.96 \times 10^{10} \) cm. The surface flux is scaled by \( (R^*/R_{\text{sun}})^2 \times (L_{\text{sun}}/L_{\text{pc}})^2 \) = \( 3.08 \times 10^{-18} \) where \( L_{\text{pc}} = 3.08 \times 10^{17} \) cm. The result of integrating and scaling is:

\(~1024 \text{ photons/cm}^2\text{sec.}~\)

Typical scale factors are:

- Lens area: 26.7 cm²
- CCD response peak efficiency = .60
- I filter transmission peak efficiency = 0.85
-CCD effective area utilization = 0.46
-Integration time = 0.1 sec.

If we desire a minimum of 7500 photons/star for a sufficient signal-to-noise ratio, we compute the I magnitude limit of:

\[ m_{\text{limit}} = 6.2 + 2.5 \log \frac{641}{7500} = 3.5 \]

Note that many factors are uncertain or could be altered. Integration time could be increased to 1 sec to give a limit of 6.0. We have chosen 5.0 to be the cutoff magnitude since this seems obtainable and gives approximately 5400 stars, a sufficient number for the pattern recognition process to work reliably.

We also note that model atmospheres for a variety of spectral types could be used to repeat the above calculation to yield a more precise magnitude limit.

The magnitude limit is flexible since the integration time for the star sensor is variable over a wide range. If the integration time is changed by a factor of 10 the magnitude limit is changed by 2.5 mag. In addition, the dynamic range to typical CCD arrays is 200 or about 6 magnitudes. The response is linear over the range to allow accurate magnitude calibration and detection.
Appendix 3: Star Data Base And Mission Catalog Creation

The star catalog data base system SKYMAP (Ref. 5) has been selected as the master star data base. The SKYMAP catalog was developed from the SAO catalog and other sources specifically for attitude determination programs by NASA-GSFC. It is complete to the eighth magnitude in either the blue (B) or visual (V) magnitudes. Additionally, the catalog contains right ascensions, declinations, and, when known, the spectral type, luminosity class, and amount of interstellar absorption in the V wavelength range. Recent work [at the Naval Surface Weapons Center] has uncovered a significant number of corrections to the SKYMAP data base which will be reflected in future revisions of the present SKYMAP data base.

The on-board (or mission) star catalog is divided into celestial sphere cells so as to permit efficient microcomputer access during the pattern recognition process. In order to keep storage requirements for the mission catalog to a minimum, the cells do not overlap. The placement of the cell centers is given by the polar angle $\theta$ and longitude $\lambda$ according to

$$\theta_n = \cos^{-1}(\xi_n) \quad n = 0,1,2...N$$  \hspace{1cm} (A3.1)

and

$$\lambda_{nj} = \frac{2\pi j}{2n+1} \quad j = 0,1,2...2n$$  \hspace{1cm} (A3.2)

$$\xi_n = (-1)^n \cos\left(\frac{n\pi}{2N+1}\right), \quad n = 0,1,2...N$$  \hspace{1cm} (A3.3)

These formulae yield $(N+1)^2$ points: $N+1$ polar angles or declination zones with spacing $2\pi/(2N+1)$, and $(2n+1)$ equally spaced regions in each zone.
The choice of \( N \) is somewhat arbitrary. A large \( N \) yields small cells which would require more than one cell to be accessed; a small \( N \) yields large cells which would increase the number of trials in the pattern recognition process as well as causing a possible storage problem. Taking into account the \( 7^\circ \times 9^\circ \) field-of-view, a value of \( N = 22 \) was chosen, yielding 529 cells.

To facilitate computer access, the cells are ordered within memory according to a parameter \( n^2 + j \); a table lists the starting relative address of each cell and the number of stars in each. Thus, given a boresight estimate \((\theta, \lambda)\), the primary cell location is given by

\[
\begin{align*}
    n &= 2[\theta/\Delta\theta + 0.5] & (\theta < 90^\circ) \\
    &= 2N + 1 - 2[\theta/\Delta\theta + 0.5] & (\theta > 90^\circ) \\
    j &= [\lambda/\Delta\lambda + 0.5],
\end{align*}
\]

where \([x]\) indicates integer arithmetic (truncation to next smallest integer). The table of cells is then consulted for identification of the appropriate memory location. In all, the catalog access routine reads data from the 4 nearest neighboring cells around the estimated boresight (Figure A3.1) and thus provides nearly complete coverage of the estimated FOV by the 4 cells.

The CCD is assumed (see Appendix 2) to respond to stars of \( I \) magnitude 5 or lower - approximately 5400 stars. If these 5400 stars are assumed to be distributed uniformly over the celestial sphere, the star density \( \rho \) would be

\[
\rho = \frac{5400 \text{ stars/sphere}}{41,253 \text{ square degrees/sphere}} = 0.13 \text{ stars/square degree}
\]

(A3.6)
Figure A3.1  Catalog cell pattern obtained for a typical estimated field of view.
For a field-of-view of $7^\circ \times 9^\circ \approx 63$ square degrees, we would expect

$$(63 \text{ square degrees})(0.13 \text{ stars/square degree}) \approx 8.2 \text{ stars}^*$$

in a field-of-view (assuming uniform density). To obtain a measure of the range of the number of stars actually detectable per field-of-view, the boresight was randomly oriented over the entire celestial sphere 100 times. For each trial, the mission catalog was consulted and the number of stars in the field-of-view recorded. The average number of stars per field-of-view was six; in no case were fewer than two stars in the field-of-view.

*Due to non-uniform star population of the celestial sphere, this number decreases to about 5 at the north galactic pole.
Appendix 4: Inter-Star Cosine Calculations

The key to efficient star identification is to take advantage of the sub-ten-arcsecond precession of Process A; the angles between pairs of measured stars are very well determined by the measured coordinates and can be used to identify the corresponding catalog stars. The cosine of the angle between a typical pair of measured stars can be computed from the measured image coordinates as

$$c_{ij} = \cos \theta_{ij} = \frac{x_i x_j + y_i y_j + f^2}{\sqrt{(x_i^2 + y_i^2 + f^2)(x_j^2 + y_j^2 + f^2)}}$$

(A4.1)

The cosine of the angle between a typical pair of catalog stars can be computed from the catalog direction cosines as

$$c_{ij} = \cos \theta_{ij} = L_{i1} L_{j1} + L_{i2} L_{j2} + L_{i3} L_{j3}.$$  

(A4.2)

The pattern recognition logic we developed makes use of the smallness of the difference between (A4.1) and (A4.2) as a means to tentatively identify measured stars in the catalog. Our strategy assumes a steady-state condition in which the estimated boresight is within a degree or so of the true boresight direction. Thus, the highest probability of finding a pair match lies in comparing stars from the center of the sub-catalog distribution. For this reason, we sort the sub-catalog stars by angular distance from the boresight. We proceed to pair catalog stars by using the sum of the star indices (after sorting) as our criterion for the pairing order. Each catalog pair is compared with the pairs of measured stars (cosines are stored in a table). We eliminate from consideration star pairs with separation less than one degree because of the possible large "roll" error about
the boresight. In addition, we do not use catalog pairs with separations greater than about 10 degrees (greater than the FOV size). If we find agreement between a catalog pair and measured pair we perform a magnitude test to resolve the $180^\circ$ ambiguity.

The above strategy is not necessarily optimal. However, we have found it to be very efficient and it allows for mismatch of several degrees, at least, between the estimated and true boresight vectors.
Appendix 5: Stellar Aberration

The effect of stellar aberration is to cause a star's apparent direction to shift towards the direction of the observer's motion. The amount of shift depends on both the velocity of the observer and on the angle between the observer's line of sight (the star direction) and the velocity vector. The shift is:

\[ a = \frac{v}{c} \sin \alpha \]  

where \( a \) = aberration in radians
\( v \) = observer's speed
\( c \) = velocity of light
\( \alpha \) = angle between velocity vector and the true star direction.

For our purpose, we must express a star's shifted direction in terms of the true direction, the vehicle velocity and the angle between the velocity vector and true direction; that is,

\[
\begin{bmatrix}
L'_{x} \\
L'_{y} \\
L'_{z}
\end{bmatrix}
= f(L_{x}, L_{y}, L_{z}, v, \alpha)
\]  

(A5.2)

If we let \( v_{s} \) be the velocity of starlight in the inertial frame and \( v \) be observer velocity, then the relative velocity of the starlight as seen by the observer is:

\[
\frac{v_{s}}{0} = \frac{v_{s} - v}{v}
\]  

(A5.3)

Now, if we let \( \hat{L}_{n} \) be the unit vector in the true direction and \( \hat{L}'_{n} \) the unit vector in the shifted direction, we can rewrite this as (refer to
Figure A5.1)

\[(c + v \cos \alpha) \hat{\mathbf{r}}_n = c \hat{\mathbf{r}}_{n1} + \mathbf{v} \]  

(A5.4)

To first order this becomes

\[\hat{\mathbf{r}}_n = (1 - \frac{v}{c} \cos \alpha) \hat{\mathbf{r}}_{n1} + \frac{\mathbf{v}}{c} \]  

(A5.5)

or

\[
\begin{bmatrix}
L'_{x} \\
L'_{y} \\
L'_{z}
\end{bmatrix} = (1 - \frac{v}{c} \cos \alpha) \begin{bmatrix}
L_{x} \\
L_{y} \\
L_{z}
\end{bmatrix} + \frac{1}{c} \begin{bmatrix}
v_{x} \\
v_{y} \\
v_{z}
\end{bmatrix}
\]  

(A5.6)

This equation is used for calculating the displacement of a star's unit vector. The velocity vector is computed for the combined velocity of the earth and satellite and Herrick's "f and g" solution is used to calculate the individual velocities each time Process B accepts new data from Process A.

We note several points concerning the effects of aberration on the star tracker. The speed of the earth in its orbit is 30 km/s and the maximum speed of an earth orbiting satellite is < 8 km/s relative to the earth. Therefore, the maximum shift in a star's direction is about 26 arcseconds. This maximum occurs for stars 90° from the velocity vector. However, all stars in this neighborhood will be shifted by nearly this amount and, thus, the distortion of the FOV will be insignificant. However, aberration will displace the boresight direction.

To avoid orientation errors in the combined FOV(A) and FOV(B) solution we must correct the catalogue direction cosines by applying Eq. (A5.6).
Figure A5.1  Star direction displacement of stellar aberration due to observer's velocity.
For those stars in the direction of the velocity vector, the shift in direction will be small. But since the shift is always towards the velocity vector the distortion is noticeable (an apparent shrinking of the FOV). In this case, the aberration should be applied before the final least-squares solution for the single FOV.

We have chosen to correct for aberration when a sub-catalog is selected from the mission catalog. This decision was based on programming ease although it does require more time to correct the whole sample rather than only the matched stars. The impact is not severe, however, since the calculation is very simple.
Appendix 6: Least-Squares Correction Techniques

In Process B we seek to minimize the sum of the squares of the residuals between measured star image coordinates and predicted coordinates for the same stars, using direction cosines from the on-board catalog. The mapping of catalog positions onto the CCD image plane is a function of Euler parameters via the stellar colinearity equations:

\[
x = f\left(\frac{A_{N11}L_1 + A_{N12}L_2 + A_{N13}L_3}{A_{N31}L_1 + A_{N32}L_2 + A_{N33}L_3}\right)
\]

\[
y = f\left(\frac{A_{N21}L_1 + A_{N22}L_2 + A_{N23}L_3}{A_{N31}L_1 + A_{N32}L_2 + A_{N33}L_3}\right)
\]

where

\[f = \text{lens focal length}\]
\[A_{Nij} = \text{elements of the coordinate frame rotation matrix } [AN].\]
\[L_i = \text{star direction cosines for the particular star, measured in the } N \text{ frame.}\]

If we let:

\[X = \{(x_i, y_i)\} = \text{vector of calculated CCD image plane coordinates.}\]
\[\tilde{X} = \{(x_i, y_i)\}_m = \text{vector of measured star image positions on the CCD.}\]

and

\[\Delta X = \tilde{X} - X = \text{vector of residuals,}\]

then we seek to find the set of Euler parameters, \(\beta\), such that the weighted sum of the squares of the residuals is minimized; i.e., minimize

\[\phi = \Delta X_p^T W \Delta X_p\]  \hspace{1cm} (A6.2)

where

\[\Delta X_p = \tilde{X} - X_p\]
and

\[ X_p = \text{vector of linearly predicted image coordinates.} \]

But, by first-order Taylor expansion

\[ \Delta X_p = \Delta X_c - A \Delta \beta \]

where

\[ \Delta X_c = \text{vector of current image coordinate residuals based current estimates of } \beta. \]

\[ A = \text{matrix of partial derivatives of the colinearity equations with respect to Euler parameters.} \]

\[ A \Delta \beta = \text{corrections to the current estimates of Euler parameters.} \]

Thus, we can write:

\[ \phi_p = (\Delta X_c - A \Delta \beta)^T W (\Delta X_c - A \Delta \beta). \]  

(A6.4)

In addition to finding the set of Euler parameters to minimize \( \phi_p \), we must also satisfy the constraint equation:

\[ \beta^T \beta = 1. \]  

(A6.5)

Letting \( \beta_p = \beta_c + \Delta \beta \), we find:

\[ (\beta_c + \Delta \beta)^T (\beta_c + \Delta \beta) = 1 \]

or to first order:

\[ 1 - \beta_c^T \beta_c = 2 \beta_c^T \Delta \beta. \]  

(A6.6)

Thus, our problem requires that we minimize Eq. A6.4 subject to the constraint equation, Eq. A6.6. The constraint equation can be incorporated in Eq. A6.4 as an additional perfect observation equation but with a large weight. That is, \( \Delta Y = (1 - \beta^T \beta) \) is appended to the \( \Delta X_c \) vector and \( 2 \beta^T \) appended as an additional row into the A matrix. The relative weight
for this equation, the last element of W, is chosen large enough (about $10^3$) so that $(A^TWA)^{-1}$ does not change appreciably for variations in this weight. Then, for minimization, we require:

$$\nabla_{\Delta B} \phi = -2A^TWA_X + 2(A^TWA)\Delta B = 0$$

or

$$\Delta B = (A^TWA)^{-1} A^TWA_X.$$  \hspace{1cm} (A6.7)

**Determination of Interlock Euler Parameters Between FOV(A) and FOV(B)**

Process B, in analyzing star image data, first treats FOV(A) and FOV(B) independently. The least-squares differential correction determines the best estimate of the Euler parameters $(\beta_{VN})$ orienting the vehicle frame, V (see Appendix 7), relative to the inertial frame, N. For FOV(B) the interlock relationship between FOV(A) and FOV(B) $(\beta_{BA})$ is assumed known and $\beta_{VN}$ is adjusted again. In reality, however, the interlocks do vary slightly with time. Therefore, we have expanded our algorithm to treat the combined data from FOV(A) and FOV(B) in order to determine, simultaneously, the $\beta_{VN}$ and $\beta_{BA}$ which minimize (in a least-squares sense) the star image coordinate residuals (see Appendix 7 for further details).

In order to rigorously interpret $(A^TWA)^{-1}$ as the $8 \times 8$ covariance matrix of the estimated Euler parameters, W should be chosen as the inverse of the "measurement" covariance matrix. However, since a scale factor on W is formally immaterial in the least squares solution and assuming all measurement errors are uncorrelated we exercise the simple option of setting W to an identity matrix except for the larger constraint weights; the correct covariance matrix is obtained by simply multiplying the converged $(A^TWA)^{-1}$ matrix by the image coordinate measurement variance. The two constraints of the form (A6.6) are treated as "perfect
measurements". Thus, it is clear that the two corresponding formal weights are ∞; it is equally clear that we are limited to choosing a sufficiently large number (about $10^3$) in practice. We have carried out sufficient experimentation to expect no implementation problems here; the differential correction process converges well.

We have found it desirable to further process the interlock Euler parameters, $\beta_{BA}$. We assume (justifiably!) that these interlocks vary slowly with time, but the interlocks calculated by the combined least-squares method, discussed above, show relatively large scatter about their true values. This sensitivity is due to the relatively poor determination of the roll about each boresight (this does not affect the determination of $\beta_{VN}$). To better monitor the interlock parameters we adopted a discrete Kalman filter algorithm to combine the predicted $\beta_{BA}$, determined from previous data frames, with the $\beta_{BA}$ computed by the least-squares correction. The equations needed for this are:

$$\hat{\beta}_{BA}(k) = \overline{\beta}_{BA}(k) + K(k)(\tilde{\beta}_{BA}(k) - \overline{\beta}_{BA}(k))$$

$$K(k) = P_{k-1}(k)(L_{VK}V_k + P_{k-1}(k))^{-1}$$

$$P_{k-1}(k) = P_{k-1}(k-1) + Q(k)$$

$$P_k(k) = (I - K(k))P_{k-1}(k)$$

$$Q(k) = BQ'B^T$$

$$\beta_{BA} = \frac{\partial \beta_{BA}}{\partial \Phi}$$

where $\hat{\beta}_{BA}(k) = \text{optimal estimate of interlock parameters at time } t_k$, obtained by combining predicted and computed values,
\( B_{BA}(k) \) = predicted interlock parameters at time \( t_k \) based on previous analysis,

\( \hat{B}_{BA}(k) \) = interlock parameters at time \( t_k \) obtained via least-squares correction,

\( K(k) \) = Kalman gain matrix at time \( t_k \),

\( P_{i(j)} \) = covariance matrix associated with \( S_{BA} \) at time \( t_i \) based on analysis of interlock values through time \( t_i \),

\( L_{V_k V_k} \) = covariance matrix associated with calculated interlock parameters, \( \tilde{B}_{BA}(k) \),

\( Q(k) \) = process noise matrix,

\( Q' \) = process noise matrix for 3 interlock angles (\( \phi = (\phi_1, \phi_2, \phi_3) \))

Formally, \( P_{k-1}(k) \) would be obtained by forward integration of the matrix Riccati equation, or by other methods. However, since to first order, \( \dot{B}_{BA} = 0 \) and assuming the process noise is constant in time as well, \( P_{k-1}(k) \) is obtained by simply adding the time integral of the process noise, \( Q(k) \), to the previous covariance, \( P_{k-1}(k-1) \). In addition, for simplicity we assume \( Q' \) is diagonal with equal noise for each angle. For programming ease we have used a pre-calculated B matrix, valid for our nominal interlock arrangement.

The process noise matrix \( Q' \) essentially controls the scatter of the estimated Euler parameters. Small values for the elements of \( Q' \) will permit little change in the \( \tilde{B}_{BA} \) (making it insensitive to each new \( \dot{B}_{BA} \)). However, this may cause the \( \hat{B}_{BA} \) not to "track" the true variations. Conversely, large values for \( Q' \) elements will cause more scatter in \( \dot{B}_{BA} \). Obviously, \( Q' \) is a "tuning" parameter which must be selected for the particular system.
Appendix 7: Orientation of the Vehicle Frame

We describe the vehicle orientation by the set of Euler parameters, $\beta_{VN}$, which orient a "V" frame (defined below) relative to the inertial frame, "N". The V frame is defined entirely by the boresight vectors of the two star sensors [FOV(A), FOV(B)]. Given the boresight of FOV(A) as $a_3$ and of FOV(B) by $b_3$, we define the V frame unit vectors as follows:

$$
\begin{align*}
    v_1 &= (a_3 + b_3) / |a_3 + b_3| \\
    v_3 &= (a_3 \times b_3) / |a_3 \times b_3| \\
    v_2 &= v_3 \times v_1
\end{align*}
$$

(A7.1)

The advantage of this frame is that since the boresights of the two FOV are well determined so also will be the V frame and, hence, the $\beta_{VN}$ parameter set. The poorly determined roll angle about each boresight will not affect $\beta_{VN}$.

In addition to $\beta_{VN}$, we also make use of $\beta_{BA}$, the Euler parameters orienting FOV(B) with respect to FOV(A). These parameters are monitored as a means for monitoring the interlock angles between FOV(A) and FOV(B).

As we will show below, we do not actually need the $v$ unit vectors calculated by the above equations. We do need, however, the rotation matrix $AV$ which rotates the V frame into the FOV(A) frame. The matrix $AV$ can be calculated from the BA rotation matrix.

Matrix $AV$ can be constructed by filling its columns with the $v$ unit vectors expressed in the A frame. We first express $b_3$ in terms of $a$ unit vectors in order to calculate the vectors $v_1, v_2, v_3$ (in the A frame).

$$
    b_3^A = BA a_3^A
$$

(A7.2)

And since
we see that

\[
A_3 = \begin{bmatrix}
0 \\
0 \\
1 
\end{bmatrix}
\]

(A7.3)

Therefore, in the A frame:

\[
v_1 = (a_3 + b_3)/(|a_3 + b_3|)
\]

\[
= \begin{bmatrix}
\frac{BA_{31}}{(2 + 2BA_{33})^{1/2}}, & \frac{BA_{32}}{(2 + 2BA_{33})^{1/2}}, & \frac{1 + BA_{33}}{(2 + 2BA_{33})^{1/2}}
\end{bmatrix}^T
\]

(A7.5)

\[
v_3 = (a_3 \times b_3)/(|a_3 \times b_3|)
\]

\[
= \begin{bmatrix}
\frac{-BA_{32}}{(1 - BA_{33})^{1/2}}, & \frac{BA_{31}}{(1 - BA_{33})^{1/2}}, & 0
\end{bmatrix}^T
\]

(A7.6)

\[
v_2 = v_3 \times v_1
\]

\[
= \begin{bmatrix}
\frac{BA_{31}}{(2 - 2BA_{33})^{1/2}}, & \frac{BA_{32}}{(2 - 2BA_{33})^{1/2}}, & \frac{-(2 - 2BA_{33})^{1/2}}{2}
\end{bmatrix}^T
\]

(A7.7)

The vectors, \(v_i\), expressed in the A frame, form the columns of matrix \(AV\). We see that \(AV\) is a function of just 3 variables \(BA_{3i}, i = 1, 2, 3\). For the least-squares differential correction of Process B we need the partial derivatives \(\partial AV/\partial B_j, j = 0, 1, 2, 3\). To simplify the calculations we expand this:
\[
\frac{\partial \mathbf{V}}{\partial \beta_j} = \sum_{i=1}^{3} \frac{\partial \mathbf{V}}{\partial \beta_{3i}} \frac{\partial \beta_{3i}}{\partial \beta_j}
\]  
(A7.8)

where \( \frac{\partial \mathbf{V}}{\partial \beta_{3i}} \) is a \( 3 \times 3 \) matrix and \( \frac{\partial \beta_{3i}}{\partial \beta_j} \) are elements of a \( 3 \times 3 \) matrix \( \frac{\partial \mathbf{B}}{\partial \beta_j} \). The partials \( \frac{\partial \mathbf{V}}{\partial \beta_{3i}} \), after some algebra, are

\[
\frac{\partial \mathbf{V}}{\partial \beta_{31}} = \begin{bmatrix}
1/(2 + 2\beta_{33})^{1/2} & 1/(2 - 2\beta_{33})^{1/2} & 0 \\
0 & 0 & 1/(1 - \beta_{33})^{1/2} \\
0 & 0 & 0
\end{bmatrix}
\]

\[
\frac{\partial \mathbf{V}}{\partial \beta_{32}} = \begin{bmatrix}
1/(2 + 2\beta_{33})^{1/2} & 1/(2 - 2\beta_{33})^{1/2} & 0 \\
0 & 0 & -1/(1 - \beta_{33})^{1/2} \\
0 & 0 & 0
\end{bmatrix}
\]

\[
\frac{\partial \mathbf{V}}{\partial \beta_{33}} = \begin{bmatrix}
-\beta_{33}/(2 + 2\beta_{33})^{3/2} & \beta_{33}/(2 - 2\beta_{33})^{3/2} & -\beta_{32}\beta_{33}/(1 - \beta_{33})^{1/2} \\
-\beta_{32}/(2 + 2\beta_{33})^{3/2} & \beta_{32}/(2 - 2\beta_{33})^{3/2} & \beta_{32}\beta_{33}/(1 - \beta_{33})^{1/2} \\
1/(2(2 + 2\beta_{33})^{1/2}) & 1/(2(2 - 2\beta_{33})^{1/2}) & 0
\end{bmatrix}
\]

We now consider in some detail how the derivative matrix for the least-squares differential correction is filled to recover \( \beta_{VN} \) and \( \beta_{BA} \). The
elements of the A matrix for treating a single FOV are found by expanding \( \frac{\partial X}{\partial \beta_{VN}} \):

\[
\frac{\partial X}{\partial \beta_{VN}} A = \frac{\partial X}{\partial \alpha N} \frac{\partial \alpha N}{\partial \beta_{VN}} \quad \text{[for FOV(A)]}
\]

\[
= \frac{\partial X}{\partial \alpha N} \left( \frac{\partial \alpha V}{\partial \beta_{VN}} VN + \frac{\partial VN}{\partial \beta_{VN}} \right) \quad \text{(using } \alpha N = \alpha V \cdot VN)\]

\[
= \frac{\partial X}{\partial \alpha N} \left( \frac{\partial VN}{\partial \beta_{VN}} \right) \quad \text{(A7.10)}
\]

\[
\frac{\partial X}{\partial \beta_{VN}} B = \frac{\partial X}{\partial \beta N} \frac{\partial \beta N}{\partial \beta_{VN}} \quad \text{[for FOV(B)]}
\]

\[
= \frac{\partial X}{\partial \beta N} \left( \frac{\partial \beta V}{\partial \beta_{VN}} VN + \frac{\partial VN}{\partial \beta_{VN}} \right) \quad \text{(using } \beta N = \beta V \cdot VN \text{ and } \beta V = \beta A \cdot \alpha V)\]

\[
= \frac{\partial X}{\partial \beta N} \left( \frac{\partial VN}{\partial \beta_{VN}} \right). \quad \text{(A7.11)}
\]

After matching at least 3 stars in each FOV, we can combine both FOV to recover the interlock parameter \( \beta_{BA} \) as well as \( \beta_{VN} \). In this case, we need the following matrix derivatives:

\[
\frac{\partial X}{\partial \beta_{VN}} A = \frac{\partial X}{\partial \alpha N} \left( \frac{\partial VN}{\partial \beta_{VN}} \right)
\]

\[
\frac{\partial X}{\partial \beta_{BA}} A = \frac{\partial X}{\partial \alpha N} \left( \frac{\partial AV}{\partial \beta_{BA}} \frac{\partial VN}{\partial \beta_{BA}} + \frac{\partial VN}{\partial \beta_{BA}} \right) = \frac{\partial X}{\partial \alpha N} \left( \frac{\partial AV}{\partial \beta_{BA}} \frac{\partial BN}{\partial \beta_{BA}} \right) \quad \text{(A7.12)}
\]

\[
\frac{\partial X}{\partial \beta_{VN}} B = \frac{\partial X}{\partial \beta N} \left( \frac{\partial VN}{\partial \beta_{VN}} \right)
\]

\[
\frac{\partial X}{\partial \beta_{BA}} B = \frac{\partial X}{\partial \beta N} \left( \frac{\partial BN}{\partial \beta_{BA}} \frac{\partial BN}{\partial \beta_{BA}} + \frac{\partial BN}{\partial \beta_{BA}} \frac{\partial BN}{\partial \beta_{BA}} \right)
\]
\[ \frac{\partial X}{\partial \beta_{BA}}_B = \frac{\partial X}{\partial \beta_{BN}} + \frac{\partial X}{\partial \beta_{BA}}_A + \frac{\partial X}{\partial \beta_{VA}} + \frac{\partial X}{\partial \beta_{BA}}_A \tag{A7.13} \]

The elements of the vectors \( \Delta X \) and \( \Delta \beta \) and matrix \( A \) are:

\[
\Delta X = \begin{bmatrix}
(\Delta x, \Delta y)_A & = \text{vector of image residuals for FOV(A)} \\
(\Delta x, \Delta y)_B & = \text{vector of image residuals for FOV(B)} \\
1 - \beta_{VN}^T \beta_{VN} & = \text{constraint condition for } \beta_{VN} \\
1 - \beta_{BA}^T \beta_{BA} & = \text{constraint condition for } \beta_{BA}
\end{bmatrix}
\]

\[
\Delta \beta = \begin{bmatrix}
\Delta \beta_{VN} = \text{correction vector for } \beta_{VN} \\
\Delta \beta_{BA} = \text{correction vector for } \beta_{BA}
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
\frac{\partial (x, y)}{\partial \beta_{VN}} & A & \frac{\partial (x, y)}{\partial \beta_{BA}} \\
\frac{\partial (x, y)}{\partial \beta_{VN}} & B & \frac{\partial (x, y)}{\partial \beta_{BA}} \\
\beta_{VN}^T W_{1/2} & 0 & \beta_{BA}^T \beta_{1/2}
\end{bmatrix}
\]

Then \( \Delta \beta = (A^T W A)^{-1} A^T \Delta X \); the corrections are added to \( \beta_{VN} \) and \( \beta_{BA} \) and the process is repeated until \( \Delta \beta \) is small. The final values for \( \beta_{VN} \) are passed to Process C.
Appendix 8: Riccati Equation Covariance Propagation

The Kalman filter formulated for Process C includes a direct numerical integration of the covariance matrix between two time points. This allows a more rigorous incorporation of the process noise component in the covariance matrix propagation. Our initial method was to compute the state transition matrix to use this to propagate covariance by pre- and post-multiplication of the previous covariance matrix. An estimate of the process noise estimate was then added on. We outline the covariance integration technique below.

We have chosen as our state vector:

\[
X = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\
\end{bmatrix} = \begin{bmatrix}
4 \text{ Euler parameters, } \beta_{VN}, \text{ orienting the vehicle frame with respect to inertial frame, } N, \text{ and } 3 \text{ gyro bias values, } b.
\end{bmatrix}
\]

The state differential equations for our system are:

\[
\begin{align*}
\{\dot{x}_1\} &= \{\dot{\beta}_{VN}\} = [\beta_{VN}][V](\omega_{GN}) \\
&= [\beta_{VN}](\omega_{VN}) \quad \text{(A8.1a)} \\
&= [\omega_{VN}](\beta_{VN}) \quad \text{(A8.1b)} \\
\{\dot{x}_2\} &= \{\dot{b}\} = 0 \quad \text{(A8.2)}
\end{align*}
\]

where
\[
\begin{bmatrix}
-\beta_1 & -\beta_2 & -\beta_3 \\
\beta_0 & -\beta_3 & \beta_2 \\
\beta_3 & \beta_0 & -\beta_1 \\
-\beta_2 & \beta_1 & \beta_0
\end{bmatrix}
\]
\[\mathbf{[\beta]} = \frac{1}{2}\]

\[
\begin{bmatrix}
0 & -\omega_1 & -\omega_2 & -\omega_3 \\
\omega_1 & 0 & \omega_3 & -\omega_2 \\
\omega_2 & -\omega_3 & 0 & \omega_1 \\
\omega_3 & \omega_2 & -\omega_1 & 0
\end{bmatrix}
\]
\[\mathbf{[\omega]} = \frac{1}{2}\]

\{\omega_{\text{GN}}\} = \text{vector of true rotation rates of the vehicle about three orthogonal body fixed axes,}

\[\mathbf{[VG]} = 3 \times 3 \text{ rotation matrix which rotates the gyro rates from the gyro frame to the vehicle frame.}\]

Our model for each gyro measurement, \(\dot{\omega}_{\text{GN}}\), includes an unknown noise term, \(v\), and a gyro bias, \(b_{\text{GN}}\), which we can estimate. Then, the state differential equation for \(\{X_1\}\) becomes:

\[
\dot{\{X_1\}} = [\mathbf{B}_{\text{VN}}][\mathbf{V}G]\{(\omega_{\text{GN}}) - \{b_{\text{GN}}\} + \{v\})
\]

\[
= [\dot{\omega}_{\text{VN}}]\{\mathbf{B}_{\text{VN}}\} - [\mathbf{B}_{\text{VN}}][\mathbf{V}G]\{b_{\text{GN}}\} + [\mathbf{B}_{\text{VN}}][\mathbf{V}G]\{v\}.
\]

(A8.3)

Combining the two state differential equations, they can be rewritten in a linear form:

\[\dot{X} = F_X + Gv,\]

where

\[
F = \begin{bmatrix}
F_{11} & F_{12} \\
- \frac{1}{2} & - \frac{1}{2} \\
F_{21} & F_{22}
\end{bmatrix}
\]
\[
F = \begin{bmatrix}
\mathbf{\bar{w}}_{\text{VN}} & \mathbf{B}_{\text{VN}}[\mathbf{V}G]^T \\
0 & 0
\end{bmatrix}
\]

\[
G = \begin{bmatrix}
\mathbf{G}_{11} & \mathbf{G}_{12} \\
\mathbf{G}_{21} & \mathbf{G}_{22}
\end{bmatrix} = \begin{bmatrix}
\mathbf{B}_{\text{VN}}[\mathbf{V}G] & 0 \\
0 & \mathbf{S}[\mathbf{I}]
\end{bmatrix}
\]

\(S\) is a scale factor or "tuning" parameter.

The covariance matrix propagation is calculated by numerical integration of the matrix Riccati equation:

\[
\dot{P} = FP + PF^T + GQG^T
\]

where \(P\) is the covariance matrix and \(Q\) is the process noise covariance matrix which, in our case, represents a measure of the noise covariance between gyro rates about the three axes and between the three gyro biases. This matrix is taken to be a 6 x 6 diagonal matrix. In order to speed computation we partition this equation:

\[
\begin{bmatrix}
\dot{\mathbf{p}}_{11} & \dot{\mathbf{p}}_{12} \\
\dot{\mathbf{p}}_{21} & \dot{\mathbf{p}}_{22}
\end{bmatrix} = \begin{bmatrix}
F_{11} & F_{12} \\
F_{21} & F_{22}
\end{bmatrix} \begin{bmatrix}
\mathbf{p}_{11} & \mathbf{p}_{12} \\
\mathbf{p}_{21} & \mathbf{p}_{22}
\end{bmatrix} + \begin{bmatrix}
\mathbf{p}_{11} & \mathbf{p}_{12} \\
\mathbf{p}_{21} & \mathbf{p}_{22}
\end{bmatrix} \begin{bmatrix}
F_{11}^T & F_{21}^T \\
F_{12}^T & F_{22}^T
\end{bmatrix}
\]

\[
+ \begin{bmatrix}
\mathbf{G}_{11} & \mathbf{G}_{12} \\
\mathbf{G}_{21} & \mathbf{G}_{22}
\end{bmatrix} \begin{bmatrix}
\mathbf{Q}_{11} & \mathbf{Q}_{12} \\
\mathbf{Q}_{21} & \mathbf{Q}_{22}
\end{bmatrix} \begin{bmatrix}
\mathbf{G}_{11}^T & \mathbf{G}_{21}^T \\
\mathbf{G}_{12}^T & \mathbf{G}_{22}^T
\end{bmatrix}
\]

Since \(F_{21} = 0\), \(F_{22} = 0\), \(G_{12} = 0\), \(G_{21} = 0\), \(Q_{12} = 0\), and \(Q_{21} = 0\) we can write the set of four equations implied in the above equation as
\[ \hat{P}_{11} = F_{11}P_{11} + F_{12}P_{21} + P_{11}F_{11}^T + P_{12}F_{12} + G_{11}Q_{11}G_{11}^T \]
\[ \hat{P}_{12} = F_{11}P_{12} + F_{12}P_{22} \]
\[ \hat{P}_{21} = P_{21}F_{11} + P_{22}F_{12}^T \]
\[ \hat{P}_{22} = G_{22}Q_{22}G_{22}^T. \]

Also, since \( P \) is symmetric, \( P_{12} = P_{21}^T, P_{11} = P_{11}^T, P_{22} = P_{22}^T \) and since \( F_{11} = [\omega], \) which is skew symmetric, \( F_{11}^T = -F_{11}, \) and \( G_1 = -F_{12}. \) Therefore,
\[ \hat{P}_{11} = (F_{11}P_{11} + F_{12}P_{21}) + (F_{11}P_{11} + F_{12}P_{21})^T + F_{12}QF_{12} \]
\[ \hat{P}_{21} = P_{22}F_{12}^T - P_{21}F_{11} = \hat{P}_{12}^T. \]

We must integrate three matrix differential equations, \( \hat{P}_{11}, \hat{P}_{21} \) and \( \hat{P}_{22} \) to propagate the covariance matrix. Only two matrices need to be filled at each time step in the integration: \( (\hat{P}_{22} \) is a constant)
\[ F_{11} = [\omega_{VN}] \]
and
\[ F_{12} = -[\beta][\Gamma G]. \]

The simplest form for the state differential equation for the \( \beta_{VN} \) is then
\[ \dot{\beta} = [F_{12}](b_{GN} - \phi_{GN}) \]

Currently, we are using a two cycle Runge-Kutta numerical integration for the Riccati equation.

**Kalman Filter State Update with A-Priori Information**

As mentioned in the Phase II report, the Kalman filter, as formulated in that report, did not estimate the gyro bias values with the desired precision. This shortcoming was due, in part, to the gyro rate noise
having a magnitude similar to the biases themselves. We have remedied this problem by reformulating the Kalman filter to treat the biases as observable with an associated covariance matrix. Such a method seems justified since, in general, the biases vary slowly on the time scale of minutes and, therefore, we would have some knowledge of them obtained from previous iterations.

The general form for the discrete Kalman filter state update equation for a linear system is

$$\hat{X}_{k+1}(k+1) = \hat{X}_k(k+1) + K(k+1)[Z(k+1) - H(k+1)\hat{X}_k(k+1)]$$  \hspace{1cm} (A8.5)

where $Z$ is the measurement vector at time $t_{k+1}$ and $\hat{X}_k(k+1)$ is the state, integrated from time $t_k$ to $t_{k+1}$. In our formulations, $Z$ contains the output of Process B, the set of Euler parameters $p_{VN}$ relating the V frame to the inertial frame, and the three bias values from the previous pass through the Kalman filter. Thus, the observation vector $Z$ contains the same variables as the state vector making matrix $H = [I]$.

Looking now at the other equations for the Kalman filter:

$$K(k+1) = P_k(k+1)H^T(k+1)[L(k+1) + H(k+1)P_k(k+1)H^T(k+1)]^{-1}$$

$$= P_k(k+1)[L(k+1) + P_k(k+1)]^{-1}$$  \hspace{1cm} (A8.6)

where $L(k+1) =$ covariance matrix for the measurements and $P_k(k+1) =$ integrated covariance from the Riccati equation. The upper left $4 \times 4$ portion of $L$ is the covariance matrix from the least-squares results of Process B. The lower right $3 \times 3$ portion of $L$ is the covariance matrix of the gyro biases. This we have assumed to be diagonal (gyro biases are assumed independent); off diagonal portions of $L$ are assumed to be zero.

The covariance matrix update equation is
\[ P_{k+1}(k + 1) = [I - K(k + 1)H(k + 1)]P_k(k + 1) \]
\[ = P_k(k + 1) - K(k + 1)P_k(k + 1) \]  \hspace{1cm} (A8.7)

**Bias Estimation Results**

Our simulations indicate that the formulation discussed above functions extremely well. We note that as the estimated variance is decreased we get an improvement in the bias estimate. However, a stricter value on the bias variance also decreases the response time of the system in tracking a bias change.

It must be kept in mind that the gyro biases, as we have defined them, include not only true gyro bias but also other effects such as gyro nonorthogonality, gravity or magnetic effects, and poorly known interlock angles between sensor and gyro frames. It is expected that on the short term the gyro bias terms will absorb these errors and permit the state integrations to yield good updates.
Appendix 9: Rate Gyro Readout Data Generation

The simplest or nominal rotation history for a satellite for our study is one rotation per revolution around the earth. For our geometry this would be rotation about the \( g_2 \) unit vector, normal to the orbit plane. To be more realistic and provide a challenge to the software, we have formulated a more complicated read-out record for the gyroscopes. In a satellite there may be various motors, panels and antennae. Each of these may vibrate at various frequencies. Therefore, we have generated gyro rates with a spectrum of frequencies, phases and amplitudes. To do this we have created data in the (Discrete) Fourier Transform space or frequency space. A Fast Fourier Transform of the data yields a gyro record. We used the formula

\[
f_j = e^{-10j G_j}
\]

(A9.1)

to generate a frequency spectrum, where \( G_j \) is a Gaussian distributed random number. We make the real transform symmetric about zero frequency and the imaginary data anti-symmetric in order yield a real gyro record. With proper scaling, we have used the above form to generate a gyro readout every 0.5 sec. for each gyro axis.

Proper application of the above technique yields a realistic gyro record, but it has the disadvantage of making it difficult to specify a true rotation and attitude history for the spacecraft. That is, Runge-Kutta integration is designed to integrate smooth functions whereas this rate simulation is discrete. Also, in a real system some of the frequencies recorded by the gyroscopes are just vibration or noise and not rotation. For consistency, we decided to integrate the gyro record using two-cycle
Runge-Kutta in both the truth model programs and Process C. This at least allows comparison between models with and without additional noise. We have performed several tests to determine the effects of gyro noise and a complicated signal. These tests confirm our intuition that if we take a sufficient number of samples, the rapid zero mean oscillations about the mean motion do not significantly affect the integrated solution.
Appendix 10: Software Documentation, Program Listings and Sample Output

This appendix is divided into several parts. We first describe and list the program which generates simulated data, Datgen, and the program which processes the data using Processes B and C, Combin. Then, since these two programs have several subroutines in common, we describe and list all the subroutines for these programs, in alphabetical order. Figures A10.1 and A10.2 display the hierarchy of subroutine calls for these two programs.

A sample of program output is presented also.
Table of Contents for Appendix 10.

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Programs</td>
<td></td>
</tr>
<tr>
<td>Datgen Description</td>
<td>93</td>
</tr>
<tr>
<td>Datgen Listing</td>
<td>99</td>
</tr>
<tr>
<td>Combin Description</td>
<td>109</td>
</tr>
<tr>
<td>Combin Listing</td>
<td>112</td>
</tr>
<tr>
<td>Subroutines</td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td>118</td>
</tr>
<tr>
<td>Bias</td>
<td>121</td>
</tr>
<tr>
<td>Cross</td>
<td>123</td>
</tr>
<tr>
<td>Deriv</td>
<td>125</td>
</tr>
<tr>
<td>Circosb</td>
<td>127</td>
</tr>
<tr>
<td>Dxdbeta</td>
<td>130</td>
</tr>
<tr>
<td>Fill-y</td>
<td>133</td>
</tr>
<tr>
<td>Subroutine</td>
<td>Page</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
</tr>
<tr>
<td>Gauss</td>
<td>135</td>
</tr>
<tr>
<td>Kalman</td>
<td>137</td>
</tr>
<tr>
<td>Least</td>
<td>140</td>
</tr>
<tr>
<td>Least-1</td>
<td>142</td>
</tr>
<tr>
<td>Least-2</td>
<td>145</td>
</tr>
<tr>
<td>Mata</td>
<td>149</td>
</tr>
<tr>
<td>Mat-av</td>
<td>152</td>
</tr>
<tr>
<td>Orbit</td>
<td>155</td>
</tr>
<tr>
<td>Pairit</td>
<td>158</td>
</tr>
<tr>
<td>Perturb</td>
<td>164</td>
</tr>
<tr>
<td>Phoegn</td>
<td>166</td>
</tr>
<tr>
<td>Post-mul</td>
<td>168</td>
</tr>
<tr>
<td>Pre-mul</td>
<td>170</td>
</tr>
<tr>
<td>Proc-b</td>
<td>172</td>
</tr>
</tbody>
</table>
Subroutine

Runge ............................................ 177
Secant ............................................ 181
Sort .............................................. 184
Sample Output from Datgen ...................... 186
Sample Output from Combin (Program start) .......... 189
Sample Output from Combin (frame 5) ............. 190
Datgen

This program generates simulated data which is used for tests of Processes B and C. The data for a sequence of frames are written to a disk or tape file for later analysis. For each simulation test we can select variations in any of a number of parameters. However, the program itself must be modified to change the amplitude, period or form of the variation. To run this program it is necessary to supply a file containing realistic gyroscope read-out rates (currently, enough for 15 minutes of satellite motion with a spacing of 0.5 seconds between readouts), and a mission catalog of star positions (ordered into cells by the method of Appendix 3).

To begin generating data, the gyro rates and the table of star catalog cell positions are read from the external files. Next, we select the variations we wish to include in this simulation. The earth and satellite orbit constants and initial positions and velocities are specified in the program but can be changed if desired. These orbit parameters, along with our assumed geometry, are used to determine the initial orientation of the vehicle via rotation matrix VN, orienting the vehicle frame relative to the inertial frame; from VN we recover the initial values of Euler parameters, βVN. All constants for this simulation are stored in the first record of the data file (see Table A10.1).

We are now ready to compute image coordinates for successive frames of data (separated by 30 seconds of satellite motion in the present software version). We note that all variations such as gyro biases and coordinate frame perturbations are slowly varying and, therefore,
the perturbed values are calculated only once per 30 second interval (subroutines Bias and Perturb). For each frame of data we first integrate forward (via subroutine Runge) the kinematic equation for $\beta_{VN}$ using the "true" gyro rates, perhaps including a perturbed rotation matrix $VG$, found from $\beta_{VG}$ and used to rotate the gyro rates from the gyro frame into the vehicle frame. Gaussian noise is added to the integrated values of $\beta_{VN}$ in order to provide an "estimate" of $\beta_{VN}$ to start Process B, if it is run separately from Process C (see description of Combin). Euler parameters $\beta_{BA}$ which can be time varying and rotation matrix $BA$, relating FOV(B) with respect to FOV(A), are determined at this time also. We then compute the position and velocity vectors of the earth and satellite (with subroutine Orbit) and find the total velocity (which is later used by subroutine Access to add stellar aberration to star direction cosines). The various Euler parameters and total velocity are saved in the first part of each data record.

The next step is to calculate image coordinates for stars in each FOV. We calculate rotation matrix $AN$ (or $BN$ for FOV(B)) using subroutines Dircosb and Mat-av, and use the last row as the boresight unit vector of the star tracker. This unit vector is used by Access to retrieve a subcatalog of stars. Each star is then projected onto the focal plane (via Phoegn); if it lies within the CCD border, we save its position and magnitude. These "true" positions are then perturbed with Gaussian noise (subroutine Gauss) to produce "measured" star coordinates (up to 10 stars). The steps outlined above are repeated for FOV(B). Both the true and measured image coordinates and magnitudes are saved on the data file.
The last step in each frame is to add noise and/or bias values to the rate gyro data (for that frame) and save the results on the data file (see Table A10.2). Sufficient space is left between the end of this information and the end of the record so results of the analysis of Processes B and C can be saved for later evaluation.
Figure A10.1: Hierarchy of subroutine calls (left to right) for simulation program Datgen.
Table A10.1: Format of First Record of Simulation File

<table>
<thead>
<tr>
<th>Variable</th>
<th>Size</th>
<th>Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satellite orbit major axis</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>G*(mass of earth)</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Initial satellite position</td>
<td>3</td>
<td>3-5</td>
</tr>
<tr>
<td>Initial satellite velocity</td>
<td>3</td>
<td>6-8</td>
</tr>
<tr>
<td>Earth orbit major axis</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>G*(mass of sun)</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Initial earth position</td>
<td>3</td>
<td>11-13</td>
</tr>
<tr>
<td>Initial earth velocity</td>
<td>3</td>
<td>14-16</td>
</tr>
<tr>
<td>Satellite orbit inclination</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>Image centroid error (1-sigma)</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>Image magnitude error (1-sigma)</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>Euler parameter error (1-sigma)</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>Gyro rate error (1-sigma)</td>
<td>1</td>
<td>21</td>
</tr>
<tr>
<td>Gyro read-out spacing</td>
<td>1</td>
<td>22</td>
</tr>
<tr>
<td>Starting time (seconds)</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>Frame spacing (seconds)</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>Nominal interlock Euler parameters $\beta_{BA}$</td>
<td>4</td>
<td>25-28</td>
</tr>
<tr>
<td>Variation amplitudes</td>
<td>4</td>
<td>29-32</td>
</tr>
<tr>
<td>Variation frequency factors</td>
<td>4</td>
<td>33-36</td>
</tr>
<tr>
<td>Nominal gyro biases</td>
<td>3</td>
<td>37-39</td>
</tr>
<tr>
<td>Variation amplitudes</td>
<td>3</td>
<td>40-42</td>
</tr>
<tr>
<td>Variation frequency factors</td>
<td>3</td>
<td>43-45</td>
</tr>
<tr>
<td>Nominal interlock Euler parameters $\beta_{VG}$</td>
<td>4</td>
<td>46-49</td>
</tr>
<tr>
<td>Variation amplitudes</td>
<td>4</td>
<td>50-53</td>
</tr>
<tr>
<td>Variation frequency factors</td>
<td>4</td>
<td>54-57</td>
</tr>
<tr>
<td>Weight for apriori Euler parameters $\beta_{BA}$</td>
<td>1</td>
<td>58</td>
</tr>
<tr>
<td>Standard deviation for gyro biases</td>
<td>1</td>
<td>59</td>
</tr>
</tbody>
</table>
Table A10.2: Format of Data Records of Simulation File

<table>
<thead>
<tr>
<th>Variable</th>
<th>Size</th>
<th>Bytes/Number</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \beta_{VN} ) (true)</td>
<td>4</td>
<td>8</td>
<td>1-4</td>
</tr>
<tr>
<td>( \beta_{VN} ) (estimated)</td>
<td>4</td>
<td>8</td>
<td>5-8</td>
</tr>
<tr>
<td>( \beta_{VG} ) (true)</td>
<td>4</td>
<td>8</td>
<td>9-12</td>
</tr>
<tr>
<td>( \beta_{VG} ) (estimated)</td>
<td>4</td>
<td>8</td>
<td>13-16</td>
</tr>
<tr>
<td>( \beta_{BA} ) (true)</td>
<td>4</td>
<td>8</td>
<td>17-20</td>
</tr>
<tr>
<td>( \beta_{BA} ) (estimated)</td>
<td>4</td>
<td>8</td>
<td>21-24</td>
</tr>
<tr>
<td>Velocity components</td>
<td></td>
<td></td>
<td>25-27</td>
</tr>
<tr>
<td>No. stars in FOV(A)</td>
<td>1</td>
<td>8</td>
<td>28</td>
</tr>
<tr>
<td>((x,y,m)) (true)</td>
<td>10x3</td>
<td>4</td>
<td>29-58</td>
</tr>
<tr>
<td>((x,y,m)) (measured)</td>
<td>10x3</td>
<td>4</td>
<td>59-88</td>
</tr>
<tr>
<td>No. stars in FOV(B)</td>
<td>1</td>
<td>8</td>
<td>89</td>
</tr>
<tr>
<td>((x,y,m)) (true)</td>
<td>10x3</td>
<td>4</td>
<td>90-119</td>
</tr>
<tr>
<td>((x,y,m)) (measured)</td>
<td>10x3</td>
<td>4</td>
<td>120-149</td>
</tr>
<tr>
<td>Rate gyro data ( (\omega_1) )</td>
<td>61</td>
<td>4</td>
<td>150-210</td>
</tr>
<tr>
<td>Rate gyro data ( (\omega_2) )</td>
<td>61</td>
<td>4</td>
<td>211-271</td>
</tr>
<tr>
<td>Rate gyro data ( (\omega_3) )</td>
<td>61</td>
<td>4</td>
<td>272-332</td>
</tr>
<tr>
<td>Gyro biases (true)</td>
<td>3</td>
<td>8</td>
<td>333-335</td>
</tr>
<tr>
<td>( \beta_{VN} ) (calc.)</td>
<td>4</td>
<td>8</td>
<td>336-339</td>
</tr>
<tr>
<td>( \beta_{BA} ) (calc.)</td>
<td>4</td>
<td>8</td>
<td>340-343</td>
</tr>
<tr>
<td>Covariance matrix for ( \beta_{VN} ) (calc.)</td>
<td>4x4</td>
<td>4</td>
<td>344-359</td>
</tr>
<tr>
<td>No. stars matched in FOV(A)</td>
<td>1</td>
<td>8</td>
<td>360</td>
</tr>
<tr>
<td>((x,y)) (calc.)</td>
<td>5x2</td>
<td>4</td>
<td>361-370</td>
</tr>
<tr>
<td>No. stars matched in FOV(B)</td>
<td>1</td>
<td>8</td>
<td>371</td>
</tr>
<tr>
<td>((x,y)) (calc.)</td>
<td>5x2</td>
<td>4</td>
<td>372-378</td>
</tr>
<tr>
<td>( \beta_{VN} ) (optimal estimate)</td>
<td>4</td>
<td>8</td>
<td>382-385</td>
</tr>
<tr>
<td>Gyro biases (optimal estimate)</td>
<td>3</td>
<td>8</td>
<td>386-388</td>
</tr>
<tr>
<td>( \beta_{VN} ) (integrated from previous value)</td>
<td>4</td>
<td>8</td>
<td>389-392</td>
</tr>
<tr>
<td>Covariance matrix for ( \beta_{VN} ) (opt. est.)</td>
<td>4x4</td>
<td>4</td>
<td>393-408</td>
</tr>
</tbody>
</table>
This program computes data for Processes B and C. Data consist of Euler parameters, assorted other parameters, and image coordinates for both fields of view. We use Euler parameters in this version to orient the 'V' frame relative to inertial frame and FOV(B) w.r.t. FOV(A).

Tom Strickerca...V.P.I. & S.U. 14 JANUARY 1981.

OVERLAP

OPTION BASE 1

FLOAT 5

DIM An(3,3), Gn(3,3), Ba(3,3), Bn(3,3), Av(3,3), Vg(3,3), Vn(3,3)

DIM Gl(3), Gs(3), G3(3), Bore(3)

DIM Ps(3), Vs(3), Ps(3), Lp(3), Bs(3)

DIM Pe(3), Ve(3), P(3), V(3), Vo(3), N(3), Voc(3)

DIM Bun(4), Bunest(4), Bug(4), Bba(4)

SHORT Xym(1,3), Xyr(1,3), Fov(100,4)

SHORT Xym(10,3), Xyr(10,3), Fov(100,4)

SHORT Xym(10,3), Xyr(10,3), Fov(100,4)

SHORT Xym(10,3), Xyr(10,3), Fov(100,4)

SHORT Xym(10,3), Xyr(10,3), Fov(100,4)

SHORT Xym(10,3), Xyr(10,3), Fov(100,4)

INTEGER Ta(29,2)

RANDOMIZE

Wread=0

! Come here to do another run.

PRINT USING "K"; "************* PROGRAM DATGEN *************"

INPUT 'Do you want to use realistic gyro rate history (Y/N)?', H$

PRINT USING "/k", R; "Do you want to use realistic gyro rate history (Y/N)?"

IF H$="N" THEN Plain_rates

IF Wread=1 THEN Star_Cat

PRINT USING "/k", R; "Place disk with gyro rates (Filename: 'Wtrue') in :FS,1...
...then push CONT."

PAUSE

ASSIGN #1 TO "ktrue:FS,1"

DISP "Reading rate gyro history....Please wait."

READ #1; Wl(*), W2(*), W3(*)

ASSIGN #1 TO *

Wread=1

DISP

GOTO Star.cat

Plain_rates:  *

Wread=0

MAT W1=ZER
530 MAT W2=(2^2)/5400  ! One rotation every 90 minutes.
540 MAT W3=ZER
550 !
560 Star_cat: !
570 PRINT USING /k/k;"Place star catalog disk (Filenames: 'Tab22' and 'Miss2 20') in :F8,1","...then push CONT."
580 PAUSE
590 !
600 N$="H"
610 INPUT 'Has table of star catalog cell positions been read-in (Y/N)?",N$
620 PRINT USING /k,A;"Has Table of star catalog cell positions been read-in? (Y/N) ";N$
630 !
640 IF N$="N" THEN Have_table
650 ASSIGN #1 TO "Tab22:F8,1"
660 READ #1,1
670 READ #1;Table(*)     ! Read in table of cell positions.
680 ASSIGN #1 TO *      ! Close this file.
690 !
700 Have_table:         ! Come here if table has been read-in.
710 ASSIGN #1 TO "pss220:F8,1"
720 !
730 Num=30           ! Number of records.
740 Len=2048         ! Bytes/record.
750 Fn$="Bdat: F8,1"  ! Dummy file name.
760 INPUT 'File name for simulation run ('Simnnn: F8,1'...where nnn is 3 num.): ",Fn$
770 ON ERROR GOTO 0 Error
780 PRINT USING /k/k;"File name for simulation run ('Simnnn: F8,1'...where nnn is 3 num.): ",Fn$
790 !
800 Done: !
810 ASSIGN #2 TO Fr$
820 CHECK READ #2
830 OFF ERROR
840 GOTO Ok
850 !
860 Error: !
870 PRINT USING /k;ERRM$
880 IF ERR<55 THEN STOP
890 INPUT 'Should I create this file for you?',N$
900 N$="H"
910 PRINT USING /k,k;"Should I create this file for you? ",N$
920 IF N$="N" THEN STOP
930 CREATE Fn$,Num,Len
940 GOTO Done
950 !
960 Ok: !
970 READ #2,1 ! Position pointer at beginning of file.
980 File=1
990 !
1000 Continue: !
1010 ! Set satellite orbit parameters.
1020 ! See write-up on "Orbit" for explanation.
1030 As=6652.56 ! Satellite major axis.
SET PARAMETERS FOR EARTH ORBIT.

Ae=.4959378*1E8
Ue=3.642972685E5
Ue=Ue*lJe
Ve0=29.784584E Total earth velocity (km/sec).
Earth_incl=3.3
Pe0(1):=-Ae I Put earth at vernal equinox for initial position (km).
Pe0(2):=0
Pe0(3)::8
Ve0(1)::0 I Initial earth velocity (km/sec).
Ve0(2)=Ve0*COS(-Earth_incl*PI/180) I Earth is heading downward in inertial frame.
Ve0(3)=Ve0*SIN(-Earth_incl*PI/188)
Re8=SQR(DOT(Pe8,Pe0))
Vei=SO(T(Ve,Ve8)) I Constants needed by Orbit.

COMPUTE VARIOUS EULER PARAMETER VALUES.

MAT Bv,;nom=;ER
MAT En=ZER
N$="N"
INPUT "Do you want variations in Euler para. relating V frame to Gyro frame (Y/N)?",N$
PRINT USIN;"/KtA";"Do you want variations in Euler para. relating V frame to Gyro frame (Y/N)? ",N$
1570 IF NS='N' THEN No_vg
1580 Evg(1)=Del
1590 Evg(2)=-2*Del          ! Amplitude of variations (radians).
1600 Evg(3)=3*Del          
1610 Evg(4)=4*Del          
1620
1630 Nvg(1)=3
1640 Nvg(2)=4          ! Frequency of variations.
1650 Nvg(3)=5
1660 Nvg(4)=6
1670
1680 No_vg:
1690 ! Compute the orientation of FOV(B) w.r.t. FOV(A).
1700
1710 Bbanom(1)=1/SQR(2)    ! SQR(1/SQR(2)+1)^2
1720 Bbanom(2)=1/SQR(2)    ! Bbanom(1)      ! Nominal values.
1730 Bbanom(3)=3        ! 1/(SQR(2)*4*Bbanom(1))
1740 Bbanom(4)=3        ! Bbanom(3)
1750
1760 MAT Ebias=ZER
1770 NS='N'
1780 INPUT 'Do you want variations in Euler para. relating B frame to A frame (<Y/N)?',NS
1790 PRINT USING "/K,K";'Do you want variations in Euler para. relating B frame to A frame (<Y/N)?","NS
1800 IF NS='N' THEN No_ba
1810 Eba(1)=10*Del
1820 Eba(2)=-30*Del          ! Amplitude of variations (radians).
1830 Eba(3)=20*Del
1840 Eba(4)=5*Del
1850
1860 Nba(1)=2
1870 Nba(2)=3          ! Frequency of variations (Oscillations/orbit).
1880 Nba(3)=5
1890 Nba(4)=4
1900
1910 No_ba:
1920 ! Set gyro bias values.
1930 MAT Biasnom=ZER
1940 MAT Ebias=ZER
1950 Bias$='N'
1960 INPUT 'Do you want time varying gyro biases (<Y/N)? ',Bias$
1970 PRINT USING "/K,K";'Do you want time varying gyro biases (<Y/N)? ",Bias$
1980 IF Bias$='Y' THEN No_bias
1990
2000 Biasnom(1)=Del
2010 Biasnom(2)=2*Del          ! Nominal values for biases (radians/sec).
2020 Biasnom(3)=3*Del
2030
2040 Ebias(1)=Biasnom(1)/2
2050 Ebias(2)=Biasnom(2)/3          ! Amplitude of variations (radians/sec).
2060 Ebias(3)=Biasnom(3)/4
2070
2080 Nbias(1)=4
2090 Nbias(2)=6          ! Frequency of variations (Oscillations/orbit).
Nbias(:3)=0  

No_bias:   ! Set various constants.

Sigxy=3.4E-3  ! Standard deviation of image coordinates in mm.
Sigm=.95     ! Sigm is the deviation in magnitude.

Xynoise$=""  

INPUT 'Do you want to add noise to image coordinates (Y/N)?',Xynoise$  
PRINT USING "/K,A";"Do you want to add noise to image coordinates(Y/N)?",Xynoise$

IF Xynoise$="N" THEN Sigxy=0

F1=72.425    ! Focal length of star sensor lens (mm).
Sigt=P(1/183) ! Standard deviation in Euler parameters. This is used if Process B is to run separately.

Sigm$=""  

INPUT 'Do you want to add noise to rate gyro data(Y/N)?',Gyro$
PRINT USING "/K,A";"Do you want noise added to rate gyro data (Y/N)?",Gyro$

IF Gyro$="N" THEN Sigm=0

Radius=6*P(1/186)  ! Angle from FOV center for accepting catalog stars.

PLOTTER IS 'GRAPHICS'
LORG 5

Get the initial gyro orientation w.r.t. the inertial system.

T0=0   ! Set reference time.
Time=0   ! Set initial time (seconds) for this run.

CALL Orbit(Time,Ps(*),Vs(*),Ps0(*),Vs0(*),T0,Us,As,Ds0,Rs0)
PRINT USING Form1;"Position (km) and velocity (km/sec) of Satellite:",Ps(*),Vs(*)
Sc=SQR:DOT(Ps,Fs))  ! Normalize the satellite position.
MAT G3=Ps/(Sc)  ! G3 is along position vector.
CALL Cross(G2(*),G3(*),Vs(*))  ! G2 is normal to orbit plane defined by position and velocity vectors.
G2=SQR:DOT(G2,C2))  ! Normalize G2.
CALL Cross(G1(*),G2(*),G3(*))  ! G1 = G2 cross G3.

FOR I=1 TO 3   ! Fill the Gn(*) rotation matrix.
Gn(1,1)=G1(I)
Gn(2,1)=G2(I)
Gn(3,1)=G3(I)
NEXT I

PRINT USING Form2;"Matrix GN:",Gn(*)
Coirpute Euler parameters for VN rotation.
First get true Euler parameters for VG rotation.

CALL Parturb(Bvn,gnom(*),Eug(*),Nug(*),Omega,0,Bug(*))

PRINT USINJ; Form3; "Bug...nominal Euler Parameters between frames V-C": Bugnom(*)

PRINT USINJ; Form3; "Bug...true Euler Parameters between frames V-G": Bug(*)

CALL Dircosb(Bug(*),Vg(*))

MAT Vn*Vg*In

Recover Euler parameters.

Bvn(1):=5*30*(Vn(1,1)+Vn(2,2)+Vn(3,3)+1)
B0:=Bvn(1)
Bvn(2):=(Vn(2,3)-Vn(3,2))/(4*B0)
Bvn(3):=(Vn(3,1)-Vn(1,3))/(4*B0)
Bvn(4):=(Vn(1,2)-Vn(2,1))/(4*B0)

PRINT USINJ; Form3; "Bvn...Initial Euler Parameters between frames V-N": Bvn(*)

Step=30 ! seconds between frames.
Delt=.5 ! seconds between gyro readouts.

Save all constants for this run.

PRINT 42,1;Hs,ls,Ps0(*),Vs0(*),Re,Ue,Pe0(*),Ve0(*),Incl,Sigxy,Sigm,Sigb,Sigv,Delt,Time,Step

PRINT 42;Bannon(*),Eba(*),Nba(*),Biasnom(*),Ebias(*),Nbias(*),Bvgnom(*),Ev

g(*),Nug(*)

! Now generate frames of data.

Timeloop:

FOR It=2 TO 30 ! number of frames.

DISP ' FRAME: ";It

PRINT USING "/K,DD,K"; "********** FRAME ":It; " **********

Locp over time interval to get true gyro rates.

FOR J=1 TO Step/Delt+1
R1(J)=W1(Ii+J)
R2(J)=W2(Ii+J)
R3(J)=W3(Ii+J)
NEXT J

Ii=1+Step*Delt
Dt=Time-T3

CALL Parturb(Eugnom(*),Eug(*),Nug(*),Omega,Dt,Bug(*))

CALL Dircosb(Eug(*),Vg(*))

PRINT USING Form2; "Matrix VG": Vg(*)

Call Runge-Kutta to compute true values vehicle frame.
CALL Runge(:Bvn(,),Time,Delt,Step,R1(*),R2(*),R3(*),Vg(*))

PRINT USING Fcrn4:"Satellite time from start of simulation: ",Time,"seconds"
PRINT USING Fcrn3:"Bvn...True Euler Parameters between frames V-N:",Bvn(*)

CALL Jircsb(Ivn(*),Vn(*))
!
Perturb Beta(BA) about their nominal values.
CALL Perturb(Ebanom(*),Eba(*),Nba(*),Omega,Dt,Bba(*))
CALL Jircsb(Eba(*),Ba(*))  ! Compute rotation matrix.

PRINT USING Fcrn3;"Bba...Nominal Euler Parameters between frames B-A:",Bbanom(*)
PRINT USING Fcrn3;"Bba...True Euler Parameters between frames B-A:",Bba(*)

CALL Jircsb(Eba(*),Ba(*))
PRINT USING Fcrn3;"Matrix BA :",Ba(*)
CALL Mat_uv(Ba(*),Av(*))
!
Compute AN rotation matrix.
MAT An=Av*N
!
Update the satellite and earth position.
CALL Orbit(Time,Ps(*),Vs(*),Ps0(*),Vs0(*),T0,Us,As,Ds0,Rs0)
CALL Orbit(Time,Pe(*),Ve(*),Pe0(*),Ve0(*),T0,Ue,Re,De0,Re0)
!
PRINT USING Fcrn1;"Position (km) and velocity (km/sec) of Satellite:",Ps(*),Vs(*)
PRINT USING Fcrn1;"Position (km) and velocity (km/sec) of Earth:",Pe(*),Ve(*)
!
MAT V=Vs+Vv;  ! Compute total velocity.
PRINT USING Fcrn1;"Total velocity of satellite (km/sec):",V(*)
!
Mag=SQR(DOT(V,V))
MAT Vc=V/(C)
!
FOR I=1 TO 4
    CALL Gauss(Sigb,Bvn(I),Bunest(I))
! Perturb Euler parameters.
    CALL Gauss(Sigb,Bbanom(I),Bbanest(I))
! These are used if only Proc. B is run.
    NEXT I
!
! Normalize Euler parameters.
MAT Bunest=Bunest/Mag
!
File=File+1
PRINT 12,File
PRINT 12,Bvn(*)
PRINT 12,Bunest(*)
PRINT 12,Bvq(*)
PRINT 12,Bvq(*)
PRINT 12,Baq(*)
PRINT 12,Baq(*)

! Position file pointer.
! Save true Euler parameters.
! Save estimated Euler parameters.
! Save Betas for frame G to V:Evq.
! Save estimated Bug.
! Save Betas for frame A to B - Bba
3620 PRINT #2;Ba(*); "Save Bba(est.)."
3630 PRINT #2;V(*)
3640 ! Save velocity vector.
3650 GCLEAR
3660 LOCATE 10,110,*0,100
3670 !
3680 Pass=0
3690 Pass_2:
3700 ! Come here for FOV(B)--Second pass through loop.
3710 SHOW -5.7,5.7,-4.4,4.4
3720 CLIP -5.7,5.7,-4.4,4.4
3730 FRAME
3740 AXES 1,1,0,0
3750 !
3760 FOR J=1 TO 3
3770 Bore:*J)=An(J,J)
3780 NEXT J
3790 ! Boresight direction cosines.
3800 PRINT USING Form2;"Matrix AN:",An(*)
3810 PRINT USING Form2;"Boresight unit vector:",Bore(*)
3820 !
3830 CALL Access(#1,Fov(*),Nfov,Bore(*),Sigma,Radius,Fld,Table(*),Voc(*))
3840 !
3850 Nm=0
3860 PRINT USING Form4;"Number of stars from the catalog=",Nfov
3870 MAT Xy:=ZER
3880 MAT Xym=ZER
3890 !
3900 FOR N=1 TO Nfov
3910 FOR I=1 TO 3
3920 L(I)=Fov(N,I)
3930 NEXT I
3940 CALL Photon(L(*),An(*),Xx,Yy) ! Compute x,y.
3950 Xx=Xx+F1
3960 Yy=Yy+F1
3970 IF A33(Xx)>5.7 THEN Skip ! Test for star in field.
3980 IF A33(Yy)>4.4 THEN Skip
3990 !
4000 Nm=Nm+1
4010 ! Got one!
4010 Xym(Nm,1)=Xx
4020 Xym(Nm,2)=Yy
4030 CALL Gauss(Sigxy,0,X)
4040 Xym(Nm,1)=Xx+X
4050 CALL Gauss(Sigxy,0,Y)
4060 Xym(Nm,2)=Yy+Y
4070 Mt=Fov(N,I)
4080 Xym(Nm,3)=Mt
4090 CALL Gauss(Sigm,0,Dm)
4100 Xym(Nm,3)=Mt+Dm
4110 !
4120 MOVE Xym(Nm,1),Xym(Nm,2)
4130 LABEL USING "K";"*
4140 IF Nm=10 THEN Output
4150 Skip: !
4160 NEXT N
PRINT USING Form4;"Number of stars in this FOV:",Nm
PRINT USING Form2;"True image coordinates (mm):",Xyt(*)
PRINT USING Form2;"Measured image coordinates (mm):",Xym(*)
PRINT USING Form4; Output:
PRINT Form;"Number of stars in field."
PRINT Form;"True and measured positions."
MAT Bn=Bn*N
MAT An=An*Bn
LOCATE 10,110,E,50
PASS=PASS+1
IF PASS=1 THEN Na=Nm
IF PASS=1 THEN PASS_2
IF Gyr$="N" THEN Skip_noise
DISP "Adding noise to gyro rates...please wait."
FOR J=1 TO Step/Delt
CALL Gauss(Siggy,0,X)
R1(J)=R1(J)+X
CALL Gauss(Siggy,0,X)
R2(J)=R2(J)+X
CALL Gauss(Siggy,0,X)
R3(J)=R3(J)+X
NEXT J
Skip_noise:
IF Bias$="N" THEN Skip_bias
CALL Biar Dt,Biasnom(*),Ebias(*),Nbias(*),Omega,Bias(*)
PRINT USING Form3;"Biases...true values:",Bias(*)
MAT R1=R1+Bias(1)
MAT R2=R2+Bias(2)
MAT R3=R3+Bias(3)
Skip_bias:
OUTPUT 0 USING "K,DD,K,DD,XX,DD";"Frame:",It,"Number of stars: ",Ha,Nm
DISP
PRINT Form;"Save measured gyro rates (60*3)"
PRINT Form;"Save true bias values (3)."
N$="N"
INPUT "Do you want to do another run(Y/N)?",N$
PRINT USING "/K,K","Do you want to do another run(Y/N)? ";N$
IF N$="Y" THEN Restart
Form1: IMAGE K/3(3<MD-DDDDE,X>/)
4720 Form2: IMAGE K-10<3(MD.DDDDDD,X)/>
4730 Form3: IMAGE K<4(MD.DDDDDD,X)/>
4740 Form4: IMAGE K,X,MDDDDD.DD,X,K
4750 I
4760 END
Combin

This program analyses data produced by simulation program Datgen, directing the data to subroutines for Process B (Proc-b) and Process C (Orbit, Runge and Kalman). Process B and Process C can be run separately (Process C alone only if Process B was run previously with these data) or they can be run together. Combin requests several parameters from the user: 1) the process noise standard deviation associated with the variation in interlock Euler parameters, $\beta_{BA}$, used in Kalman filter update of $\beta_{BA}$ (see Section 3 of the Final Report), and 2) the gyro bias "standard deviation" which controls the variations in the recovered gyro biases (see Section 4 of the Final Report). In addition, we can also offset, by a constant amount, the interlocks between the vehicle and gyro frame (nominally set to zero).

The data frames, read from an external file, are processed one at a time. For each frame, the Euler parameters, $\beta_{VN}$, and the associated covariance matrix from analysis of the previous frame, are integrated forward by subroutine Runge (for the first frame we can use the true values for $\beta_{VG}$ or some offset). Subroutine Orbit computes the position and velocity of the earth and satellite (the total velocity is used by Access to add aberration effects to the star direction cosines). We then call Proc-b to 1) match measured stars with specific catalog stars and 2) update the Euler parameters $\beta_{VN}$ and $\beta_{BA}$ via least-squares correction. These Euler parameters, the covariance matrix associated with $\beta_{VN}$, and the calculated image coordinates for all matched stars are saved by storing them at the end of the current data record.
Subroutine Kalman is then called to combine the integrated values for $\beta_{VN}$ with the corrected values from Process B analysis to yield an "optimal estimate" of $\beta_{VN}$ and the gyro biases, at the current time. These parameters and the $4 \times 4$ covariance matrix associated with the estimate of $\beta_{VN}$ are saved on the data file also.
Figure A10.2: Hierarchy of subroutine calls (left to right) and approximate order of calls top to bottom) for program Combin.
**PROGRAM: COMBINE**

00 ! T. STRIKWERDA ...... 14 JANUARY 1981.
90 ! This program combines processes B and C of Star Wars. This version uses
100 ! Euler parameters relating the "V" frame to the inertial frame.
110 ! This program also recovers the Euler parameters relating FOV(A) tc
120 ! FOV(3) and the gyro bias estimates.
130 !
140 OVERLAP
150 FLOAT 5
160 OPTION BASE 1
170 DIM Ps(3),Vs(3),Ps(3),Vs(3),Pe(3),Ve(3),Pe(3),Ve(3)
180 DIM T1(4),T2(3,3)
190 DIM Cov(8,8)
200 DIM Bun(4),Buntrue(4),Bug(4),Buns(4)
210 DIM Bba(4),Bbaatrue(4),Voc(3),Bbtrue(4),Bbalsq(4)
220 SHORT Xyma(10,3),Xymb(10,3),Xyc(5,2),Xycb(5,2),Xyt(10,3)
230 SHORT Cov(4,4),W1(61),W2(61),W3(61)
240 DIM Pk(7,7),Lam(7,7)
250 DIM Xk(7),Xktrue(7),Xkb(7),O(3,3),Qba(4,4),Pba(4,4)
260 COM Vg(3,3),INTEGER Table(529,2)
270 RAD
280 !
290 PRINT USING "K";"################################### PROCESS B AND C
300 !
310 NS="""H"
320 INPUT "Doing Proc B (Y/N)?",N$
330 PRINT USING "/K,A";"Doing Proc B (Y/N)? ",N$
340 Pb=0
350 IF N$="Y" THEN Pb=1
360 |
370 NS="""H"
380 INPUT "Doing Proc C (Y/N)?",N$
390 PRINT USING "/K,A";"Doing Proc C (Y/N)? ",N$
400 Pc=0
410 IF N$="Y" THEN Pc=1
420 |
430 IF Pb=0 THEN Have_table
440 |
450 PRINT USING "K";"Insert star catalog disk into F8,1....Then press CONT"
460 PAUSE
470 NS="".vn Set answer to blank.
480 INPUT "Has the table of star cell positions been read-in?",N$
490 PRINT USING "/K,K";"Has cell table been read-in? ",N$
500 IF N$="Y" THEN Have_table
510 |
520 ASSIGN #1 TO "Tab22,F8,1"
530 READ #1,1
READ #1; Table(*) ! Read in table of cell positions.
ASSIGN #1 TO #1 ! Close this file.

ASSIGN #2 TO "Mss220:F8,1" ! Star catalog file.

READ in table; ! Come here if this is a continuation.

INPUT "Input file name and device with simulation data (Simnnn:F8,0):",D
PRINT USING "/K,K";"Input file name and device with simulation data: ",D
ASSIGN #1 TO Dim$
BUFFER #1
READ #1,1;As,Us,Ps0(*),Vs0(*),Re,Ue,Pe0(*),Ve0(*),Incl,Sigxy,Sign,Sigb,S
Sigxy,Del,Time,Step
PRINT USING "/K,DD.DD";"Satellite orbit major axis (km): ",As
PRINT USING "/K,D.DDDE";"Earth orbit major axis (km): ",Re
PRINT USING "/K,D.DDDE";"Satellite orbit inclination (deg.): ",Incl
Sigxy=MA<(Sigxy,1E-3)
PRINT USING "/K,DD.DD";"Rate gyro data spacing (sec): ",Delt
PRINT USING "/K,DD.DD";"Runge-Kutta time step (sec): ",Step
PRINT USING "/K,D.DDDE";"Gyro standard deviation (rad/sec): ",Siggy
Siggy=MA<(Siggy,2.42E-6) ! Must keep gyro std. dev. non-zero.

Ds0=DOT(2p0,Vs0)
Rs0=SQR(DOT(Fs0,Ps0)) ! Constants for orbit calculation.
Delt=DOT(2p0,Ve0)
Re0=SQR(DOT(Fe0,Pe0))
T0=0 ! Set reference time.

REDim WI(3)
READ #1; WI(*) ! Read all the perturbation constants.
FOR I=1 TO 4
   Bbanom(I)=WI(I) ! Retrieve the nominal interlock values
   NEXT I ! from this list.

REDim WI(61)

IF Pb=6 THEN READ #1;Wba ! Read the weight for interlock recovery.
IF Pb=0 THEN Skip

INPUT "Input weight in arcseconds for interlock variance (2,5,etc.):",Wba
PRINT USING "/K,D.DDDE";"Input weight in arcseconds for interlock varian
cce (2,5,etc.): ",Wba

PRINT #1;Wba
MAT Pba=INH ! Set up interlock Kalman filter matrices.
MAT Pba=3ba*(1E-6) ! Set covariance matrix to large initial value.

MAT Qba=INH !
Qba(1,1)=1/8
Qba(1,2)=-1/6 ! Qba is the process noise matrix for the
interlock angles between FOV(A) and FOV(B).
The Q matrix has been converted to Euler parameters. NOTE: This matrix is valid for
(3,1,3) rotation of (0,90,0) only!
MAT Qba=Qba((Wba*4.848E-6)^2)  ! (3,1,3) rotation of (0,90,0) only!
MAT Pba=Pba  ! Can start with good estimate.

NOTE: This matrix is valid for
MAT Qb=Qba((Wba*4.848E-6)^2)  ! (3,1,3) rotation of (0,90,0) only!
MAT Pba=Pba  ! Can start with good estimate.

IF P=0 THEN Skip2
INPUT "Input Gyro Bias Standard Deviation (Degrees/Hr)", Sigbias
PRINT USING "/K,DD.DD";"Input Gyro Bias Standard Deviation (Degrees/Hr)"", Sigbias
Sigbias=Sigbias*PI/(180*3600)  ! Convert to radians/sec.
MAT Lam=ZER
FOR I=5 TO 7
   Lam(I,1:)=Sigbias^2  ! Set observation covariance matrix for gyro bias.
   NEXT I
MAT Pk=P*(IE-7)  ! Set covariance to large value.
MAT Q=ID1  ! Initialize process noise matrix.
PRINT USING Form3;"Q Matrix:", Q(*)
MAT Bvg=ZER
CALL Dir:osb(Bvg(*),Vg(*),0,TK(*),T2(*),T2(*),T2(*))
PRINTr USING Form3;"Matrix Vg for this run:",Vg(*)

C=3E5  ! Speed of light (km/sec).
MAT Xk=ZER
MAT Xk=ZER
Begin Loop Over Data Frames.

FOR Ifile=2 TO 30
   PRINT USING ">/K,DDD,K"; "RECORD NUMBER: ", Ifile, " 
   Pass=Pass+1
   READ #1,Ifile
   READ #1; Buntrue(*)
   PRINT USING Form1; "Bun....True Euler parameters between V and N frames:" ,Buntrue(*)
   REDIM Xkt(4)
   MAT Xkt=Buntrue
   REDIM Xkt(7)
   IF Pass=1 THEN MAT Xki=Xkt ! This causes displacement of first estimate.
      READ #1; Bun(*) ! These are not used unless we do Proc. B only.
   PRINT USING Form1; "Bun....Estimated Euler parameters between V and N frames:" ,Bun(*)
   READ #1; Bvn(*) ! True Euler parameters between V and G frames.
      CALL Dircon(Bvn(*),Vg(*),8,T2(*),T2(*),T2(*),T2(*)) ! Could use truth.
   READ #1; Bvn(*) ! same comment as Bun....not used.
   READ #1; Bbatrue(*) ! True Euler parameters between B and A frame.
   READ #1; Bbatrue(*) ! same....not used.
   IF Pass=1 THEN MAT Bba=Bbatrue ! Can help out by setting estimate=truth.
   READ #1; Bbatrue(*) ! True Euler parameters between B and F frame.
   PRINT USING Form1; "Bba....True Euler parameters between B and F frame:" ,Bbatrue(*)
   PRINT USING Form1; "Bba....Current Euler parameters between B and F frame:" ,Bba(*)
   READ #1; Voc(*)
   PRINT USING Form3; "Components of total velocity (km/sec):"; Voc(*)
   MAT Voc=Voc/(C)
   REDIM Xyn(1E1,3), Xymb(1E1,3)
   PRINT USING "K"; "Number of stars in each FOV:" 
   READ #1; Nfovma
   PRINT USING "K,X,DD"; FOV(A):", Nfovma
   READ #1; Kyma(*) ! True coordinates.
   READ #1; Kyma(*) ! Measured coordinates.
   REDIM Xynb(*)
   PRINT USING "K,X,DD"; FOV(B):", Nfovmb
   READ #1; Kymb(*) ! True.
   READ #1; Kymb(*) ! Measured.
   READ #1; W1(*), W2(*), W3(*) ! Read rate gyro data for each axis.
   REDIM Xkt(7)
   READ #1; Xkt(4), Xkt(6), Xkt(7) ! Read true bias rates.
IF P = 0 THEN Skip3
CALL Runge Tk, DelT, Step, W1(*), W2(*), W3(*), Xki(*), Pk(*), Q(*), Sigbias)
PRINT USING Form2; "Bun....Integrated Euler parameters between V and N frames","Xki(*)
REDIM Xki(4)
MAT Bun= Xki ! Estimates for Process B.
! IF Pass=1 THEN MAT Bun=Buntrue ! Can help out by setting est.=truth.
REDIM Xki(7)
Skip3: IF P = 0 THEN Tk=Tk+DelT
CALL Orbit Tk, Ps(*), Vs(*), Ps0(*), Vs0(*), TB, Us, As, Ds0, Rs0
CALL Orbit Tk, Pe(*), Ve(*), Pe0(*), Ve0(*), TB, Ue, Ae, De0, Re0
MAT Voc= Vs+Ve ! Total velocity.
PRINT USING Form3; "Components of total velocity (km/sec):"; Voc(*)
MAT Voc=Voc/C
IF Pb=0 THEN Skip4
MAT Bae= Bba ! Save the estimated interlock vector.
CALL Proc: b(12, Bun(*), Bba(*), Voc(*), Hfouma, Hfovmb, Ka, Kb, W, Sigxy, Xma(*), Xmb(*), Xycma(*), Xycb(*), Cou(*), Pba(*), Qba(*), Balsq(*))
OUTPUT & USING Form9; file, Hfouma, Hfovmb, Ka, Kb ! Progress indicator.
Form9; IMAGE 5 DD, XX

FOR I = 1 TO 4
FOR J = 1 TO 4
CoV(I, J)= Cov8(I, J)
NEXT J
NEXT I
PRINT #1; Bun(*), Bba(*) ! Save Process B results.
PRINT #1; Cou(*)
PRINT #1; Ka, Xycma(*)
PRINT #1; Kb, Xycb(*)
FOR I = 1 TO 4
FOR J = 1 TO 4
NEXT J
NEXT I
REDIM Xki(4)
MAT Xki=Bun
REDIM Xki(7)

FOR I = 1 TO 4
FOR J = 1 TO 4
NEXT J
NEXT I
REDIM Xki(4)
MAT T1=Xki
CALL Kalman(kt(*),Xki(*),Pk(*),Xkb(*),Lam(*))

Proc_b_failure:  ! Come here if Proc B failed to find stars in both FOV.

PRINT #1;Xki(*)  ! Save Proc. C results.
PRINT #1;Li(*)

FOR I=1 TO 4
  FOR J=1 TO 4
    Short=Pk(I,J)
    PRINT #1;Short
  NEXT J
NEXT I
PRINT #1;lbalsq(*)

Endloop:  !
PRINT "    (end of frame)"

NEXT I;ile

Stop:       PRINT "     THE END"
FOR I=1 TO 5
  BEEP
  WAIT 120
NEXT I

Form1: IMAGE /K,/,4<MD.DDDDD,X>
Form2: IMAGE /K,/,K,/,4<MD.DDDDD,X>,/,3<MD.DDDE,X>
Form3: IMAGE /K,3/<3<MD.DDDE,XX>

END
Access
determines which catalog cells the camera boresight lies in or near and then retrieves the star positions for the stars contained in those cells. The first step is to determine the polar angle and longitude angle of the boresight unit vector. These angles are converted to primary cell indices by dividing by the cell size. In a similar manner, we also determine three nearest neighbor cells. The location of each cell in memory or on a storage device and the number of stars in each cell are found by referring to a table. We then read the star data from these cells (consisting of direction cosines and magnitude) and compute the vector dot product of each star with the boresight unit vector. If the product is less than some specified tolerance we reject the star; this product is also used to sort the subcatalog by distance off the boresight. A list of up to 100 stars is returned to the calling program.

See Appendix 3 for details on the catalog format.
**ACCESS**

6260 ! Access gets stars from the catalog for cells surrounding the boresight.
6270 SUB Access(I1,lfov,Bore(*),Sigma,Radius,Voc(*),SHORT Fov(*))
6280 OPTION BASE 1
6290 DIM Ang(*),Cc(3)
6300 INTEGER K(4),J(4),Nt,Kk,M,N
6310 SHORT Mag,Num
6320 COM Vg(3,3),INTEGER Table(529,2)
6330 REDIM Fov(100,4)
6340 !
6350 DISP "Access"
6360 !
6370 Nt=22 ! Number of latitude bands for this catalog.
6380 Dphi=2*PI/(2*Nt+1) ! Latitude spacing.
6390 !
6400 Hfov=0
6410 Ctest=COS(Radius+Sigma) ! Maximum angle off the boresight.
6420 Phi=ACOS(Bore(3)) ! Polar angle.
6430 IF Phi=0 THEN Phi=Phi+2*PI
6440 !
6450 Lam=PI/2 ! Calculate longitude angle.
6460 IF Bor(2)=0 THEN Lam=Lam+PI
6470 IF Bor(1)=0 THEN Lam=ATN(Bore(2)/Bore(1))
6480 IF Bor(1)=0 THEN Lam=Lam+PI
6490 IF Lam=0 THEN Lam=Lam*2*PI
6500 !
6510 PRINT USING "/,K,X,DDD.D,X,K";"Polar angle:",Phi*180/PI,"Degrees"
6520 PRINT USING "/,K,X,DDD.D,X,K";"Longitude angle:",Lam*180/PI,"Degrees"
6530 !
6540 ! Calculate cell indices.
6550 IF Phi>Dphi THEN North ! Near north pole--special case.
6560 IF Phi<2*PI-Dphi+.5 THEN South ! Near south pole--special case.
6570 !
6580 K(1)=2*INT(Phi/Dphi+.5) ! Calculate two neighboring
6590 K(3)=2*INT(Phi/Dphi) ! latitude bands.
6600 IF K(3)=K(1) THEN K(3)=K(3)+2 ! Make sure we're on the correct
6610 IF K(1)>Nt THEN K(1)=2*Nt+1-K(1) ! side of equator.
6620 IF K(3)>Nt THEN K(3)=2*Nt+1-K(3)
6630 K(2)=K(1)
6640 K(4)=K(3)
6650 !
6660 Diam=2*PI/(2*K(1)+1) ! Now determine two neighboring cells
6670 J(1)=INT(Lam/Diam+.5) ! in each latitude band.
6680 J(2)=INT(Lam/Diam)
6690 IF J(2)=J(1) THEN J(2)=(J(1)+1) MOD (2*K(1)+1)
6700 J(1)=J(1) MOD (2*K(1)+1)
6710 Diam=2*PI/(2*K(3)+1)
6720 J(3)=INT(Lam/Diam+.5)
6730 J(4)=INT(Lam/Diam)
6740 IF J(4)=J(3) THEN J(4)=(J(3)+1) MOD (2*K(3)+1)
6750 \[ J(3) = J(3) \mod (2*K(3)+1) \]
6760 GOTO Around
6770 !
6780 North: ! Special case for north pole.
6790 K(1) = 0
6800 K(2) = 2
6810 K(3) = 2
6820 K(4) = 2
6830 J(1) = 0
6840 J(3) = INT(Lam/(2.PI/5)+.5)
6850 J(2) = J(3)+1 MOD 5
6860 J(4) = J(3)+1 MOD 5
6870 GOTO Around
6880 !
6890 South: ! Special case for south pole.
6900 K(1) = 1
6910 K(2) = 1
6920 K(3) = 1
6930 K(4) = 3
6940 J(1) = 0
6950 J(2) = 1
6960 J(3) = 2
6970 J(4) = INT(Lam/(2.PI/7)+.5)
6980 !
6990 Around: ! Skip north and south pole stuff.
7000 PRINT USING "/,K,/,2(4(DDD),X)/"; "Cell indices:",K(*),J(*)
7010 !
7020 FOR I=1 TO 4
7030 M = K(I)*K(J)+J(I)+1
7040 N = Table(1,1)
7050 READ #1, I
7060 Kk = Table(N,2)
7070 FOR K=1 TO Kk
7080 READ #1; Co(*), Mag, Num
7090 A = DOT(Co,Bcre) ! Compute cos of interstar angle.
7100 IF A < Test THEN Continue
7110 Nfou = Nfou+1 ! Star lies within range of boresight.
7120 Ang(Nfou) = F
7130 !
7140 Scale = 1 - DOT(Voc, Co)
7150 MAT Co = Co*Scale ! Add aberration effects.
7160 MAT Co = Co + Voc
7170 !
7180 FOR J=1 TO 3
7190 Fov(Nfou,J) = Co(J) ! Save this star.
7200 NEXT J
7210 Fov(Nfou,4) = Mag
7220 Continue: NEXT K
7230 Skip: !
7240 NEXT I
7250 !
7260 IF Nfou < 0 THEN CALL Sort(Ang(*), Fov(*), Nfou, 4) ! Sort stars by angle off boresight.
7270 !
7280 DISP
7290 SUBEND
Bias

Subroutine Bias computes bias terms which are later added to the rate gyro data. We have used a simple sinusoidal variation added to each nominal bias value. For input we specify the time, orbital frequency, nominal bias values, and amplitude and frequency of each variation.
This subroutine computes the bias rates to be added to gyro rates.

Option Base 1

! Bias(1) = Biasnom(1) + Ebias(1) * COS(Nbias(1) * Omega * Dt)
! Bias(2) = Biasnom(2) + Ebias(2) * SIN(Nbias(2) * Omega * Dt)
! Bias(3) = Biasnom(3) + Ebias(3) * SIN(Nbias(3) * Omega * Dt)

SUBEND
Cross

Subroutine Cross computes the vector cross-product of two vectors.
This subroutine computes the cross product of two vectors.

SUB Cross(R(*)+V1(*)+,V2(*))
OPTION BASE 1

FOR K=1 TO 3
K1=K MOD 3+1  ! Determine the order of multiplication.
K2=K1 MOD 3+1
R(K)=V1(K1)*V2(K2)+V1(K2)*V2(K1)
NEXT K

SUBEND
Deriv

This subroutine forms the right-hand-side of the matrix Riccati equation, which is integrated by subroutine Runge to propagate the covariance matrix. We use the partitioned form of the Riccati equation as presented in Appendix 8 of the Final Report.

The input data consist of the upper $4 \times 4$ portion, lower left $3 \times 4$ portion and lower right $3 \times 3$ portion of the covariance matrix. In addition, the subroutine requires the $3 \times 3$ process noise matrix and two portions of the matrices used for the state differential equations (see Appendix 8). The output is the time derivative of the upper $4 \times 4$ and lower left $3 \times 4$ portions of the covariance matrix evaluated for the current state.
** DERIV **

14890 ! This subroutine sets up the RHS of the matrix Riccati equation in
14900 ! partitioned form.
14910 SUB Deriv(P1(*),P21(*),P22(*),Q(*),A11(*),A12(*),Kn(*),Ln(*))
14920 OPTION BASE 1
14930 DIM T1(4,4),T2(4,4),A12t(3,4)
14940 !
14950 MAT T1=A11*F11
14960 MAT T2=A12*F21
14970 MAT T1=T1+T2
14980 MAT T2=TRN(T1)
14990 MAT Kn=T1+T2
15000 MAT A12t=TRN(A12)
15010 MAT T1=A12t*F
15020 MAT T2=T1+A12t
15030 MAT Kn=Kn+T2
15040 !
15050 MAT T1=P21*A11
15060 MAT T2=P22*A12t
15070 MAT Ln=T2-T1
15080 !
15090 SUBEND
Dircosb

This subroutine computes the rotation matrix between two coordinate frames as a function of the Euler parameters. If selected, the partial derivatives of the rotation matrix with respect to each Euler parameter, are calculated also. Refer to Section 2 of the Final Report for the form of the rotation matrix.
* IIRCOSB *
*
******************************************************************************
* SUB IIRCOSB(B(*),C(*),Ndc,Dc1(*),Dc2(*),Dc3(*),Dc4(*))
* OPTION BASE 1
* DISP "IIRCOSB"
* B0=B(1)
* B1=B(2)
* B2=B(3)
* B3=B(4)
* B12=B1+B1
* B22=B2+B2
* B32=B3+B3
* B02=B0+B0
* C(1,1)=B02+B12-B22-B32
* C(1,2)=2*(B1*B2+B0*B3)
* C(1,3)=2*(B1*B3-B0*B2)
* C(2,1)=2*(B1*B2+B0*B3)
* C(2,2)=B02+B12+B22-B32
* C(2,3)=2*(B1*B3+B0*B2)
* C(3,1)=2*(B1*B3+B0*B2)
* C(3,2)=2*(B1*B2+B0*B3)
* C(3,3)=B02-B12-B22+B32
* DISP "IIRCOSB"
* IF Ndc>1 THEN SUBEXIT ! Don't need partials.
* DISP "IIRCOSB"
* B0=B0+30
* B1=B1+31
* B2=B2+32
* B3=B3+33
* Dc1(1,1)=B3
* Dc1(1,2)=B3
* Dc1(1,3)=-B2
* Dc1(2,1)=-B3
* Dc1(2,2)=B3
* Dc1(2,3)=B1
* Dc1(3,1)=B2
* Dc1(3,2)=-B1
* Dc1(3,3)=B3
* Dc2(1,1)=B1
* Dc2(1,2)=B2
* Dc2(1,3)=B2
* Dc2(2,1)=B1
* Dc2(2,2)=B2
* Dc2(2,3)=B2
* Dc2(3,1)=B2
* Dc2(3,2)=B1
* Dc2(3,3)=B3

******************************************************************************
* Computes the direction cosine matrix using EULER parameters.
* 0880 SUB IIRCOSB(B(*),C(*),Ndc,Dc1(*),Dc2(*),Dc3(*),Dc4(*))
* 0890 OPTION BASE 1
* 0900 ! Compute the partials of C w.r.t. each beta.
* 0910 ! Do this for factor of two.
* 0920 ! Partials w.r.t. Beta0.
* 0930 ! Partials w.r.t. Beta1.
9350  Dc2(1,3)=B3
9360  Dc2(2,1)=B2
9370  Dc2(2,2)=-B1
9380  Dc2(2,3)=B3
9390  Dc2(3,1)=B3
9400  Dc2(3,2)=-B0
9410  Dc2(3,3)=-B1
9420  !
9430  Dc3(1,1)=-B2  ! Partials w.r.t. Beta2.
9440  Dc3(1,2)=B1
9450  Dc3(1,3)=-B0
9460  Dc3(2,1)=B1
9470  Dc3(2,2)=B2
9480  Dc3(2,3)=B3
9490  Dc3(3,1)=B3
9500  Dc3(3,2)=B3
9510  Dc3(3,3)=-B2
9520  !
9530  Dc4(1,1)=-B3  ! Partials w.r.t. Beta3.
9540  Dc4(1,2)=B3
9550  Dc4(1,3)=B1
9560  Dc4(2,1)=-B0
9570  Dc4(2,2)=-B3
9580  Dc4(2,3)=B2
9590  Dc4(3,1)=B1
9600  Dc4(3,2)=B2
9610  Dc4(3,3)=B3
9620  !
9630  DISP
9640  SUBEND
This subroutine computes the partial derivatives of the calculated image coordinates with respect to each of the four Euler parameters orienting the star tracker. We have rearranged the partial derivative calculation into a simple form. We want, for FOV(A) for example,

\[ \frac{\partial X_m}{\partial \beta_{V_n}} = \sum_{ij} \frac{\partial X_m}{\partial \beta_{V_n}} \frac{\partial \alpha_{N_{ij}}}{\partial \beta_{V_n}} \]  

(m = 1, 2, ... no. of stars \n = 0, 1, 2, 3).

We note that matrix \( \frac{\partial \alpha_{N}}{\partial \beta_{V_n}} = AV \cdot \frac{\partial \alpha_{V_n}}{\partial \beta_{V_n}} \) where AV is calculated by Mat-av and \( \frac{\partial \alpha_{V_n}}{\partial \beta_{V_n}} \) is calculated by Dircosb and this matrix is independent of the particular star.

The terms \( \frac{\partial X_m}{\partial \alpha_{N_{ij}}} \) are derivatives of the stellar colinearity equations with respect to the terms in the rotation matrix. If we rotate a star's direction cosines \( \{L\} \) into the FOV(A) frame, we have

\[ \{L\}' = AN \cdot \{L\} \]

Then, the normalized image coordinate \( X/f = L_1/L \) where \( f = \) lens focal length. The 3 x 3 matrix \( \frac{\partial X_m}{\partial \alpha_{N}} \) becomes

\[
\begin{bmatrix}
L_1 & L_2 & L_3 \\
L_1' & L_2' & L_3' \\
0 & 0 & 0 \\
L_1L_1 & L_2L_1 & L_3L_1 \\
L_1' & L_2' & L_3' \\
L_1' & L_2' & L_3'
\end{bmatrix}
\]

or simply the outer product:

\[
\begin{bmatrix}
1 \\
L_1 \\
L_2 \\
L_3
\end{bmatrix}
\begin{bmatrix}
L_1 & L_2 & L_3
\end{bmatrix}
\]
To obtain the derivative we can proceed to multiply each term in this $3 \times 3$ matrix by the corresponding term in $\frac{\partial AN}{\partial VN_n}$ and then sum all terms. However, it is not too difficult to see that this can be accomplished by writing

$$
\frac{\partial X_m}{\partial VN_n} = \begin{bmatrix} 1 & L_1 & L_2 \\ L_3 & 0 & L_1 \\ L_3 & L_2 & 0 \\ L_3 & L_2 & L_3 \\ \end{bmatrix} \begin{bmatrix} \frac{\partial AN}{\partial VN_n} \\ L_1 \\ L_2 \\ L_3 \\ \end{bmatrix},
$$

a form very suitable for computation. The partial derivatives of the $y$ coordinate are similar.
**IXDBETA**

8350 | Dxdbeta computes partial derivs of (x,y) w.r.t. Euler parameters.
8360 | \[
     d(x \text{ or } y) / d(\text{beta}(i)) = \text{Sum}[(d(x \text{ or } y) / d(C_{ij}))(d(C_{ij}) / d(\text{beta}(i)))]
\]
8370 | where 'i' represents mult. corresponding terms.
8380 | We have rearranged this sum into compact form used here--(see
8390 | notes of T. Strikwerda).
8400 | SUB Dxdbeta(Flc(*), R(*), Kk, C(*), Dc1(*), Dc2(*), Dc3(*), Dc4(*))
8410 | OPTION BASE 1
8420 | DIM T1(3), T3(3), T4(3), L(3) | Dimension temporary matrices.
8430 | !
8440 | DISP "Dxdbeta"
8450 | !
8460 | FOR K=1 TO Kk ; Loop over all stars in this FOV.
8470 | K1=K+K-1
8480 | K2=K1+1
8490 | FOR i=1 TO 3 ; Get direction cosines for this star.
8500 | L(i)=Flid(K, i)
8510 | NEXT I
8520 | MAT T1=C*L ; Compute direction cosines in FOV frame.
8530 | T3(1)=1/T1(3)
8540 | T3(2)=0
8550 | T3(3)=-T1(1)*T3(1)*T3(1)
8560 | T4(1)=0
8570 | T4(2)=T3(1)
8580 | T4(3)=-T1(2)*T3(1)*T3(1)
8590 | !
8600 | MAT T1=D:1*L
8610 | A(K1,1)=DOT(T1, T3) ; d(x)/d(\text{beta}0)
8620 | A(K2,1)=DOT(T1, T4) ; d(y)/d(\text{beta}0)
8630 | !
8640 | MAT T1=D:2*L
8650 | A(K1,2)=DOT(T1, T3) ; d(x)/d(\text{beta}1)
8660 | A(K2,2)=DOT(T1, T4) ; ! etc.
8670 | !
8680 | MAT T1=D:3*L
8690 | A(K1,3)=DOT(T1, T3)
8700 | A(K2,3)=DOT(T1, T4)
8710 | !
8720 | MAT T1=D:4*L
8730 | A(K1,4)=DOT(T1, T3)
8740 | A(K2,4)=DOT(T1, T4)
8750 | !
8760 | NEXT K
8770 | !
8780 | DISP
8790 | SUBEND
The primary purpose of this subroutine is to calculate the column vector of differences between the calculated and measured star coordinates. As input we pass the coordinate frame rotation matrix, and an array containing the direction cosines and measured coordinates for up to five paired stars.

Subroutine Phoegn is called for each star to compute the image coordinates. These are subtracted from the measured values to fill the deviation vector.
FILL-Y

7260  "Fill-y fills Dely(*) with deviation between measured and calc.
7270  "star positions.
7280  SUB Fill_y(Fld(*),Kk,C(*),Dely(*),Isum)
7290  OPTION BASE 1
7300  DIM Xx(2),Cosines(3)
7310  "
7320  DISP "Fill_y"
7330  "
7340  Idim=Isum+k+2
7350  REDIM Dely(Idim)
7360  IF Kk=0 THEN SLBEXIT
7370  "
7380  FOR K=1 TO Kk
7390    FOR I=1 TO 3
7400      Cosines(I)=Fld(K,I)  "Get direction cosines for this star.
7410    NEXT I
7420    CALL Phoeon(Cosines(*),C(*),Xx(1),Xx(2))  "Compute (x,y).
7430    FOR I=1 TO 2
7440      Isum=Isum+1
7450      Dely(Isum)=Fld(K,3+I)-Xx(I)  "Compute deviations from measured (x,y).
7460    NEXT I
7470  NEXT K
7480  "
7490  REDIM Dely(Isum)
7500  "
7510  DISP
7520  SUBEND
Gauss generates and adds pseudo-Gaussian noise to a variable. The method is to add together 12 random numbers between 0 and 1 and subtract 6. This yields a number from a pseudo-Gaussian distribution with a mean of 0 and a standard deviation of one. Obviously, this distribution will be truncated at ±6 sigma. This number is then scaled by the standard deviation and added to the mean value (both required as input) and the result returned to the calling program.
This subroutine adds Gaussian noise to a variable.

SUB Gauss(Sig, Mean, Var)

R = 0
FOR I = 1 TO J2
   R = R + RND
NEXT I

Var = Sig*(R-5) + Mean  ! Scale the noise by sigma and add to mean.

SUBEND
Kalman performs the Kalman filter calculations outlined in Section 4 of the Final Report. The integrated and calculated states and corresponding covariance matrices are passed to Kalman. Currently, we also provide the true state for comparison. The integrated state is replaced by the optimal state on return and the integrated covariance matrix is replaced by the updated matrix.
KALMAN

12100 | Kalman computes the optimal state estimate at each time.
12110 |
12130 !
12140 SUB Kalman(Xkt(*),Xki(*),P(*),Xkb(*),Lam(*))
12150 OPTION BASE 1
12160 DIM Xk(7),corr(7)
12170 DIM S1(7,7),S2(7,7)
12180 DIM Kal(7,7,7),Dev(7)
12190 !
12200 DISP "Kalman"
12210 U1=16
12220 U2=8
12230 !
12240 PRINT USING "/K"; "Kalman Filter State Estimation"
12250 !
12260 MAT S1=TRN(Fk) !
12270 MAT Pk=S1+S2 ! Ensure symmetric Pk.
12280 MAT Pk=(.5)+Pk !
12290 !
12300 FOR I=5 TO 7 ! Fill measurement vector with bias values.
12310 Xkb(I)=Xk(I) !
12320 NEXT I
12330 !
12340 MAT S1=Lam+P ! Add obs. cov. matrix to integrated cov. matrix.
12350 !
12360 MAT S2=TRN(S1) !
12370 MAT S1=S1+S2 ! Ensure symmetric matrix.
12380 MAT S1=(.5)+S1 !
12390 !
12400 MAT S2=INV(S1) !
12410 !
12420 MAT S1=TRN(S2) ! Ensure symmetric matrix.
12430 MAT S2=S2+S2 !
12440 MAT S2=(.5)+S2 !
12450 !
12460 MAT Kal=Pk*S2 ! Compute Kalman gain matrix.
12470 !
12480 MAT Dev=Xk-Xki ! Deviations between states—(Obs.-integ. state).
12490 !
12500 MAT Corr=Kal*Dev! Correction vector.
12510 MAT Xk=Xki+Corr ! Calc. optimal state.
12520 MAT S1=Kal*P ! Calc. updated covariance matrix.
12530 MAT Pk=Pk-31 !
12540 !
12550 MAT S1=TRN(Pk) !
12560 MAT Pk=Pk+31 ! Ensure symmetric matrix.
12570 MAT Pk=(.5)+Pk
12580 |
12590 PRINT USING "K": State Vectors

| Differences |

<table>
<thead>
<tr>
<th>(B-T)</th>
<th>(O-T)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

12610 |
12620 FOR I=1 TO 7
12630   PRINT USING Form2;Xk(I),Xki(I),Xkb(I),Xk(I),Xki(I)-Xkt(I),Xkb(I)-Xkt(I)
12640   NEXT I
12650 |
12660 REDIM Xk(4)
12670 Norm=SQR(DOT(Xk,Xk))
12680 PRINT USING Form1:"Norm of optimal estimate - 1!",Norm-1
12690 MAT Xk=Xk/Norm ! Normalize the optimal estimate.
12700 REDIM Xk(7)
12710 |
12720 MAT Xki=Xk ! Set state = optimal state...This is the starting
12730   estimate for the next frame.
12740 |
12750 Form1:(MAGE /Xk,MD.DDDDDDDDD/
12760 Form2:(MAGE  /MD.DDDDE,X)
12770 Form3:(MAGE  /MD.DDE,X)/
12780 |
12790 DISP
12800 SUBEND
Least

Least solves the least-squares problem for differential corrections to the Euler parameters. Formally, the solution is:

\[ \Delta \beta = (A^T W A)^{-1} A^T W \Delta x \]

where \( W \) is a weight matrix, \( A \) is a matrix containing partial derivatives of image coordinates with respect to Euler parameters, and \( \Delta x \) is a vector of differences between measured and calculated coordinates.

However, we have adopted a diagonal weight matrix and thus absorb the weights into \( A \) and \( \Delta x \) (in the calling program). Therefore, we write

\[ \Delta \beta = (A^T A)^{-1} A^T \Delta x, \]

the form used in this subroutine.

The covariance-like result \((A^T A)^{-1}\) is returned to the calling program along with the corrections, \( \Delta \beta \).
This routine computes \((A^T A)^{-1} A^T Dv\) where \(Dv\) is the vector of deviations.

```plaintext
SUB Least(*,Dx(*),Cov(*),Dy(*),Idim,Jdim)
OPTION BASE 1
DIM T1(Jdim,Jdim),T2(Jdim),At(Jdim,Idim)

! This routine computes \((A^T A)^{-1} A^T Dv\)

MAT At=TRN(A)
MAT T1=At*3
MAT Cov=INV(T1)  ! Cov = \((A^T A)^{-1}\)
MAT T2=At*Dy
MAT Dx=Cov*T2

DISP
SUBEND
```
Least-1

This subroutine performs an iterative least-squares correction using coordinate data from one field of view. The current values of $\beta_{VN}$ are passed to Least-1, along with the rotation matrix $AV$ or $BV$ and the matrix containing direction cosines and measured image coordinates for paired stars for that field of view. The corrected Euler parameters are returned along with the covariance matrix result from the least-squares correction.

The method employed is to first call Dircosb to compute the rotation matrix $VN$ and its partial derivatives. Fill-y then computes the differences between measured and calculated image coordinates. Pre-mult converts partial derivatives of $VN$ to derivatives of $AN$ or $BN$ by multiplying by $AV$ or $BV$, respectively. Dxdbeta uses these partials to compute the partials of image coordinates with respect to Euler parameters, $\beta_{VN}$. The last row of the derivative matrix is filled with the constraint equation.

Finally, subroutine Least is called to compute corrections to $\beta_{VN}$. If these are small enough we return to the calling program. Otherwise, we iterate again, up to a limit of six times.
**LEAST-SQUARES CORRECTION FOR ONE FOV**

```
1430 SUB Least_1(Bun(*), Av(*), Fld(*), Kmax, W, Cov(*), Converge)
1440 OPTION BASE 1
1450 DIM Saveb(4), Bs(4), Dun1(3,3), Dun2(3,3), Dun3(3,3), Dun4(3,3), An(3,3), Vn(3,3)
1460 DIM Delx(4), Dely(11), R(11,4)
1470 MAT Saveb=Bun ! Save the original values of Euler parameters.
1480 !
1490 PRINT USING "/K"; "Least-Squares Correction For One FOV"
1500 FOR It=1 TO 6
1510 CALL Dir:cosb(Bun(*), Vn(*), 1, Dun1(*), Dun2(*), Dun3(*), Dun4(*))
1520 MAT An=Av+Vn
1530 Isum=0
1540 REDIM Delx(Kmax*2)
1550 CALL Fill_y(Fld(*), Kmax, An(*), Dely(*), Isum)
1560 Sq1=SQR(DOT(Dely, Dely)/(Kmax*2))
1570 PRINT USING "/,K,D.DDDDE"; "RMS error for normalized image coordinates: ", Sq1
1580 !
1590 REDIM A(Kmax*2+1,4), Dely(2*Kmax+1)
1600 CALL Pre_mult(A(*), Dun1(*), Dun2(*), Dun3(*), Dun4(*))
1610 CALL Dxdeta(Fld(*), A(*), Kmax, An(*), Dely(*), Dun1(*), Dun2(*), Dun3(*), Dun4(*))
1620 !
1630 FOR i=1 TO 4
1640 A(2*Kmax+1,i)=2*W*Bvn(i) ! Constraint equation.
1650 NEXT i
1660 !
1670 Dely(2*Kmax+1)=W*(1-DOT(Dvn, Bun))
1680 !
1690 CALL Least(A(*), Delx(*), Cov(*), Dely(*), Kmax*2+1, 4)
1700 !
1710 MAT Bs=Saveb
1720 !
1730 PRINT USING "/,K"; " Beta(old) Beta(new) Delta(Beta)"
1740 FOR i=1 TO 4
1750 Bun(i)=Bvn(i)+Delx(i)
1760 PRINT USING "/X3,M.D.DDDDDDD.XXX"; Bs(i), Bun(i), Delx(i)
1770 NEXT i
1780 Deu=SQRT(DOT(Delx, Delx)/4)
1790 PRINT USING "/,K,D.DDDDE"; "RMS change in Euler parameters: ", Deu
1800 !
1810 IF Deu<1E-6 THEN Morestars ! Small corrections...exit loop.
1820 NEXT I:
1830 !
1840 Nosoln: PRINT 

****** LEAST-SQUARES FOR ONE FOV DID NOT CONVERGE ******
1850 CONverge=0 ! Failure of least-squares.
1860 MAT Bun=Saveb ! Replace Euler parameters with original values.
1870 MAT Cov=ZER
1880 PRINT USING Form1; "Number of iterations: ", It-1, Converge
```
11940  SUBEXIT
11950  
11960  Morestars: Converge=1
11970  MAT Cov=Cov*(Sql*Sql)  ! Compute cov. matrix...mult. by sigma**2.
11980  PRINT USING Form1;"Number of iterations:",It,Converge
11990  Form1: IMAGE K,X,DD,XXX,"Converge=";D
12000  
12010  SUBEND
Least-2

Least-2 updates the Euler parameters $\beta_{VN}$ and $\beta_{BA}$ via least-squares differential correction, using between 3 and 5 matched stars from each FOV. Both $\beta_{VN}$ and $\beta_{BA}$ should initially be very near their final corrected values ($\beta_{VN}$ has been corrected by Pair-It and $\beta_{BA}$ does not vary rapidly); thus the least-squares requires only 2 or 3 iterations to converge. Since this also means the derivatives do not vary substantially between iterations, we can use the secant method (subroutine Secant) to update the matrix of partial derivatives used in the least-squares.

As input we pass the current values of $\beta_{VN}$ and $\beta_{BA}$ and the arrays containing the direction cosines and measured coordinates for up to 5 stars per FOV. We return the updated Euler parameters $\beta_{VN}$ and $\beta_{BA}$ and the $4 \times 4$ covariance matrix associated with $\beta_{VN}$. For each least-squares iteration we compute the differences between measured and calculated images for corresponding stars and the change in these differences compared with the previous iteration (to be used by Secant). On the first iteration we calculate the exact derivatives of image coordinates with respect to both $\beta_{VN}$ and $\beta_{BA}$ (via calls to Dircosb, Mat-av, Pre-mult, Post-mult and Dxdbeta - see Appendix 7). The last several rows of the derivative matrix are filled with the constraint equations, one for each set of Euler parameters (multiplied by an appropriate weight) $\beta_{BA}$, see Section 3 and Appendix 7). A call to Least returns corrections to all eight Euler parameters; if these are small we return, otherwise we iterate again, using Secant to update derivatives.
DIM A(2,6,8), Del y(26), Ddy(26), At l(16,4), Rt2(10,4), Del x(8), Bbasave(4)

DIM An:3,3),Bn(3,3)

DIM Av:3,3), Da 1(3,3), Dav2(3,3), Dav4(3,3), Dav3(3,3)

DIM Ba:3,3), Dba.1 (3,3), Dba2(3,3), Dba3(3,3), Dba4(3,3)

DIM Vn:3,3), Dvn1(3,3), Dvn2(3,3), Dvn3(3,3), Dvn4(3,3)

PRINT "JJSIN;","K", "Least-Squares Correction For Two FOY"

PRINT "JJSIN;",KD,K,D,K" Correct orientation using ",Ka," stars from FOY (A) and ",Kb," stars from FOV (B).

Kk=Ka2*Kb2

Ipass2=I

Jdim=8

Idim=Ka2+Kb2+2+4

Mat Dely=ZEF!

Mat A=.!ER

Mat Cov/S=flF!

Mat Bba&sav?e:Bba. ISave Euler parameters in case of failure.

FOR it=1 TO 4

Exact:

Isum=3

CALL Dirccsb(Bun(*),Vn(*), Ipas 2,Dun1(*),Dun2(*), Dun3(*), Dun4(*))

CALL Dirccsb(Bba(*),Ba(*), Ipas2,Dbal(*),Dba2(*), Dba3(*), Dba4(*))

CALL Mat_ev(Ba(*),Dbal(*),Dba2(*),Dba3(*), Dba4(*), Av(*), Ipas2,Davl(*), Dav2(*), Dav3(*), Dav4(*))

Mat An=Ru+Vn

Mat Bn=Ra*An

REDIM Dcy(Kk), Dely(Kk)

MAT Ddy=Dely

CALL fill_y(Fl da(*), Ka,An(*), Dely(*), Isum) ! Deviations for FOY(A).

CALL fill_y(Fl db(*), Kb,Bn(*), Dely(*), Isum) ! Deviations for FOY(B).

MAT Ddy=Dcy-Dely

Sqt=S2F(DCT(Dely,Dely)'Kk)

PRINT USING "/",K,K,D,K,D,D,D"RMS error in normalized image coordinate s: Sqt

IF Sqt<Sqt THEN Decreasing

Ipass2=I ! Flag to compute exact derivs. because the

Sqt=1E40 ! image error is increasing.

GOTO Exact
Decreasing: Come here is solution is converging.

Sqt=Sqt

REDIM Jdy(Idim)

IF Ipas<2=2 THEN Secant_method

Ipas<2=2

CALL pre_mult(Av(*),Dun1(*),Dun2(*),Dun3(*),Dun4(*))

CALL post_mult(Vn(*),Dav(*),Dav2(*),Dav3(*),Dav4(*))

CALL D:dbeta(Flva(*),At1(*),Ka,An(*),Dun1(*),Dun2(*),Dun3(*),Dun4(*))

CALL D:dbeta(Flva(*),At2(*),Ka,An(*),Dav1(*),Dav2(*),Dav3(*),Dav4(*))

FOR I=1 TO 4
  FOR K=1 TO Ka2
    ACK<2I)=At1(K,I) ! Fill A(*) with FOV(A) derivatives.
    ACK<2I+4)=At2(K,I)
    NEXT K
  NEXT I

! Compute partial deriv. of (x,y) for FOV(B).

CALL pre_mult(Bv(*),Dun1(*),Dun2(*),Dun3(*),Dun4(*))

CALL post_mult(An(*),Dav(*),Dav2(*),Dav3(*),Dav4(*))

CALL pre_mult(Bv(*),Dav1(*),Dav2(*),Dav3(*),Dav4(*))

MAT Dav=Ibal+Dav1
MAT Dav2=Iba2+Dav2
MAT Dav3=Iba3+Dav3
MAT Dav4=Iba4+Dav4

CALL D:dbeta(Flva(*),At1(*),Kb,Bn(*),Dun1(*),Dun2(*),Dun3(*),Dun4(*))

CALL D:dbeta(Flva(*),At2(*),Kb,Bn(*),Dav1(*),Dav2(*),Dav3(*),Dav4(*))

FOR I=1 TO 4
  FOR K=1 TO Kb2
    ACK<2K,I)=At1(K,I) ! Fill A(*) with FOV(B) derivatives.
    ACK<2K,1+4)=At2(K,I)
    NEXT K
  NEXT I

GOTO -sq

Secant_method:

CALL Secart(A(*),Delx(*),Dy(*),Idim,Jdim)

LSq: REDIM Dly(Idim)

FOR I=1 TO 4
  ACK<2I+)=Bun(I)*W<2 ! Constraint eq. for Bun.
  ACK<2I+1)=0
  ACK<2I+2)=Bba(I)*W<2 ! Constraint eq. for Bba.
  ACK<2I+3)=0
  NEXT I

Dly(1)=W<1-DOT(Bun,Bun)) ! More constraint eq.

Dly(2)=W<1-DOT(Bba,Bba)) !

MAT Dly=Dly

CALL least(A(*),Delx(*),Cov8(*),Dly(*),Idim,Jdim)
PRINT USING "/,K,K";
Euler parameters and corrections:" , B
(V-N) delta-B(V-N) B(B-A) delta-B(B-A)

FOR I=1 TO 4
Bun(I)=Evun(I)+Delx(I)
Bba(I)=Eba(I)+Delx(4+I)
PRINT USING "4<MD.DDDDDDD,XX>";Bun(I);Delx(I);Bba(I);Delx(4+I)
NEXT I

Dv=SUM(DCT(Delx,Delx)/8)
PRINT USING "/,K,X,D.DDDE";"RMS change in Euler parameters:";Dev
IF Dev<1E-7 THEN Covariance

NEXT I
GOTO Two_failed

PRINT "!
(End of least-squares for two FOV)
MAT CovB=CovB*(Sqrt2) !Mult.([A transpose]*A)(inverse) by sigma^2.
PRINT USING "8<MD.DDE,X>";CovB(*)
SUBEXIT

Two failed: !
PRINT "--->--->---->--->LEAST-2 FAILED <---<---<---<---<---"
This subroutine computes part of the right hand side of kinematic differential equations governing Euler parameters. For input data 
requires the current Euler parameters, $\beta_{VN}$, the rate gyro data and the rotation matrix $VG$ to rotate the gyro rates from the gyro to vehicle frame. We can express the differential equations as

\[
\{\dot{\beta}\} = [\omega]\beta
\]

\[
= [\beta]\omega
\]

(see Section 4 and Appendix 8 of the Final Report for details); fills matrices $[\omega]$ and $[\beta]$. These two forms are also needed for integrating the matrix Riccati equation for covariance propagation.
**MATLAB**

12890 SUB Ma:a(F,H(*),X(*),A11(*),A12(*))
12900 OPTION BASE 1
12910 COM Yg:3,3)
12920 DIM A12p(4,3),hU(3)
12930 |
12940 MAT Wv=Wg*4     I Rotate gyro rates into V frame.
12950 MAT Wv=Wv*(.5)  I Divide by 2 now instead of later.
12960 W1=Wv(1)
12970 W2=Wv(2)
12980 W3=Wv(3)
12990 |
13000 | Calc. matrix A11=(B0,B1,B2,B3)DOT)/D((B0,B1,B2,B3)
13010 | where B0,B1,B2,B3 are Euler parameters.
13020 |
13030 A11(1,1)=0
13040 A11(1,2)=-41
13050 A11(1,3)=-42
13060 A11(1,4)=-43
13070 |
13080 A11(2,1)=W1
13090 A11(2,2)=0
13100 A11(2,3)=W3
13110 A11(2,4)=-42
13120 |
13130 A11(3,1)=W2
13140 A11(3,2)=-43
13150 A11(3,3)=0
13160 A11(3,4)=W1
13170 |
13180 A11(4,1)=W3
13190 A11(4,2)=W2
13200 A11(4,3)=-41
13210 A11(4,4)=0
13220 |
13230 IF F=0 THEN SUEEXIT
13240 |
13250 | Calc. matrix A12 = -D((B0,B1,B2,B3)DOT)/D((W1,W2,W3)
13260 | = D((B0,B1,B2,B3)DOT)/D(b1,b2,b3)
13270 |
13280 B0=X(1)*.5
13290 B1=X(2)*.5
13300 B2=X(3)*.5
13310 B3=X(4)*.5
13320 |
13330 A12p(1,1)=B1
13340 A12p(1,2)=B2
13350 A12p(1,3)=B3
13360 !
13370 A12p(2,1)=-10
13380 A12p(2,2)=12
13390 A12p(2,3)=-12
13400 !
13410 A12p(3,1)=-13
13420 A12p(3,2)=-10
13430 A12p(3,3)=12
13440 !
13450 A12p(4,1)=82
13460 A12p(4,2)=11
13470 A12p(4,3)=-10
13480 !
13490 MAT A12=A12p*Vc ! Multiply by rotation matrix.
13500 !
13510 SUBEND
This subroutine calculates the rotation matrix, AV, between the vehicle and star tracker (A) frame. The matrix, BA, between star tracker frames is required for input. If selected, the partial derivatives of AV with respect to elements of BA are calculated. (See Appendix 7 for details).
**MAT_AV**

```plaintext
10220 SUB Mat_av(Ba(*),Da1(*),Da2(*),Da3(*),Av(*),Der,Da0(*),Iav(*),
    Dav(*),Dav0(*),Iav1(*))
10230 OPTION BASE 1
10240 DIM T1(3,3),T2(3,3),T3(3,3),T4(3,3)
10250 !
10260 DISP "Mat_av"
10270 !
10280 D1=1/SQR(2+2*Ba(3,3)) ! Two useful factors.
10290 D2=1/SQR(2-2*Ba(3,3))
10300 !
10310 Av(1,1)=Ba(3,1)*D1 ! Compute matrix AV from BA.
10320 Av(2,1)=Ba(3,2)*D1
10330 Av(3,1)=.5*D1
10340 !
10350 Av(1,2)=Ba(3,1)*D2
10360 Av(2,2)=Ba(3,2)*D2
10370 Av(3,2)=.5*D2
10380 !
10390 Av(1,3)=-2*Ba(3,2)*D1*D2
10400 Av(2,3)=2*Ba(3,1)*D1*D2
10410 Av(3,3)=0
10420 !
10430 DISP
10440 !
10450 IF Der<>1 THEN SUBEXIT ! Leave SUB if we don't need partials.
10460 !
10470 DISPB "Mat_av"
10480 !
10490 MAT T1=ZER ! Compute d(AV(*))/d(BA(3,1)).
10500 T1(1,1)=D1
10510 T1(1,2)=D2
10520 T1(2,3)=2*D1*D2
10530 !
10540 MAT T2=ZER ! Compute d(AV(*))/d(BA(3,2)).
10550 T2(2,1)=T1(1,1)
10560 T2(2,2)=T1(1,2)
10570 T2(1,3)=-T1(2,3)
10580 !
10590 T3(1,1)=-Av(1,1)*D1*D1
10600 T3(2,1)=-Av(2,1)*D1*D1
10610 T3(3,1)=.5*D1
10620 T3(1,2)=Av(1,2)*D2*D2
10630 T3(2,2)=Av(2,2)*D2*D2
10640 T3(3,2)=D2*.5
10650 T=Ba(3,3)*3+(D1*D2)^3
10660 T3(1,3)=Ba(3,2)*T
10670 T3(2,3)=Ba(3,1)*T
```
10680  T3(3,3)=0
10690  !
10700  MAT  Dav[0]=T1+(Dta0(3,1))  ! Compute d(AV(*))/d(Bba(1)).
10710  MAT  T4=T2*(BbaE(3,2))
10720  MAT  Dav[0]=Dav[0]+T4
10730  MAT  T4=T3*(BbaE(3,3))
10740  MAT  Dav[0]=Dav[0]+T4
10750  !
10760  MAT  Dav[1]=T1+(Dta1(3,1))  ! Compute d(AV(*))/d(Bba(2)).
10770  MAT  T4=T2*(Bba1(3,2))
10790  MAT  T4=T3*(Bba1(3,3))
10810  !
10820  MAT  Dav[2]=T1+(Dta2(3,1))  ! Compute d(AV(*))/d(Bba(3)).
10830  MAT  T4=T2*(Bba2(3,2))
10850  MAT  T4=T3*(Bba2(3,3))
10870  !
10880  MAT  Dav[3]=T1+(Dta3(3,1))  ! Compute d(AV(*))/d(Bba(4)).
10890  MAT  T4=T2*(Bba3(3,2))
10910  MAT  T4=T3*(Bba3(3,3))
10930  !
10940  DISP
10950  SUBEND
Orbit

We use Herrick's "f and g" solution for a two body case (see Reference 6) to update the satellite and earth position and velocities. For either the satellite or earth we set:

\[ r_0 = (\mathbf{r}_0 \cdot \mathbf{r}_0)^{1/2} \]  
\[ = (X_0^2 + Y_0^2 + Z_0^2)^{1/2} \]  
\[ V_0 = (\mathbf{V}_0 \cdot \mathbf{V}_0)^{1/2} = (\dot{r}_0 \cdot \dot{r}_0)^{1/2} \]  
\[ = (\dot{X}_0^2 + \dot{Y}_0^2 + \dot{Z}_0^2)^{1/2} \]  
\[ D_0 = \mathbf{r}_0 \cdot \mathbf{V}_0 \]  
\[ = X_0 \dot{X}_0 + Y_0 \dot{Y}_0 + Z_0 \dot{Z}_0 \]  
\[ 1/a = 2/r_0 - \mathbf{v}_0 \cdot \mathbf{u} \]  
\[ \mathbf{u} = GM \]  
\[ t_0 = \text{initial time} \]

In the current version of this program we have set both orbits to be circular. The inclinations of the earth orbit is 23.5° and the satellite orbit is 70°.

To obtain the position and velocity of either body at some later time, \( t \), we solve the following equation for \( M \) by Newton's method:

\[ u^{1/2}(t - t_0)a^{-3/2} = M - (1 - r_0/a)\sin M + D_0(1 - \cos M)(ua)^{1/2} \]

using the initial estimate:

\[ M = u^{1/2}(t - t_0)a^{-3/2}. \]

Then, the position \((X, Y, Z)\) at time \( t \) is:
\[
\begin{align*}
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} &= \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + \begin{bmatrix} \dot{X}_0 \\ \dot{Y}_0 \\ \dot{Z}_0 \end{bmatrix} \\
\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} &= \dot{f} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + \dot{g} \begin{bmatrix} \dot{X}_0 \\ \dot{Y}_0 \\ \dot{Z}_0 \end{bmatrix}
\end{align*}
\]

where

\[
\begin{align*}
f &= 1 - a(1 - \cos M)/r_0 \\
g &= (t - t_0) - a^{3/2}(M - \sin M)u^{-1/2}
\end{align*}
\]

Also, the velocity \((\dot{x}, \dot{y}, \dot{z})\) at time \(t\) is:

\[
\begin{align*}
\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} &= \dot{f} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + \dot{g} \begin{bmatrix} \dot{X}_0 \\ \dot{Y}_0 \\ \dot{Z}_0 \end{bmatrix}
\end{align*}
\]

where \(r = (x^2 + y^2 + z^2)^{1/2}\)

\[
\begin{align*}
\dot{f} &= -(ua)^{1/2} \sin M/rr_0 \\
\dot{g} &= 1 - a(1 - \cos M)/r.
\end{align*}
\]
This subroutine computes the orbital position and velocity using Herrick's F and G solution. (See J.L. Junkins Text)

SUB Orbit(T, P(*), V(*), P0(*), V0(*), T0, U, A, D0, R0)
OPTION BASE 1
DIM P1(3), P2(3)

X=1-R0-A
Y=D0/SQR(U*A)
Rho=SQR(U*(T-T0)/SQR(A*A+A))
Phi=Rho
Dphi=1

FOR I=L TO 18
Find Phi by Newton's method.
Cphi=COS(Phi)
Sphi=SIN(Phi)
IF ABS(Dphi)<1E-5 THEN Got_it
Rhoc=Phi-X*Sphi+Y*(1-Cphi)
Drdp=1-X*Cphi+Y*Sphi
Dphi=(Rho-Rhoc)/Drdp
Phi=Phi+Dphi
NEXT I

PRINT '****** HELP ****** ORBIT DID NOT FIND PHI !!!!!!!!'

Got_it: Newton's method worked.

F=1-A*(1-Cphi)/R0
G=T-T0-A*3(R/U)*Sphi
MAT P1=(F)*P0
MAT P2=(C)*V0
Update position.
MAT P=P1+P2

R=SQR(OOT(P))
Fd=-SQR(U*:;*Sphi/(R*R0)
Gd=-SQR(U*3*:Sphi/(R*R0)
MAT P1=(Fd)*P0
MAT P2=(Gd)*V0
Update velocity.
MAT V=P1+P2
SUBEND
The task of Pair-it is to identify measured stars with specific catalog stars. We do this in two ways. The first method compares the cosine of the interstar angle between measured star pairs with the cosine of interstar angles of catalog star pairs. If there is a match we perform a least-squares correction (by calling Least-1) to refine the attitude estimate.

The second method to match stars uses the improved attitude estimate to mathematically project all the sub-catalog stars onto the focal plane and then compare each position with measured stars. We require at least three matches to confirm the attitude found by least-squares. The direction cosines and measured coordinates for each confirmed star image (up to 5 stars) are stored in an array and returned to the calling program for later processing.

The confirmation tests discussed above require an error tolerance between projected and measured stars in order to accept or reject a specific catalog star. The technique we use is discussed below. We first calculate, for one star of the initial matched pair, the angular size of twice the estimated one-sigma error (10% pixel) as seen from the midpoint between images and perpendicular to the line connecting the two stars. Then, for stars more distant than one-half the pair separation, we scale the angle by the distance from the midpoint in order to get the tolerance for each star. For nearer stars we simply use the estimated two-sigma image error. This technique helps to account for rotation errors due to displacements of one or both stars of the initial pair, normal to the separation vector. (See Figure A10.3).
Let: \( \tan \varepsilon = \varepsilon = \frac{2\sigma}{\rho} \)

where \( \rho \) = one-half of separation between the initial matched pair,

and \( \sigma \) = estimated error (1-sigma) in position of star centroid.

Then: \( 2\sigma' = \varepsilon \rho' \) (for \( \rho' > \rho \))

\( 2\sigma'' = 2\sigma \) (for \( \rho'' \leq \rho \))

where \( \rho' \) and \( \rho'' \) are distances from midpoint \((x_c, y_c)\) between stars of initial pair.

Figure A10.3: Calculation of error tolerance values to be used for matching measured and calculated star images.
**PAIR-IT**

4620 ! Pair-it pairs stars from the catalog and stars from Proc. A to find
4630 ! the orientation of the vehicle.
4640 SUB Pair-it(SHORT Fovc(*),REAL Nfouc,Fovm(*),Nfoum,Bun(*),Av(*),Tol,Sigy,
F1,H,Fld(*),Kmax)
4650 OPTION BASE 1
4660 DIM Vn(3,3),An(3,3),Sb(4)
4670 DIM Xx(2),Dist(5),L(3),Cov(4,4),T(3,3)
4680 SHORT Cosm(5),Epsp(45)
4690 INTEGER Indx(42,2)
4700 RAD
4710 !
4720 MAT Sb=Bun ! Save original values of Euler parameters.
4730 Cosmax=COS(F1/180)
4740 Cosmin=COS((II+FI)/180)
4750 L1=1
4760 !
4770 Mm=Nfouc+Nfouc-1 ! Maximum sum of indices for measured stars.
4780 PRINT USING "/',K";"Table of Cos(Theta) for Measured Stars."
4790 PRINT USING Form2
4800 Form2: (MAG-2" Star Star Cos(Theta)"
4810 !
4820 FOR M=3 TO Mm ! Loop over all possible sums of indices for
4830 J1=(M-1)/2 ! measured stars.
4840 FOR J1=1 TO J1 ! Loop over all pairs whose indices sum to M.
4850 K=M-J
4860 IF K>Nfouc THEN GOTO Nextj
4870 Cosm(L1)=Fovm(J,3)*Fovm(K,3)*(Fovm(J,1)*Fovm(K,1)+Fovm(J,2)*Fovm(K,2)+
1)
4880 IF Cosm(L1)>Cosmax THEN GOTO Nextj
4890 PRINT USING Form1;J;K;Cosm(L1)
4900 Form1: IMAGE 2X,DDD,2XDDD,3XDDD,DDD,DDD,DDD
4910 Indx(L1,1)=J
4920 Indx(L1,2)=K
4930 L1=L1+1
4940 Nextj: NEXT J1
4950 NEXT M
4960 Imax=L1-1 ! Total number in list.
4970 !
4980 Mm=Nfouc+Nfouc-1 ! Maximum sum of indices for catalog stars.
4990 PRINT USING "/,K";"Begin Pairing Catalog Stars and Comparing To Measured Pairs."
5000 PRINT USING "/,K,X,DDD";"Number of stars from catalog:",Nfouc
5010 PRINT USING Form2
5020 !
5030 FOR M=3 TO Mm ! Loop over all possible sums of indices of
5040 J1=(M-1)/2 ! catalog stars.
FOR Jj=1 TO J1 ! Loop over all pairs whose indices sum to M.
K=M-J:
IF K>5 THEN GOTO Nextjj
Cost=0
FOR L=1 TO 3
Cost=Cost+Fovc(Jj,L)*Fovc(K,L) ! Compute dot product.
NEXT L
PRINT USING Form1;Jj;K;Cost
IF Cost>Costmax THEN GOTO Nextjj
IF Cost<Costmin THEN GOTO Nextjj
FOR Ii=1 TO Imax
IF ABS(Cost-Cosm(Ii))<Tol THEN Match ! Test for match.
Nextii:
NEXT Ii
Nextjj:
NEXT M
Failed: PRINT "*********** NO PAIR MATCH FOUND FOR THIS FOV ***********"
Kmax=0
SUBEXIT
Match: PRINT USING "/,K;" Catalog Pair Matched with Measured Pair < 
Kmax=2
Im1=IncX(Ii,1)
Im2=IncX(Ii,2)
Ic1=Jj
Ic2=K
IF Fovc(Im1,4)<Fovc(Im2,4) THEN Okm ! Test magnitude order.
Is=Im1
Im1=Im2 ! Switch magnitude.
Im2=Is
IF Fovc(Ic1,4)<Fovc(Ic2,4) THEN Okc ! Test magnitude order.
Ic1=Ic1
Ic1=Ic2 ! Switch magnitude.
Ic2=Is
Okc: PRINT USING "/K;" Measured pair:
PRINT JSING Form2
PRINT JSING Form1;Im1;Im2;Cosm(Ii)
! Compute separation of pair/2.
Rho12=SQRT([Fovc(Im1,1)-Fovc(Im2,1)]^2+[Fovc(Im1,2)-Fovc(Im2,2)]^2)/2
Eps=2*Sigma*(Fl*Rho12)
Xcent=(Fovc(Im1,1)+Fovc(Im2,1))/2 ! Compute average position.
Ycent=(Fovc(Im1,2)+Fovc(Im2,2))/2
Fld(1,1)=Fovc(Ic1,1)
Fld(1,2)=Fovc(Ic1,2)
Fld(1,3)=Fovc(Ic1,3)
Fld(1,4)=Fovc(Ic1,4)
Fld(1,5)=Fovc(Ic1,5)
Fld(2,1)=Fovc(Ic2,1) ! Fill array with data for paired stars.
CALL Least(Dn(*),Vn(*),Fld(*),Kmax,W,Cov(*),Converge)
IF Converge=0 THEN Nextil ! Try another pair--this one didn't work.

Search for confirming stars.

CALL Dircoste(Dn(4),Vn(*),0,T(*),T(*),T(*))
MAT Rnz = v(Fld)
Kmax = Kmax
MAT Fld = Ze
PRINT LSING "11/12.K"; "Test For Additional Stars" x(calc) y(calc) y(meas) Dx Dy
FOR J=1 TO Nfovum
  FOR Is=1 TO 3
    CALL Procqr(L(Is),An(*),Xx(1),Xx(2))
    NEXT Is
  FOR J=1 TO Nfovum
    R1 = ABS(Xx(1) - Fovm(J,1)) ! Deviation in x.
    IF R1 > Epsp(J) THEN Next_j
    R2 = ABS(Xx(2) - Fovm(J,2)) ! Deviation in y.
    IF R2 > Epsp(J) THEN Next_j
    ! Confirming star found.
    PRINT USING "6(MD.DDDD,XX)"; Xx(1)*Fl,Fovm(J,1)*Fl,Xx(2)*Fl,Fovm(J,2)*Fl, R1*Fl,Epsp(J)*Fl,R2*Fl,Epsp(J)*Fl
    Kmax = Kmax+1
  END FOR
  IF Kmax = Nfovum THEN No_more ! We've matched all measured stars.
  IF Kmax = 5 THEN No_more ! We have enough confirming stars.
  GOTO Next_star
END FOR
Next_j: NEXT J
Next_star: NEXT N
No_more: IF Kmax > 2 THEN SUBEXIT
PRINT USING "+/K"; ** No additional stars found ** Assume false match"
6130 Kmax=0
6140 PRINT USING "/k";'Replace new values of Euler parameters with old values and continue pairing.'
6150 MAT Bun=Sb  ! Replace new Euler parameters with old values.
6160 GOTO Nextii  ! Continue pairing--assume this orientation isn't correct.
6170 SUBEND
Perturb

This subroutine computes a time varying perturbation to Euler parameters. We have used simple sinusoidal variations added to the nominal values. Input data consist of the orbital frequency, time, the nominal values of the Euler parameters and the amplitude and frequency of the variations. The perturbed Euler parameters are returned to the calling program.
************ P E R T U R B ************

7810 SUB Perturb(SHCR T Bnom(*),E(*),N(*),REAL Omega, Dt; B(*))
7820 OPTION BASE 1
7830 I
7840 B(1)=Bnom(1)+E(1)*COS(N(1)*Omega*Dt)
7850 B(2)=Bnom(2)+E(2)*SIN(N(2)*Omega*Dt) ! Perturb the nominal Euler parameters.
7860 B(3)=Bnom(3)+E(3)*COS(N(3)*Omega*Dt)
7870 B(4)=Bnom(4)+E(4)*SIN(N(4)*Omega*Dt)
7880 !
7890 Mag=SQR(DOT(B,E)) ! Normalize the new Euler parameters.
7900 MAT B=3/(Mag)
7910 !
7920 SUBEND
Phoegn

This small subroutine uses the stellar collinearity equations to compute image coordinates. As input it needs the star direction cosines and the $3 \times 3$ rotation matrix. This routine returns the $x$ and $y$ coordinates normalized by lens focal length:

$$
\frac{X}{f} = \frac{L_1 C_{11} + L_2 C_{12} + L_3 C_{13}}{L_1 C_{31} + L_2 C_{32} + L_3 C_{33}}
$$

$$
\frac{Y}{f} = \frac{L_1 C_{21} + L_2 C_{22} + L_3 C_{23}}{L_1 C_{31} + L_2 C_{32} + L_3 C_{33}}
$$

where $C_{ij}$ are elements of the rotation matrix and $L_i$ are star direction cosines.
**PHOEQN**

8170 ! Computes x,y coordinates for a particular star.
8180 SUB Phoeqn(L(*),C(*),Xpho,Ypho)
8190 OPTION BASE 1
8200 DIM Pho(3)
8210 !
8220 MAT Pho=C*
     ! Rotate direction cosines into new frame.
8230 Xpho=Pho(1)/Phc(3)
8240 Ypho=Pho(2)/Phc(3)
8250 !
8260 SUBEND
Post-mult

The function of this subroutine is to post-multiply, by a rotation matrix, a set of 4 matrices which are the partial derivatives, with respect to Euler parameters, of a second rotation matrix. For example, for field of view A we need the partial derivatives of the AN rotation matrix with respect to $\beta_{BA}$. Subroutine Mat-av computes the partials of AV with respect to $\beta_{BA}$. Then, since $AN = AV \cdot VN$

$$\frac{\partial AN}{\partial \beta_{BA}} = \frac{\partial AV}{\partial \beta_{BA}} \cdot VN$$

where the partials indicate derivatives of matrix elements.
************POST_MULT************

11260 SUB Post_mult(C(*), Dc1(*), Dc2(*), Dc3(*), Dc4(*))
11270 OPTION BASE 1
11280 DIM T(3, 3)
11290
11300 MAT T = Dc1 * C
11310 MAT Dc1 = T
11320 MAT T = Dc2 * C
11330 MAT Dc2 = T
11340 MAT T = Dc3 * C
11350 MAT Dc3 = T
11360 MAT T = Dc4 * C
11370 MAT Dc4 = T
11380
11390 SUBEND

Post-multiply derivative matrices by rotation matrix.
Pre-mul

This subroutine is similar to Post-mul. In this case, we pre-
multiply partial derivative matrices by a rotation matrix. For example,
since \( AN = AV \cdot VN \),

\[
\frac{\partial AN}{\partial VN} = AV \cdot \frac{\partial VN}{\partial VN}
\]

where, it will be recalled, the partial derivatives are computed by

Dircosb.
**PRE_MULT**

11040 SUB Pre_mult(C(*), Dc1(*), Dc2(*), Dc3(*), Dc4(*))
11050 OPTION BASE 1
11060 DIM T(3,3)
11070 !
11080 MAT T=C*Dc1
11090 MAT Dc1=T
11100 MAT T=C*Dc2
11110 MAT Dc2=T
11120 MAT T=C*Dc3  ! Pre-multiply derivative matrices by a rotation matrix.
11130 MAT Dc3=T
11140 MAT T=C*Dc4
11150 MAT Dc4=T
11160 !
11170 SUBEND
This subroutine controls the various functions of Process B. As input we need the Euler parameters describing the vehicle frame orientation, $\beta_{VN}$, and those describing the interlock between camera frames A and B, $\beta_{BA}$, along with the variance to associate with $\beta_{BA}$ in the Kalman filter update. Usually, both sets of Euler parameters are updated by Process B before they are returned to the calling program. Process B also requires the coordinates of each measured star in FOV(A) and FOV(B) and returns the calculated coordinates for stars matched with measured stars.

The first step in this subroutine is to compute the rotation matrices $BA$ from $\beta_{BA}$, from which we calculate $AV$, and matrix $VN$ from $\beta_{VN}$. The unit vector for the boresight of FOV(A) is contained in the last row of matrix $AN = AV \cdot VN$ and is used by Access to retrieve a subcatalog of stars near this boresight. Pair-it is then called to match catalog and measured stars and to update $\beta_{VN}$. We then compute $VN$ again and calculate $BN = BA \cdot AV \cdot VN$. The boresight unit vector, the last row of $BN$, is used by Access to again obtain a sub-catalog. Pair-it once again matches measured and catalog stars and updates $\beta_{VN}$.

There are several possible paths for Process B. If either FOV contains fewer than three stars, we skip any attempt to match stars in the FOV (we need at least three stars to confirm an orientation). Should FOV(A) and FOV(B) each contain fewer than three stars, we declare a failure condition for Process B and return to the calling program. In this case, no attempt is made to update $\beta_{VN}$ or $\beta_{BA}$ (and
no Kalman filter update is needed); the integrated values of $\beta_{VN}$ and covariance matrix are used to start the analysis of the next Process A data set.

If only one FOV contains a sufficient number of stars, we call Least-1 and use up to 5 stars in that FOV to update $\beta_{VN}$ (Pair-it updates $\beta_{VN}$ using only 2 stars). Note that the interlock parameters, $\beta_{BA}$, are not updated; the same values are used on the subsequent data frame.

Usually there are a sufficient number of stars in both FOV(A) and FOV(B) (more than 2 in each) so we can correct both $\beta_{VN}$ and $\beta_{BA}$. This is done by subroutine Least-2.

If $\beta_{VN}$ has been updated, then we compute calculated image coordinates for all matched stars and return these to the calling program along with the $4 \times 4$ covariance matrix associated with $\beta_{VN}$.
PROC - B

* *

3020 SUB Proc_b(*,Evoc(*),Bba(*),Voc(*),Nfovma,Nfovmb,Ka,Kb,W,Sigxy,SHORT Xyma(*),Xymb(*),Xyca(*),Xycb(*),REAL CovG(*),Pba(*),Qba(*),Bbslsq(*))
3040 ! This subroutine is process B of Star Wars. This version uses Euler
3050 ! parameters and recovers interlock Euler parameters.
3060 OPTION BASE 1
3070 DIM Bore(3)
3080 DIM Vn:3,3),Av(3,3),Ba(3,3)
3090 DIM An:3,3),Bn(3,3),Bv(3,3),Tn(3,3)
3100 DIM F1da(5,5),F1db(5,5),Fovm(10,4)
3110 DIM Ka(4,4),Lta(4,4),Bbae(4),T3(4),T4(4)
3120 DIM Cov(4,4)
3130 SHORT Fovst(10E,4)
3140 COM Vg(3,3),INTEGER Table(529,2)
3150 RAD
3160 REDIM Cov3(8,8)
3170 ! Some constants.
3180 Fe=2.42536
3190 F1=F+Fe
3200 Tol=7.25E-6
3210 Radius=5.7*PI/180
3220 Sigma=1.1*PI/180
3230 ! Calculate interlock matrices BA and AV.
3240 CALL Dircost(Bta(*),Ba(*),0,Tn(*),Tn(*),Tn(*))
3250 CALL Mat_ass(Ba(*),Tn(*),Tn(*),Tn(*),Au(*),0,Tn(*),Tn(*),Tn(*))
3260 Ka=0
3270 Kb=0
3280 PRINT USING "/k;" ; Start for FOV(A)
3290 REDIM Fovm(Nfovma,4)
3300 IF Nfovma>3 THEN Fovm
3310 FOR I=1 TO Nfovma
3320 Fovm(I,1)=Xyma(I,3) ! Normalize image coord. by focal length.
3330 Fovm(I,1)=Xyma(I,1)/F1
3340 Fovm(I,2)=Xyma(I,2)/F1
3350 Fovm(I,3)=1/SQR(Fovm(I,1)^2+Fovm(I,2)^2+1)
3360 NEXT I
3370 CALL Dircost(Bun(*),Vn(*),0,Tn(*),Tn(*),Tn(*))
3380 MAT An=Av*v
3390 FOR I=1 TO 3
3400 Bore(I)=An(3,I) ! Boresight unit vector for FOV(A).
3410 NEXT I
3420 PRINT USING Form5;"Boresight direction cosines for FOV(A):",Bore(*)
3430 Form5: IMAGE /k/3(MD.DDDDD,X)
CALL Access(#2,NfovA,Bore(*),Sigma,Radius,Voc(*),Fovst(*))
CALL Pair_it(Fcvst(*),NfovA,Fovm(*),Nfovmb,Bvn(*),Av(*),To1,Sigxy,F1,W,Fld
a(*),Ka)
Ka2=Ka+Ka
BEEP ! Done with FOV(A).
!
PRINT USING "/#";" Start for FOV(B)"
IF Nfovmb<3 THEN Options
!
CALL Dircost(Bin(*),Vn(*),0,Tn(*),Tn(*),Tn(*),Tn(*))
CALL Dircost(Bta(*),Ba(*),0,Tn(*),Tn(*),Tn(*),Tn(*))
MAT Bn=Bn+Bv
MAT Bn=Bn+Bv
FOR I=1 TO "
Bore(I)=Br(3,1) ! Boresight unit vector for FOV(B),
NEXT I
PRINT USING Form5;"Boresight direction cosines for FOV(B):",Bore(*)
REDIM Fovm(Nfovmb,4)
!
FOR I=1 TO Nfovmb
Fovm(I,4)=Xmb(I,3) ! Normalize image coord. by focal length.
Fovm(I,1)=Xmb(I,1)/F1
Fovm(I,2)=Xmb(I,2)/F1
Fovm(I,3)=1/SQR(Fovm(I,1)^2+Fovm(I,2)^2+1)
NEXT I
!
CALL Access(#2,NfovB,Bore(*),Sigma,Radius,Voc(*),Fovst(*))
CALL Pair_it(Fcvst(*),NfovB,Fovm(*),Nfovmb,Bvn(*),Av(*),To1,Sigxy,F1,W,Fld
b(*),Kb)
Kb2=Kb+Kb
BEEP !
Options: !
IF (Ka<2) THEN Combine ! Do least-squares for two FOV.
IF (Ka<3) THEN Failed ! PUNT!!!
REDIM Co(8,4,4)
IF (Ka<2) THEN CALL Least_1(Bun(*),Av(*),Flda(*),Ka,W,Co(8,*),Converge)
IF (Ka<3) THEN CALL Least_1(Bun(*),Av(*),Fldb(*),Kb,W,Co(8,*),Converge)
GOTO Save_results
!
Combine:
!
MAT Bba=Bba ! Save estimated interlock parameters.
CALL Least_2(Bun(*),Bba(*),Ka,Flda(*),Kb,Fldb(*),W,Co(8 *))
MAT Bbalsq=Bba
! Perform Kalman filter update for interlock parameters.
FOR I=1 TO 4
FOR J=1 TO 4
Lba(I,J)=Ccu8(4+I,4+J)
NEXT J
NEXT I
3990 MAT Pba=Pa+Cba  ! Get covariance matrix at this time.
4010 MAT Ka=a+Fa
4020 MAT La=IHV(Ka)
4030 MAT Ka2=Pa+La  ! Kalman gain matrix.
4040 MAT T3=Ba+Tae
4050 MAT T4=Ka+Tc  ! Corrections to interlock parameters.
4060 MAT Bb=Bae+T4
4070 MAT La=Ca+Fa  ! Corrections to covariance matrix.
4080 MAT Pba=Pa-La
4090 MAT La
4100 Save_results:  !
4110 CALL Dir:csb(Bun(*),Vn(*),Tn(*),Tn(*),Tn(*))
4130 CALL Dir:csb(Bba(*),Ba(*),Tn(*),Tn(*),Tn(*))
4140 CALL Mat_av(xa(*),Tn(*),Tn(*),Tn(*),Tv(*),Tn(*),Tn(*))
4150 MAT An=Av+Vn
4160 MAT Xyca=ZER
4170 IF K=0 THEN Save_b
4180 !
4190 FOR K=1 TO Ka  ! Calculate image coord. for each
4200 FOR I=1 TO 3  ! matched star.
4210 Bore(*)=Fla(K,I)
4220 NEXT I
4230 CALL P:eqr(Bore(*),An(*),X,Y)
4240 Xyca(K,1)=X
4250 Xyca(K,2)=Y
4260 NEXT K
4270 !
4280 MAT Xyca=Xyca*(Fl)
4290 Save_b:  !
4300 MAT Xycb=Xyca*(Fl)
4310 IF Kb=0 THEN End
4320 !
4340 FOR K=1 TO Kb
4350 FOR I=1 TO 3
4360 Bore(*)=Flb(K,I)
4370 NEXT I
4380 CALL P:eqr(Bore(*),Bn(*),X,Y)
4390 Xycb(K,1)=X
4400 Xycb(K,2)=Y
4410 NEXT K
4420 MAT Xycb=Xycb*(Fl)
4430 !
4440 Form1: IMAGE K/4(MD.DDDDE,X)/
4450 Form2: IMAGE K,X,DD/10(2(MD.DDDD,X)/
4460 End:  !
4470 SUBEXIT
4480 !
4490 Failed: PRINT "********** PROCESS B FAILED **********"
4500 PRINT "Fewer than 3 stars in each FOV. No attempt to perform "
4510 PRINT "least-squares correction. Use old values for orientation."
4520 GOTO Save_results
4530 SUBEND
The subroutine integrates both the state differential equations and
the matrix Riccati equation, using Runge-Kutta methods. As discussed
in Appendix 8, we partition the Riccati equation into four parts and
only two of these need to be integrated numerically.

Both of the equations are integrated with two-cycle Runge-Kutta
methods. However, since the covariance matrix should be relatively
constant in steady-state, we use a step size, for the Riccati equation,
equal to the time between data frames (currently, 30 seconds). The step
size for the state integration is much smaller (currently, 0.5 sec or
60 steps between frames).

The first task in this subroutine is to partition the covariance
matrix and evaluate the right-hand-side of the Riccati equation at the
start of the time interval. We then integrate the state equation,
through repeated use of two-cycle Runge-Kutta methods, until we reach
the end of the interval. The right-hand-side of the Riccati equation is
again evaluated, this time at the end of the interval, and the integrated
covariance matrix calculated.
**RUNGE**

13600 SUB Runge(\(t_k\), \(\Delta t\), \(\text{Step}\), \(W_1(*), W_2(*), W_3(*), \text{REAL } X_k(*), P(*), Q(*), \text{Sigb}\)
13610 OPTION BASE 1
13620 DIM X(4), P11(4,4), P21(3,4), P11p(4,4), P21p(3,4), P22(3,3)
13630 DIM D1(4), L1(4,4), L2(4,4), Sumk(4,4), Suml(4,4)
13640 DIM S1(4), L1(4,4), L2(4,4), Sumk(4,4), Suml(4,4)
13650 DIM Q22(3,3), P22p(3,3)
13660 MAT Q22 = ID4
13670 MAT Q22 = Q22 + ((Sigb/30)^2)  \(! \text{ Factor of 30 may be changed to tune Kalman filter for bias recovery.}\)
13690 !
13700 DISP "Runge!"
13710 !
13720 \(T = t_k\)
13730 FOR I = 1 TO 4
13740 FOR J = 1 TO 4
13750 \(P11(I,J) = P(I,J)\)  \(! \text{ Get upper left 4x4 portion of cov. matrix.}\)
13760 NEXT J
13770 NEXT I
13780 !
13790 FOR I = 1 TO 3
13800 FOR J = 1 TO 4
13810 \(P21(I,J) = P(4+I,J)\)  \(! \text{ Get lower left 3x4 portion of cov. matrix.}\)
13820 NEXT J
13830 NEXT I
13840 !
13850 FOR I = 1 TO 3
13860 \(B(I) = X_k(I+1)\)  \(! \text{ Get current bias values.}\)
13870 FOR J = 1 TO 3
13880 \(P22(I,J) = P(4+I,4+J)\)  \(! \text{ Get lower right 3x3 portion of cov. matrix.}\)
13890 NEXT J
13900 NEXT I
13910 !
13920 REDIM \(X(4), Xk(4)\)
13930 MAT \(X = Xk\)
13940 MAT \(P11p = P11\)
13950 MAT \(P21p = P21\)
13960 MAT \(P22p = P22\)
13970 ! \(\text{Compute time derivative of covariance matrix at time } t(\text{initial}).\)
13980 W(1) = W(1)
13990 W(2) = W(2)  \(! \text{ Use gyro rates from beginning of interval.}\)
14000 W(3) = W(3)
14010 !
14020 CALL Mata(1, W(*), Xk(*), A11(*), A12(*))
14030 CALL Derrv(P11p(*), P21p(*), P22p(*), Q(*), A11(*), A12(*), K1(*), L1(*))
14040 !
14050 MAT Sumk = K1
14060 MAT Suml = L1
BEGIN state integration.

MAT H=B

Subtract bias values.

CALL Matt(3,W(*),X(*),A11(*),A12(*))

FOR It=2 TO Step/Delt+1 ! This is a series of 2 - step Runge-Kutta integrations.

MAT :1=A11*Xk

MAT S1=(Delt)*D1

MAT X=Xk+S1

W1=W1(1t)

W2=W2(1t)

W3=W3(1t)

Note: These are measured rates in gyro frame.

MAT H=0

Subtract the biases.

CALL Matt(0, W(*), X(*), A11(*), A12(*))

MAT W1=R1*X

MAT W2=D1

MAT X=(.5+Delt)*X

DISP X (*);

Tk=Ti+Delt

NEXT It:

Mag=SQR(DOT(Xk,Xk)) ! Normalize Euler parameters.

MAT Xk=Xk/Mag

! Compute time derivative of covariance matrix at time t(final).

MAT W1=W1(61)

MAT W2=W2(61)

MAT W3=W3(61)

CALL Matt(1, W(*), X(*), A11(*), A12(*))

CALL Deriv(F11p(*), P21p(*), P22p(*), Q(*), A11(*), A12(*), K1(*), L1(*))

! Compute updated covariance matrix.

MAT Sumk=Sjk+k1

! Compute upper-left 4x4 matrix.

MAT Sumk=Sjk*(Step/2)

MAT P11=P11+Sumk

MAT Suml=Sml+L1

! Compute lower-left 3x4 matrix.

MAT Suml=Sml*(Step/2)

MAT P21=P21+Suml

MAT P22p=P22*(Step)

MAT P22p=P22+P22p

! Fill the upper left 4x4 of the covariance matrix.

FOR I=1 TO 4

FOR J=1 TO 4

P(I,J)=P11(I,J)

NEXT J
14620 NEXT I
14630     ! Fill the lower left 3x4 part of covariance matrix
14640     ! and the upper 4x3 part with the transpose.
14650 FOR I=1 TO 3
14660     FOR J=1 TO 4
14670     P(4+I,J)=P21(I,J)
14680     P(J,4+I)=P21(I,J)
14690     NEXT J
14700 NEXT I
14710 FOR I=1 TO 3
14720     FOR J=1 TO 3
14730     P(4+I,4+J)=P22(I,J)
14740     NEXT J
14750 NEXT I
14760
14770 REDIM X(7)
14780
14790 DISP
14800 SUBEND
Secant

Subroutine Secant uses the secant method to update the partial derivative matrix used for least-squares correction (containing the derivatives of image coordinates with respect to the Euler parameters, \(\beta\)). If we let \(X = X(\beta)\) be the set of function (colinearity equations) which produce image coordinates for stars as a function of Euler parameters, then at the \(k\)th iteration we have \(\beta^k\), the coordinates \(X^k\), and the partial derivative matrix \(A^k = \partial X / \partial \beta |_k\) (determined by \(Dxdbeta\)). By least-squares we obtain corrections to \(\beta^k\) to get \(\beta^{k+1} = \beta^k + \Delta \beta^k\). These are used to compute new coordinates \(X^{k+1}\) so the changes are

\[
\delta X^k = X^{k+1} - X^k.
\]

However, the linearly predicted changes in \(X\) are

\[
\Delta X^k = A^k \Delta \beta^k.
\]

We proceed to modify the derivative matrix (by adding corrections) so that the linearly predicted changes will agree with the actual changes:

\[
\delta X^k = (A^k + C^k) \Delta \beta^k.
\]

No unique solution to this equation exists so we introduce

\[
\phi = \sum \sum (C_{ij}^k)^2 \text{ and minimize this criterion subject to }
\]

\[
\delta X^k - A^k \Delta \beta^k - C^k \Delta \beta^k = 0.
\]

Using the Lagrange multiplier technique we minimize

\[
\phi = \sum \sum (C_{ij}^k)^2 + \lambda'(\delta X^k - A^k \Delta \beta^k - C^k \Delta \beta^k).
\]

The necessary conditions require

\[
\partial \phi / \partial C_{ij}^k = 0
\]

and
\[ \frac{\partial \phi}{\partial \lambda_i} = 0 \]

or

\[ C_{ij}^k = \frac{1}{2} \lambda_i \Delta \beta_j \]

and

\[ \delta x^k - A^k \Delta \beta^k - C^k \Delta \beta^k = 0. \]

Matrix \( C^k \) can be expressed as the outer product of two vectors:

\[ C^k = \frac{1}{2} \lambda (\Delta \beta^k)^T. \]

We can now substitute for matrix \( C^k \) in the second necessary condition and solve this equation for the Lagrange multipliers:

\[ \lambda = 2(\delta x^k - A^k \Delta \beta^k)/(\Delta \beta^k)^T \Delta \beta^k \]

and substitute this for \( \lambda \) in the first necessary condition. Thus, the updated partial derivative matrix is

\[ A^{k+1} = A^k + (\delta x^k - A^k \Delta \beta^k)(\Delta \beta^k)^T/(\Delta \beta^k)^T \Delta \beta^k. \]

The secant method works best when we are near the solution vector, \( \Delta \beta \). As a check on the performance of this method, we compute the root-mean-square difference between calculated and measured coordinates at each iteration. If this parameter ever increases from its last value, we start over with exact derivatives (computed by \texttt{Dxdbeta}) before continuing with the least-square correction.
***************SECANT*******************************

7850  # Secant method of derivative update.
7860  SUB Secant( H(*), Delx(*), Ddy(*), Idim, Jdim)
7870  OPTION BASE 1
7880  DIM T1:1,Jdim), T2(Idim, 1), T3(Idim, Jdim), Sumat(Idim)
7890  !
7900  DISP "Secant"
7910 !
7920  Sumt2=JOT(Delx, Delx)  # Predicted changes in coordinates.
7930  MAT Sumat=3+Delx  # Difference between actual and predicted
7940  MAT Sumat=Ddy-Sumat  # changes.
7950  MAT Sumat=Sumat/(Sumt2)  # Changes.
7960  !
7970  FOR I=1 TO Idim  # Fill vectors for outer product.
7980  T2(I, 1)=Sumat(I)  # (see Appendix of Final Report.)
7990  NEXT I
8000  FOR J=1 TO Jdim
8010  T1(I, J)=Delx(J)
8020  NEXT J
8030  !
8040  MAT T3=T2*T1  # Compute corrections to derivatives.
8050  MAT A=A+T3  # Add corrections.
8060  !
8070  DISP
8080  SUBEND
Sort

This subroutine sorts an array and a column vector according to the values in the column vector. The order of sorting is from the largest to the smallest values and the array and vector must have the same number of rows. We use a simple "bubble" sort method - make repeated searches through the list, each time bringing the next largest value to the next available location in the list. In this version, in order to save computer time, we use a vector to save the re-ordering sequence of the column vector and use this sequence to re-order the array as the last step.
SORT

SUB Sort(A(*), SHORT B(*), REAL N, M)
OPTION BASE 1
DIM C(N), S(H, M)

DISP "Sort"

FOR K = 1 TO H
    C(K) = K
    NEXT K
N = N - 1

FOR K = 1 TO H
    FOR I = K TO H
        IF Test > A(I) THEN Continue
    NEXT I
End 1 = I

Test = A(1)

IF I = K THEN Continue2

AC(I) = A(I)
A(K) = Test

T = C(K)
C(K) = C(J)
C(J) = T

Continue2: NEXT K

FOR I = 1 TO H
    K = C(I)
    FOR L = 1 TO M
        S(I, L) = B(K, L)
    NEXT L
    NEXT I

MAT B = B
DISP
SUBEND
STAR PATTERN RECOGNITION AND SPACECRAFT ATTITUDE DETERMINATION. (U)

MAY 81  T E STRIKWERDA, J L JUNKINS

DAAK70-78-C-0038

UNCLASSIFIED

ETL-0260 NL

END

DATE 10-81

3 of 3
Sample output from data generator program (DATGEN):
Program setup, data for frame 2 and beginning of frame 3.

Do you want to use realistic gyro rate history (Y/N)? Y
Place disk with gyro rates (Filename: 'Wtrue') in :F8,1.....then push CONT.
Place star catalog disk (Filenames: 'Tab22' and 'Miss220') in :F8,1...
...then push CONT.
Has Table of star catalog cell positions been read-in?(Y/N) N
File name for simulation run ('Simnnn':F8,1'...where nnn is 3 num.): Sim042:F8,1
Period : 90.03 minutes
Do you want variations in Euler para. relating V frame to Gyro frame (Y/N)? Y
Do you want variations in Euler para. relating B frame to A frame (Y/N)? Y
Do you want time varying gyro biases (Y/N)? Y
Do you want noise added to image coordinates(Y/N)?Y
Do you want noise added to rate gyro data (Y/N)?Y
Position (km) and velocity (km/sec) of Satellite:
6.6526E+03 0.000E+00 0.000E+00
0.0000E+00 2.0334E+00 7.4768E+00
Matrix GN:
0.000000 .258319 .965926
0.965326 -0.258319
1.000000 0.000000 0.000000
Bvg...nominal Euler Parameters between frames V-G:
1.000000 0.000000 0.000000
Bvg...true Euler Parameters between frames V-G:
1.000000 0.000000 0.000000
Bvn...Initial Euler Parameters between frames V-N:
0.092295 .7013-7 .092297 .701068

******* FRAME : 2 *******
Matrix VG:
1.000000 0.000000 .000000
0.000000 1.000000 0.000000
.000000 .000000 1.000000
Satellite time from start of simulation: 30.00 seconds
Bvn...True Euler Parameters between frames V-N:
0.096640 .688586 .093727 .713248
Bba...Nominal Euler Parameters between frames B-A:
.707107 .707167 0.000000 0.000000
Bba...True Euler Parameters between frames B-A:
.707131 .707363 0.000097 0.000000
Matrix DN:
\[
\begin{bmatrix}
1.000000 & 0.000137 & -0.000137 \\
0.000137 & 0.000369 & 1.000000 \\
-0.000137 & 1.000000 & 0.000069
\end{bmatrix}
\]

Position (km) and velocity (km/sec) of Satellite:
- $6.6495E+03$  $6.8390E+01$  $2.2426E+02$
- $-2.7014E-01$  $2.0322E+00$  $7.4723E+00$

Position (km) and velocity (km/sec) of Earth:
- $-1.4960E+08$  $8.1343E+02$  $-3.5630E+02$
- $1.7790E-04$  $2.7314E+01$  $-1.1877E+01$

Total velocity of satellite (km/sec):
- $-2.6997E-01$  $2.9337E+01$  $-4.4043E+09$

Matrix AN:
\[
\begin{bmatrix}
.999384 & .008766 & .033971 \\
.825019 & .500383 & -.865442 \\
-.024602 & .865759 & .499855
\end{bmatrix}
\]

Boresight unit vector:
- $-.024602$  $.865759$  $.499855$

Polar angle: 63.01
Longitude angle: 91.63
Indices for four cells:
16 16 14 14
8 9 7 8

Number of stars from the catalog: 14.00
Number of stars in this FOV: 10.00
True image coordinates (mm):
- $5.128090$  $2.681500$  $4.503900$
- $4.462200$  $3.126360$  $4.509900$
- $2.508318$  $2.440960$  $4.810000$
- $-0.891715$  $0.633460$  $3.244000$
- $-1.584868$  $0.529351$  $2.990000$
- $5.824788$  $-2.588378$  $3.085000$
- $-0.543383$  $-3.457330$  $4.166000$
- $-0.958058$  $-4.177740$  $4.574000$
- $-4.615360$  $-3.573760$  $4.966000$
- $-4.840808$  $-1.151508$  $4.936000$

Measured image coordinates (mm):
- $5.125990$  $2.680390$  $4.523250$
- $4.458730$  $3.124300$  $4.560540$
- $2.581660$  $2.440350$  $4.862560$
- $0.896847$  $0.637305$  $3.259500$
- $-1.582270$  $0.524518$  $2.987110$
- $5.821640$  $-2.587440$  $3.061420$
- $-0.547052$  $-3.453350$  $4.174510$
- $-0.953593$  $-4.175710$  $4.633090$
- $-4.610930$  $-3.573599$  $4.875600$
- $-4.840808$  $-1.155158$  $4.929290$

Matrix AN:
\[
\begin{bmatrix}
.999391 & .008736 & .033784 \\
-.024464 & .865795 & .499801 \\
-.024884 & -.500323 & .865481
\end{bmatrix}
\]

Boresight unit vector:
- $-.024884$  $-.500323$  $.865481$

Polar angle: 33.06
Longitude angle: 267.15
Indices for four cells:
8
8
6
6
13
12
10
9

Number of stars from the catalog: 9.00
Number of stars in this FOV: 5.00
True image coordinates (mm):
.438158 -3.896320 2.313000
2.604860 -3.363190 4.900000
5.575050 -1.474120 4.835000
-3.296900 -2.229360 4.625000
-1.927940 2.696350 4.342000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000

Measured image coordinates (mm):
.437323 -3.897360 2.321970
2.603590 -3.362570 4.866520
5.579470 -1.471220 4.821620
-3.305680 -2.229210 4.858870
-1.928110 2.694110 4.858870
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000

Biases...true values:
-.000007 -.000010 -.000015
Frame: 2 Number of stars: 10 5

****** FRAME : 3 ******
Matrix VG:
1.000000 0.000000 -0.000029
-.000008 1.000000 -.000003
.000029 .000030 1.000000

Satellite time from start of simulation: 60.00 seconds
Bin...True Euler Parameters between frames V-N:
.908934 .676167 .095378 .725102
Bba...Nominal Euler Parameters between frames B-A:
.787107 .707107 0.000000 0.000000
Bba...True Euler Parameters between frames B-A:
.787139 .707107 0.000955 0.000000

Matrix BA :
1.000000 0.000140 -.000130
.000130 .000090 1.000000
.000140 .000360 .000000

Position (km) and velocity (km/sec) of Satellite:
6.6364E+03 1.2311E+02 4.8025E+02
-5.3996E-01 1.2311E+02 4.8025E+02

Position (km) and velocity (km/sec) of Earth:
-1.4968E+00 1.6396E+03 -.1260E+02
3.5581E-04 2.7314E+01 -1.1877E+01

Total velocity of satellite (km/sec):
-5.3996E-01 2.9313E+01 -4.4180E+00
Sample output from data analysis program (COMBIN):

Program setup and beginning of frame 2.

=================================== PROCESS B AND C =================================

Doing Proc B \((Y/Y)?: Y\)

Doing Proc C \((Y/Y)?: Y\)

Insert star catalog cisk into F8,1....Then press CONT

Has cell table been read-in? \(N\)

Input file name and device with simulation data: Sim042:F8,1

Satellite orbit major axis (km): 6653
Earth orbit major axis (km): 1.4960E+08
Satellite orbit inclination (deg.): 75
Rate gyro data sampling (sec): \(.50\)
Runge-Kutta time step (sec): 30.00
Gyro standard deviation (rad/sec): 4.84E-06

Input weight in arcseconds for interlock variance (2,5,etc.) 5.000

Input Gyro Bias Standard Deviation (Degrees/Hr) \(.50\)

\(Q\) Matrix:

\[
\begin{bmatrix}
2.350E-11 & 0.030E+00 & 0.000E+00 \\
0.000E+00 & 2.350E-11 & 0.000E+00 \\
0.000E+00 & 0.030E+00 & 2.350E-11 \\
\end{bmatrix}
\]

Do you want to offset matrix \(VG\) \((V-frame to Gyro frame)\) \(N\)

\(M\)atrix \(VG\) for this run:

\[
\begin{bmatrix}
1.000E+00 & 0.030E+00 & 0.000E+00 \\
0.000E+00 & 1.030E+00 & 0.000E+00 \\
0.000E+00 & 0.030E+00 & 1.000E+00 \\
\end{bmatrix}
\]

~~~~~~~~\ RECORD NUMBER: 2 ~~~~~~~~~~

Bvn....True Euler parameters between V and N frames:

\(
.099671 \quad .683716 \quad .093893 \quad .713186
\)

Bba....True Euler parameters between B and A frames:

\(
.707131 \quad .707131 \quad .000097 \quad .880000
\)

Bba....Current Euler parameters between B and A frames:

\(
.707131 \quad .707131 \quad .000097 \quad .880000
\)

Components of total velocity (km/sec):

\(-2.700E-01 \quad 2.932E+01 \quad -4.404E+00\)

Number of stars in each FOV:

FOV\((A)\): 10
FOV\((B)\): 5

Bvn....Integrated Euler parameters between V and N frames and gyro biases:

\[
\begin{bmatrix}
.089241 & .676321 & .095545 & .725192 \\
0.000E+00 & 0.000E+00 & 0.000E+00 & 0.000E+00 \\
\end{bmatrix}
\]

Components of total velocity (km/sec):

\(-2.700E-01 \quad 2.932E+01 \quad -4.404E+00\)
Sample output from data analysis program (COMBIN):
Analysis of data from frame 5.

~~~~~~~~ RECORD NUMBER:  5 ~~~~~~~~

Bun.... True Euler parameters between V and H frames:
.885633  .653-46  .098509  .748253

Bba.... True Euler parameters between B and A frames:
.707153  .787061  .000004  .000010

Bba.... Current Euler parameters between B and A frames:
.707146  .787067  .000099  -.000008

Components of total velocity (km/sec):
-1.077E+00  2.930E+01  -4.473E+00

Number of stars in each FOV:
FOV(A):  8
FOV(B):  6

Bun.... Integrated Euler parameters between V and H frames
and gyro biases:
.085706  .653776  .098562  .748299
-5.633E-06  4.482E-06  -1.359E-05

Components of total velocity (km/sec):
-1.077E+00  2.930E+01  -4.473E+00

Start for FOV(A)

Boresight direction cosines for FOV(A):
-.098483  .864305  .493232

Polar angle:  60.4 Degrees
Longitude angle:  96.5 Degrees

Cell indices:
16  16  14  14
9  8  8  7

Table of Cos(Theta) for Measured Stars:

<table>
<thead>
<tr>
<th>Star</th>
<th>Star</th>
<th>Cos(Theta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>.999822</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>.999760</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>.999785</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>.999822</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>.999725</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>.999094</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>.998723</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>.999755</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>.999041</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>.998603</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>.999735</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>.999596</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>.998308</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>.998021</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>.997476</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>.998697</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>.998996</td>
</tr>
</tbody>
</table>
Begin Pairing Catalog Stars and Comparing To Measured Pairs.

Number of stars from catalog: 15

<table>
<thead>
<tr>
<th>Star</th>
<th>Star</th>
<th>Cos(Theta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.939996</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.938015</td>
</tr>
</tbody>
</table>

>>>>> Catalog Pair Matched with Measured Pair <<<<<

Measured pair:

<table>
<thead>
<tr>
<th>Star</th>
<th>Star</th>
<th>Cos(Theta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2</td>
<td>0.938021</td>
</tr>
</tbody>
</table>

Least-Squares Correction For One FOV

RMS error for normalized image coordinates: 1.6748E-04

<table>
<thead>
<tr>
<th>Beta(old)</th>
<th>Beta(new)</th>
<th>Delta(Beta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0857057</td>
<td>0.0852774</td>
<td>-0.0004283</td>
</tr>
<tr>
<td>0.6503763</td>
<td>0.6569618</td>
<td>0.005595</td>
</tr>
<tr>
<td>0.0985616</td>
<td>0.0969390</td>
<td>-0.003773</td>
</tr>
<tr>
<td>0.7482986</td>
<td>0.7477891</td>
<td>-0.000595</td>
</tr>
</tbody>
</table>

RMS change in Euler parameters: 4.8172E-04

RMS error for normalized image coordinates: 2.3134E-05

<table>
<thead>
<tr>
<th>Beta(old)</th>
<th>Beta(new)</th>
<th>Delta(Beta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0852774</td>
<td>0.0852771</td>
<td>-0.000003</td>
</tr>
<tr>
<td>0.6569618</td>
<td>0.6569621</td>
<td>0.000003</td>
</tr>
<tr>
<td>0.0969390</td>
<td>0.0969394</td>
<td>0.000004</td>
</tr>
<tr>
<td>0.7477891</td>
<td>0.7477892</td>
<td>-0.000009</td>
</tr>
</tbody>
</table>

RMS change in Euler parameters: 5.4913E-07

Number of iterations: 2 Converge=1

Test For Additional Stars

<table>
<thead>
<tr>
<th>x(calc)</th>
<th>x(meas)</th>
<th>y(calc)</th>
<th>y(meas)</th>
<th>Dx</th>
<th>Dy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.358</td>
<td>0.357</td>
<td>-1.1708</td>
<td>-1.1696</td>
<td>0.0021</td>
<td>0.0012</td>
</tr>
<tr>
<td>0.6301</td>
<td>0.6335</td>
<td>-1.3228</td>
<td>-1.3184</td>
<td>0.0034</td>
<td>0.0044</td>
</tr>
<tr>
<td>-3.6080</td>
<td>-3.5608</td>
<td>1.0992</td>
<td>1.0890</td>
<td>0.0021</td>
<td>0.0012</td>
</tr>
<tr>
<td>3.7541</td>
<td>3.7560</td>
<td>5.8655</td>
<td>5.992</td>
<td>0.0019</td>
<td>0.0128</td>
</tr>
<tr>
<td>1.0280</td>
<td>1.0338</td>
<td>-3.7184</td>
<td>-3.7160</td>
<td>0.0058</td>
<td>0.0024</td>
</tr>
</tbody>
</table>

Start for FOV(B)

Bore sight direction cosines for FOV(B):

-0.096652  -0.501563  0.859646

Polar angle: 30.7 Degrees
Longitude angle: 259.1 Degrees

Cell indices:

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>6</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>13</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>
Table of \( \cos(\Theta) \) for Measured Stars.

<table>
<thead>
<tr>
<th>Star</th>
<th>Star</th>
<th>( \cos(\Theta) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>.936801</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>.936756</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>.931635</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>.937596</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>.932972</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>.937046</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>.933075</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>.932756</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>.936696</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>.931646</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>.937282</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>.936925</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>.937517</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>.938411</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>.935345</td>
</tr>
</tbody>
</table>

Begin Pairing Catalog Stars and Comparing To Measured Pairs.

Number of stars from catalog: 16

<table>
<thead>
<tr>
<th>Star</th>
<th>Star</th>
<th>( \cos(\Theta) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>.936608</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>.937515</td>
</tr>
</tbody>
</table>

>>> Catalog Pair Matched with Measured Pair <<<

Measured pair:

<table>
<thead>
<tr>
<th>Star</th>
<th>Star</th>
<th>( \cos(\Theta) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>4</td>
<td>.937517</td>
</tr>
</tbody>
</table>

Least-Squares Correction For One FOV

RMS error for normalized image coordinates: 1.2863E-03

<table>
<thead>
<tr>
<th>Beta(old)</th>
<th>Beta(new)</th>
<th>Delta(Beta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.0852771</td>
<td>.0853085</td>
<td>.0000315</td>
</tr>
<tr>
<td>.6508621</td>
<td>.6501618</td>
<td>-.000003</td>
</tr>
<tr>
<td>.0989394</td>
<td>.0988372</td>
<td>-.0001022</td>
</tr>
<tr>
<td>.7477882</td>
<td>.7464948</td>
<td>.0007066</td>
</tr>
</tbody>
</table>

RMS change in Euler parameters: 5.3647E-04

RMS error for normalized image coordinates: 9.5265E-06

<table>
<thead>
<tr>
<th>Beta(old)</th>
<th>Beta(new)</th>
<th>Delta(Beta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.0853085</td>
<td>.0853081</td>
<td>-.000004</td>
</tr>
<tr>
<td>.6501618</td>
<td>.6501616</td>
<td>-.000002</td>
</tr>
<tr>
<td>.0988372</td>
<td>.0988367</td>
<td>-.000004</td>
</tr>
<tr>
<td>.7464948</td>
<td>.7464943</td>
<td>-.000005</td>
</tr>
</tbody>
</table>

RMS change in Euler parameters: 3.9327E-07

Number of iterations: 2 Converge=1

Test For Additional Stars

<table>
<thead>
<tr>
<th>x(calc)</th>
<th>x(meas)</th>
<th>y(calc)</th>
<th>y(meas)</th>
<th>Dx</th>
<th>Dy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.8908</td>
<td>1.8912</td>
<td>-2.1949</td>
<td>-2.1940</td>
<td>.0003</td>
<td>.0009</td>
</tr>
<tr>
<td>-1.7222</td>
<td>-1.7192</td>
<td>2.4753</td>
<td>2.4632</td>
<td>.0031</td>
<td>.0121</td>
</tr>
<tr>
<td>3.6270</td>
<td>3.6267</td>
<td>2.6151</td>
<td>2.6141</td>
<td>.0003</td>
<td>.0009</td>
</tr>
<tr>
<td>-3.6754</td>
<td>-3.6810</td>
<td>-2.1516</td>
<td>-2.1678</td>
<td>.0056</td>
<td>.0162</td>
</tr>
<tr>
<td>-4.7927</td>
<td>-4.7999</td>
<td>4.4085</td>
<td>4.3959</td>
<td>.0071</td>
<td>.0126</td>
</tr>
</tbody>
</table>

Least-Squares Correction For Two FOV

Correct orientation using 5 stars from FOV(A) and 5 stars from FOV(B).
RMS error in normalized image coordinates: 5.7645E-04

Euler parameters and corrections:

<table>
<thead>
<tr>
<th>B(V-N)</th>
<th>delta-B(V-N)</th>
<th>B(B-A)</th>
<th>delta-B(B-A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.0856257</td>
<td>.0030175</td>
<td>.7071793</td>
<td>.0000330</td>
</tr>
<tr>
<td>.6504286</td>
<td>.0026642</td>
<td>.7070343</td>
<td>-.0000329</td>
</tr>
<tr>
<td>.0985211</td>
<td>-.0031564</td>
<td>-.0000228</td>
<td>-.0001221</td>
</tr>
<tr>
<td>.7482684</td>
<td>-.0022593</td>
<td>-.0001544</td>
<td>-.0001542</td>
</tr>
</tbody>
</table>

RMS change in Euler parameters: 2.1310E-04

RMS error in normalized image coordinates: 4.8614E-05

Euler parameters and corrections:

<table>
<thead>
<tr>
<th>B(V-N)</th>
<th>delta-B(V-N)</th>
<th>B(B-A)</th>
<th>delta-B(B-A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.0856257</td>
<td>.0030001</td>
<td>.7071792</td>
<td>-.0000002</td>
</tr>
<tr>
<td>.6504279</td>
<td>-.0030001</td>
<td>.7070344</td>
<td>-.0000002</td>
</tr>
<tr>
<td>.0985212</td>
<td>.0030001</td>
<td>-.0000227</td>
<td>-.0000000</td>
</tr>
<tr>
<td>.7482683</td>
<td>-.0030001</td>
<td>-.0001544</td>
<td>-.0000000</td>
</tr>
</tbody>
</table>

RMS change in Euler parameters: 1.0901E-07

RMS error in normalized image coordinates: 4.8614E-05

Euler parameters and corrections:

<table>
<thead>
<tr>
<th>B(V-N)</th>
<th>delta-B(V-N)</th>
<th>B(B-A)</th>
<th>delta-B(B-A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.0856257</td>
<td>.0030000</td>
<td>.7071792</td>
<td>-.0000000</td>
</tr>
<tr>
<td>.6504279</td>
<td>-.0030000</td>
<td>.7070344</td>
<td>-.0000000</td>
</tr>
<tr>
<td>.0985212</td>
<td>.0030000</td>
<td>-.0000227</td>
<td>-.0000000</td>
</tr>
<tr>
<td>.7482683</td>
<td>-.0030000</td>
<td>-.0001544</td>
<td>-.0000000</td>
</tr>
</tbody>
</table>

RMS change in Euler parameters: 1.6322E-09

Kalman Filter State Estimation

(End of least-squares for two FOV)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7.483E-01</td>
<td>7.483E-01</td>
<td>7.483E-01</td>
<td>7.483E-01</td>
<td>4.579E-05</td>
</tr>
<tr>
<td>-1.725E-05</td>
<td>-1.359E-05</td>
<td>-1.359E-05</td>
<td>-1.556E-05</td>
<td>3.656E-06</td>
</tr>
</tbody>
</table>

Norm of optimal estimate - 1: .0000000

(End of frame)