IN-17

14939

p70

NASA Technical Memorandum 102174

Simulation and Analyses of the Aeroassist Flight Experiment Attitude Update Method

J. R. Carpenter

June 1991

NASA

NASA Technical Memorandum 102174

Simulation and Analyses of the Aeroassist Flight Experiment Attitude Update Method

J. R. Carpenter
Lyndon B. Johnson Space Center
Houston, Texas

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

June 1991

CONTENTS

Section		Page
1.0	Summary	1
2.0	Background	2
2.1	AFE Navigation Accuracy Requirements	2
2.1.1	Mission Overview	2
2.1.2	Pre-Aeropass Mission Error Analysis	3
2.2	Overview of AFE IMU Alignment	4
2.3	SLM Performance Analyses	6
3.0	Verification Study	8
3.1	Method	8
3.1.1	Implementation Considerations	8
3.1.2	Changes to the SLM Algorithm	8
3.1.3	Environment Model	9
3.1.3.1	State Dynamics and Measurement Models	10
3.1.3.2	Initial Conditions	12
3.1.3.3	Attitude Timeline Model	12
3.2	Results	13
3.3	Discussion	22
4.0	Monte Carlo Analysis	23
4.1	Method	23
4.1.1	Initial Conditions	23
4.1.2	Simulation Processing	23
4.1.3	Post-Processing	24
4.2	Results	25
4.3	Discussion	26
4.3.1	State Vector Estimation	26
4.3.2	Estimation of Root Mean Square Errors	27
4.3.3	Comparison of Results to Modified Monte Carlo Analysis	28
5.0	Conclusions and Recommendations	29
6.0	References	30
Appendix A - The Program SLM_SIM		A-1
Appendix B - The Program MC_POST		B-1

TABLES

Table		Page
3-1	Input Loads for Baseline Case, SLM_BSLN	14
3-2	SLM_BSLN, Final IMU Alignment Errors (arsec)	16
3-3	Input Loads for RH_BSLN	17
3-4	RH_BSLN Using NCAD sim, Final IMU Alignment Errors (arsec)	19
3-5	RH_BSLN Using CSDL sim, Final IMU Alignment Errors (arsec)	19
3-6	Input Loads for Parametric Studies	20
4-1	95% Confidence Intervals for Mean and Standard Deviation of True Errors, n = 399	26
4-2	95% Confidence Intervals for Mean and Standard Deviation of Filter RMS Error Estimates, n = 399	26

FIGURES

Figure		Page
2-1	AFE Trajectory with Selected Events (orbital data for information only and subject to change)	3
2-2	Alignment of the AFE IMU in the Space Shuttle Payload Bay	5
2-3	Observability of Coordinate Frame Misalignment Through Body Rotation	6
3-1	Orbiter Measurement Attitudes and Maneuvers for SLM_BSLN (Star in -Z ST not shown)	15
3-2	Orbiter Measurement Attitudes and Maneuvers for RH_BSLN (Star in -Z ST not shown)	18
3-3	Comparison of ECRV Magnitude Parametric Analyses	21
3-4	Comparison of ECRV Time Constant Parametric Analyses	21

NOTATION

The following convention for vector and matrix notation is used in this report:

$\underline{v}$	a lower case, underlined, italic letter represents a vector. A capitalized subscript refers to the coordinate frame in which the vector is expressed; lower case subscripts indicate components of the vector, or descriptive information.
$\underline{v}^x$	a vector superscripted with an *x* indicates the "cross-product" matrix of the vector, $\underline{v}^x = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$
A	a capital italicized letter indicates a matrix.
${}_B M_A$	a matrix subscripted in this fashion indicates a transformation matrix from frame *A* to frame *B*.

1.0 SUMMARY

The Star Line Maneuver (SLM) is a technique for updating the alignment of a Space Shuttle payload's Inertial Measuring Unit (IMU) using measurements made by the orbiter's star trackers. The technique was developed by the Charles Stark Draper Laboratory (CSDL), and is baselined for the Aeroassist Flight Experiment (AFE), a spacecraft which is manifested for the first quarter of 1995. The SLM is similar to the Attitude Match Update performed for payloads using an Inertial Upper Stage (IUS). However, the SLM is more accurate since it uses measurements of the distant stars as an alignment reference, rather than a Space Shuttle IMU. While the SLM has never been used operationally, CSDL has performed various performance analyses of the technique, including linear covariance error analyses, parametric studies, and a modified Monte Carlo analysis. In this report, analyses of the SLM are presented which verify and augment the CSDL performance analyses.

To perform the analyses presented herein, a simplified environment model was developed using a commercially available interactive software package, *Matlab*. For its state dynamics and measurement models, the environment model used those designed into the SLM algorithm's extended Kalman filter. To model inputs from the AFE IMU resulting from Space Shuttle maneuvers, a second-order expansion about the identity matrix was used in small steps. Results from this simulation compared favorably with a test case provided by CSDL, and parametric analyses of two filter design parameters were confirmed.

This simulation was also used to perform Monte Carlo analysis of the SLM. This analysis was used to confirm that the SLM Kalman filter is an unbiased and consistent estimator over the range of expected environment states. The analysis also augmented the modified Monte Carlo analysis of CSDL, by providing better mean and 1σ performance estimates than are possible with the modified technique. In order to limit the number of runs required for the analysis, the method of confidence intervals was used. Three hundred ninety-nine cases were run, which resulted in a 95% confidence level in the results. These results showed that the SLM filter's state and error estimates were unbiased and consistent, and provided an enhanced look at the filter's mean and 1σ performance.

While the results presented in this report should enhance the AFE guidance and navigation community's confidence in the SLM technique, it should be recalled that the environment models used in this and all previous performance analyses were derived from the nominal models in the SLM algorithm. Additional analyses are recommended to characterize the robustness of the technique to a wider range of environment models. Furthermore, the architecture of the interfaces between the SLM processor and the Space Shuttle and AFE navigation systems has yet to be determined. The form taken by this architecture may affect SLM processing.

2.0 BACKGROUND

The SLM is a technique for aligning the IMU of a spacecraft in the payload bay of the Space Shuttle, using an extended Kalman filter to process data from the orbiter's star trackers. Because orbiter star tracker measurements are utilized to establish a reference frame based on the distant stars, the payload IMU accuracy achievable with this alignment technique is superior to methods which utilize the orbiter IMU as a reference frame.

A deterministic version of the SLM technique was first proposed by Kevin Daly of CSDL in 1984 [1], as an alternate attitude update method for the Boeing Aerospace Company's IUS. While never used by the IUS program, Y.C. Tao of CSDL proposed using the SLM technique to perform IMU alignment for the AFE in 1987 [2], when that program was confronting weight and cost difficulties. As a result, a heavy and costly star tracker, which served only to perform this alignment, could be removed from the program. This star tracker deletion was approved by Marshall Space Flight Center (MSFC) program managers, with the concurrence of the AFE Guidance and Navigation (G&N) Mode Team at the Johnson Space Center (JSC), and the SLM was baselined as the sole IMU alignment method for the mission. Subsequent SLM design and analysis activities have been performed by Roger Hain of CSDL, with oversight from the G&N Mode Team at JSC.

In subsequent paragraphs, the AFE navigation accuracy requirements which motivate the need for a high precision IMU alignment are discussed, along with a high-level description of the technique. In addition, an overview of performance analyses previously conducted is given, as well as a discussion of how the analyses presented in this report were motivated.

2.1 AFE NAVIGATION ACCURACY REQUIREMENTS

2.1.1 Mission Overview

The purpose of the AFE is to execute an aeroassisted maneuver from a 41 x 19,323 nautical mile (nmi) orbit into a 166 x 166 nmi orbit, as currently envisioned by mission planners. Such a maneuver would form part of the mission of a proposed Aeroassisted Orbital Transfer Vehicle, and would be similar to aerocapture maneuvers which are proposed for Space Exploration Initiative (SEI) missions to the Moon and Mars. The AFE's mission objectives include [3]

- Gathering atmospheric entry environmental data,
- Gathering knowledge of high altitude, high Mach number aerodynamic performance,
- Gaining experience in guidance and control algorithms effective for aerobraking trajectories, and
- Collecting data for experiments which assess aeroassist flight technology.

The AFE trajectory is depicted in figure 2-1. Approximately one revolution after aligning its IMU with the SLM, the vehicle will be deployed from the Space Shuttle into a 160 nmi circular orbit. The spacecraft will next perform a burn which transfers it onto the elliptical orbit using a Star 63 solid rocket motor (SRM). The spent SRM case will then be ejected. The subsequent aeropass will place the vehicle into a 184 x 30 nmi orbit. Three burns will follow to place the vehicle into a 166 nmi circular recovery orbit, and experimental data will be downlinked. The Space Shuttle will then rendezvous with the AFE, and recover it using the Remote Manipulator System. Upon return to the ground, the AFE's onboard experiments and thermal protection tiles will be inspected [4].

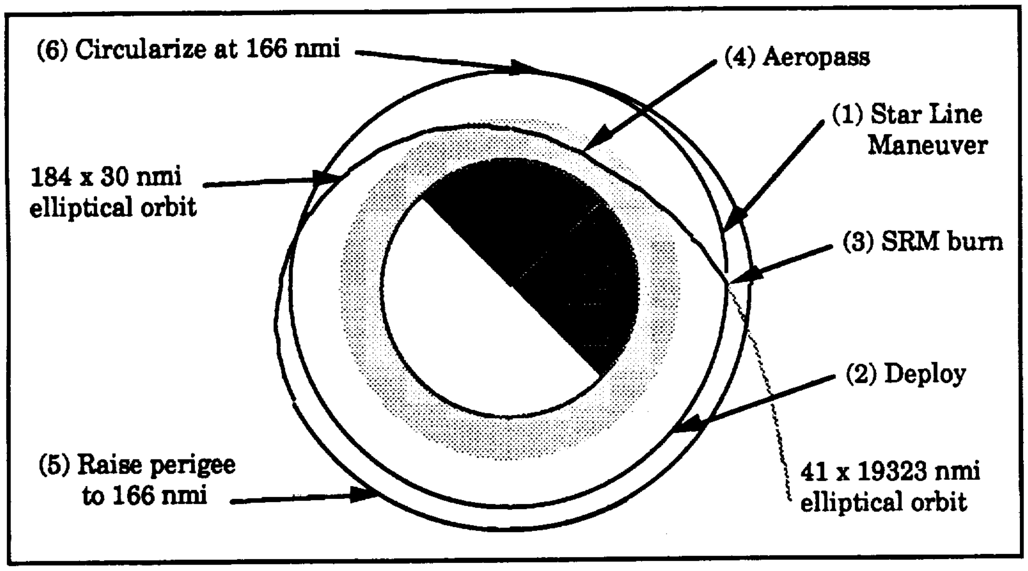

(6) Circularize at 166 nmi
(4) Aeropass
(1) Star Line Maneuver
184 x 30 nmi elliptical orbit
(3) SRM burn
(2) Deploy
(5) Raise perigee to 166 nmi
41 x 19323 nmi elliptical orbit

Figure 2-1. - AFE Trajectory with Selected Events (orbital data for information only and subject to change)

2.1.2 Pre-Aeropass Mission Error Analysis

In order to properly compute guidance commands during the aeropass, the AFE's unknown error in flight path angle at entry interface (EI) must not exceed 0.05 degrees, 3σ. This unknown error derives principally from five statistically independent navigation system initialization and measurement errors which propagate to EI as state vector uncertainties [5]:

(1) The uncertainty in the Space Shuttle navigation state transferred to the AFE during the AFE navigation system initialization.
(2) Unmodeled and unsensed translational accelerations from uncoupled jet firings, atmospheric drag, vents, etc. during the AFE pre-SRM burn coast.
(3) SRM burn acceleration measurement errors that result from AFE IMU accelerometer bias and scale factor uncertainties.
(4) Initial AFE IMU platform misalignment uncertainties that cause SRM burn thrust vector pointing errors.
(5) Initial uncompensated gyro bias drift rate uncertainties that cause the IMU platform to drift as a function of time from its ideal alignment; these also cause SRM burn thrust vector pointing errors.

Of interest to this report are the last two error sources, since their contributions to the flight path angle error at EI can be minimized with an accurate predeployment IMU alignment such as the SLM.

Linear covariance analysis performed by Frank Kreimendahl of CSDL [6] showed that a predeployment IMU fine alignment which provided root mean square (rms) errors of 50 arcsec per axis would result in a 3σ flight path angle at EI of 0.033 degrees. This IMU alignment accuracy specification included an assumption of a gyro drift bias rate of 0.01 degrees/hour.

Based on this study, a 50 arcsec/axis accuracy requirement was placed on the IMU update quaternion provided by the SLM. While this requirement allows nearly a 36 arcsec 3σ "fudge factor" at EI, it should be realized that linear covariance (lincov) analyses are often more optimistic than Monte Carlo methods. For example, the possibly nonlinear effect on EI state uncertainty of unmodeled and unsensed translational accelerations (error source (2) above) cannot be fully examined by lincov analysis. Such effects must be approximated in lincov studies, for example as velocity random walk errors.

2.2 OVERVIEW OF AFE IMU ALIGNMENT

An idealization of the problem of aligning the IMU of a payload in the Space Shuttle payload bay is depicted in figure 2-2. The problem is reduced to two dimensions to simplify the discussion. As shown, the orbiter carries a navigation base, assumed to be rigid, upon which are mounted three IMUs and two star trackers (STs). These IMUs maintain an estimate of the orientation of the mean-of-1950 (M50) inertial frame with respect to a shuttle body-fixed coordinate frame. Note that there exists an error associated with this estimate. At some distance from the navigation base, the AFE is rigidly mounted to the orbiter payload bay, but due to static and dynamic flexure in the intervening structure, an angular bias exists between the AFE body frame and the orbiter body frame. Here, the AFE body frame is considered to be nominally coincident with the shuttle body axes; the angular bias would be applied to any nominal transformation between the two body-fixed systems.

The simplest alignment scheme, known as a "coarse alignment," is simply to transfer the orbiter IMU attitude quaternion directly to the payload IMU. The resulting AFE body to M50 quaternion contains the structurally induced angular bias, which is thought to be about 1200 arcsec/axis. This quaternion also includes the estimation error associated with the orbiter IMUs, which is 82 arcsec/axis, 1σ, immediately after an orbiter IMU stellar update. This error derives largely from IMU gimbal angle resolver limitations. While a coarse alignment is not adequate for most payloads, it is used as an initialization for the fine alignment techniques discussed below.

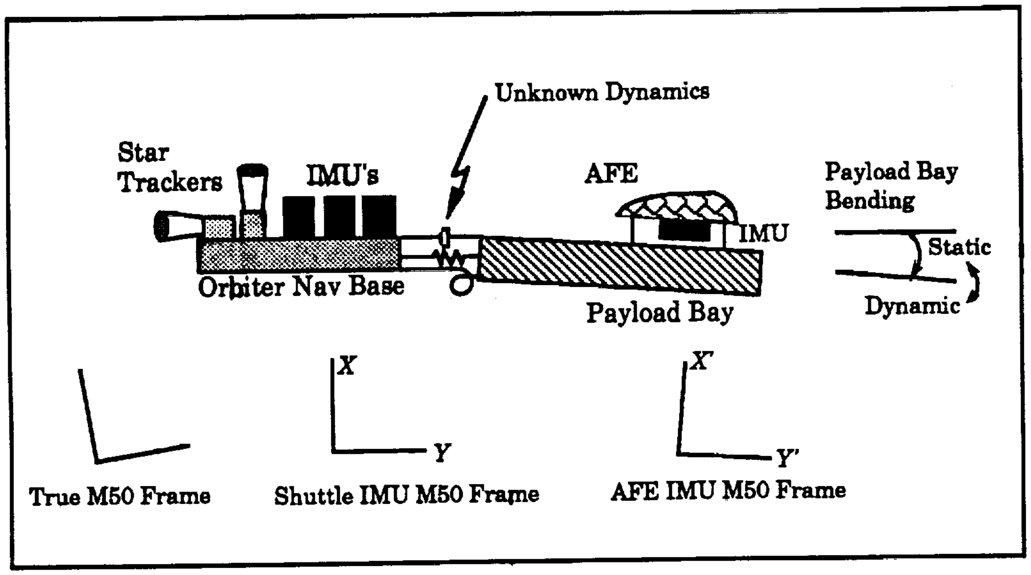


Figure 2-2. - Alignment of the AFE IMU in the Space Shuttle Payload Bay

Daly presented a method in [1] to remove the static component of this structural bias. This technique, called the "Body Axis Maneuver" by Daly, is presently used by the IUS under the name "Attitude Match Update" (AMU). In the AMU, the shuttle performs rotations which are sensed by the orbiter IMU and the payload IMU as occurring about differing axes. The differences are used to estimate the payload bay bending bias. A two-dimensional depiction of the AMU is shown in figure 2-3. The orbiter rotates 180° about the X axis (from an orbiter-fixed point-of-view, the M50 frame rotates, as shown in the figure). The payload IMU senses this maneuver as occurring about a vector $\underline{v}$ in the X'-Y' frame. If $\underline{u}_v$ and $\underline{u}_{x'}$ are the unit vectors in the directions of $\underline{v}$ and X', respectively, then the misalignment α between X-Y and X'-Y' is given by

$$\alpha = \cos^{-1}(\underline{u}_v \cdot \underline{u}_{x'}) \tag{2.1}$$

In three dimensions, two maneuvers about orthogonal axes are required, and the relations between the misalignments and the sensed maneuver directions are more involved. Also, 180° maneuvers are not required, but the accuracy of the method decreases rapidly as the maneuver sizes fall below 60°.

The accuracy achievable with the AMU technique is still limited by the orbiter IMU accuracy. For more precise alignment, an improved inertial reference is needed. The SLM uses the distant stars to provide such a reference, by means of the orbiter star trackers. Other than substituting a star-tracker-derived reference for an IMU-derived reference, the SLM is nominally identical to the AMU. However, the current SLM design also uses an extended Kalman filter to attempt to estimate errors induced from star tracker measurements and errors associated with the dynamic portion of the structural misalignment. The filter also estimates gyro drift, which allows its misalignment estimate to remain relatively "fresh" in a deployment slip scenario.

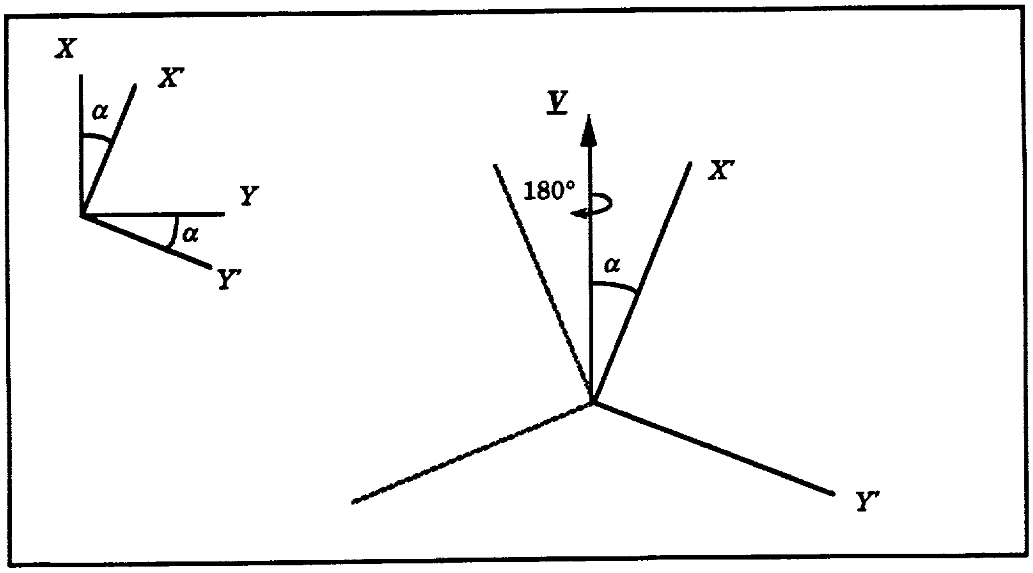


Figure 2-3. - Observability of Coordinate Frame Misalignment Through Body Rotation

2.3 SLM PERFORMANCE ANALYSES

Various performance analyses of the SLM have been accomplished for the AFE program by Tao and Hain. Mission trade studies, sensitivity analyses, and a modified Monte Carlo study have been accomplished.

Early work presented by Tao in [2] examined performance trades between mission options, such as maneuver size, star separation angle, and boresighting stars in the star tracker field-of-view. Sensitivities to IMU parameters, star tracker measurement noise, and payload bay dynamic bending were also presented. Tao also showed in [7] that using both orbiter star trackers could improve performance by 20%. In both studies, the magnitude of an exponentially correlated random variable (ECRV) used to model payload bay dynamic bending was shown to be nearly linear with respect to final IMU alignment accuracy. Tao quotes Treder, et al. [8] as giving the nominal rms value of this parameter to be 72 arcsec/axis, which results in a nominal worst axis alignment of 75 arcsec/axis for the one tracker case.

In later work by Hain [9], linear covariance, sensitivity, modified Monte Carlo, and deployment slip contingency analyses are presented. These studies indicated an improvement in final IMU alignment accuracy to 25 arcsec/axis, achieved through the use of Hain's extended Kalman filter and from lowering the ECRV magnitude to 24 arcsec, rms, per axis. This revised estimate of the dynamic bending magnitude was derived from an IUS AMU study [10]. The modified Monte Carlo method used is described by Anthony Bogner of CSDL in [11]. The purpose of this technique is to reduce the number of cases required in Monte Carlo analysis to produce accurate 3σ error statistics. Use of this approach allowed Hain to verify his linear covariance analysis with only 33 cases, a substantial savings in computer resources. As Bogner points out however, this method is less accurate than standard techniques for producing accurate mean and 1σ values.

An additional analysis of the sensitivity of the SLM dynamic bending modeling errors was performed by Hain [12]. The purpose of this study was to examine the errors induced by imperfect modeling of the true environment by the filter implemented in the SLM algorithm. In the study, the magnitude of the environment dynamic bending ECRV was parametrically varied, holding the filter value to 24 and 48 arcsec. It was found that the highest acceptable environment ECRV magnitude was 100 arcsec, for either filter value. This result echoed previous linear covariance analyses. However, environment dynamic bending models other than an ECRV were not examined.

While the performance analyses described above are comprehensive, several issues are still of concern. The primary issue is the model for the payload bay dynamic bending. The two known sources of ECRV magnitude data, [8] and [10], conflict by nearly 50 arcsec. While both are within the 100 arcsec limit indicated by linear covariance analysis, no studies of modeling errors have considered bending models other than an ECRV. However, ECRVs have historically proven to be reliable models for highly uncertain dynamics, as long as the ECRV parameters are chosen appropriately. Additional analysis in this area would nevertheless be desirable, but is not considered in this report.

A secondary concern is that the linear covariance performance analyses had not been verified with standard Monte Carlo techniques. Although the accurate 3σ data provided by the modified Monte Carlo method is more critical to mission success or failure, accurate mean and 1σ data complete the statistical picture of SLM performance, and support the results given by the newer method. Also, through the use of confidence intervals, an inordinately large number of cases need not be sampled to ensure a valid result. Standard Monte Carlo results are presented in section 4 of this report, along with a complete description of the confidence interval method of estimation.

A final concern relates to the AFE program's reliance on a sole source for navigation analysis. While verification studies are often perceived as duplication of effort, such analyses increase confidence in system performance. For this reason, an attempt was made to duplicate some of the parametric studies performed by Hain in [9]. These results are presented in section 3.

3.0 VERIFICATION STUDY

A simulation was developed which supplied inputs from the environment and from the Space Shuttle and AFE navigation computers to the SLM processor. The simulation was built using user-friendly, interactive software tools and simplified models whenever possible. In addition, a few assumptions were made concerning implementation details of the SLM algorithm, and minor changes were made to the design to improve simulation efficiency. A single-case comparison to a CSDL SLM simulation was performed, as well as verifications of parametric studies of payload bay dynamic bending model parameters. Close agreement was found for all comparison cases.

3.1 METHOD

3.1.1 Implementation Considerations

The simulation used by CSDL for performance analysis is a high fidelity, six degree-of-freedom Space Shuttle simulator. The major impediment to simulation of the SLM algorithm by the Navigation, Control, and Aeronautics Division (NCAD) has been that no high fidelity simulation tool was readily available to civil service analysts. As a result, the major issue of the simulation development was how to create an environment model which could furnish the required inputs to the SLM processors with adequate accuracy without becoming a major software development project in itself. This issue was resolved in the present simulation in a twofold manner: by implementing the simulation utilizing a user-friendly software language, and by reducing the complexity of the environment model itself.

The simulation was implemented using the syntax of *Matlab*, an interpretive, interactive, matrix-manipulation application. A listing of this simulation is included in Appendix A. *Matlab* provides matrix and vector operations, a large library of utility routines, flexible plotting capabilities, and is easy to debug due to its interpretive nature. In addition, *Matlab* scripts can be easily ported between *Macintosh* and *IBM*-compatible personal computers, and *Microvax* and *Sun* workstations. However, *Matlab* scripts run more slowly than equivalent compiled versions of the same algorithms.

3.1.2 Changes to the SLM Algorithm

The SLM algorithm was implemented as specified in [13]; however, as a result of choosing *Matlab* as the software environment, changes were made to the specified algorithm. Operations on rotation matrices were substituted whenever possible for equivalent quaternion manipulations, to take advantage of the compiled *Matlab* *C*-language matrix and vector operators. It was judged that manipulating nine elements of a matrix using a compiled routine was faster than operating on four elements of an equivalent quaternion using an interpreted, user-developed quaternion package. Also, simulation complexity was reduced by omitting the development of quaternion applications, and debugging was simplified by the use of matrix notation, which is generally more intuitive than that of quaternions. In addition, the simulation was designed to operate at 1 Hz, rather than the 25 Hz rate specified in [13]. This change significantly reduced execution time, and little change in results was noted when the time step was increased.

A further consideration of implementing the algorithm as given in [13] is that no control logic is provided which specifies the calling order of the major subprocedures. In the present simulation, it was decided to call the state propagation subprocess once per simulation step. This control logic was found necessary due to the large time step used. In contrast, [13] implies that in the flight software, state propagation will not be performed on the same cycle as measurement incorporation; presumably due to the high execution rate of the design, performing both functions on the same step will not be required.

3.1.3 Environment Model

The basic components of the environment portion of the simulation are a state dynamics model, a measurement model, and an attitude timeline model. Before the details of these models can be described, the following definitions are required:

$\{\mathbf{x}(t)\} \triangleq$ the random process, i.e. collection of functions of time, which describes the possible expected environment state vector time histories. The method of the Monte Carlo technique, discussed further in section 4, is to observe a sufficiently large number of the members of $\{\mathbf{x}(t)\}$.

$\mathbf{x}(t) \triangleq$ a member of the ensemble of functions $\{\mathbf{x}(t)\}$, determined from a given trial, i.e. a given observation of $\{\mathbf{x}(t)\}$. The initial value taken by $\mathbf{x}(t)$ is governed by a probability distribution function, described in the sequel. Subsequent values are determined by the environment model dynamics, described below.

$\hat{\mathbf{x}}(t) \triangleq$ the SLM Kalman filter's estimate of an observed $\mathbf{x}(t)$ at time t.

$\mathbf{e}(t) \triangleq$ the state error vector, at time t:

$$\mathbf{e}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t) \tag{3.1}$$

This quantity is also known as the true error vector, to distinguish it from the error estimates maintained by the state error covariance matrix.

$E(t) \triangleq$ the state error covariance matrix, at time t:

$$E(t) = E[\,\mathbf{e}(t)\ \mathbf{e}(t)^T\,] \tag{3.2}$$

where $E[\cdot]$ indicates the expected value. Note that the true value of $\mathbf{e}(t)$ is not maintained by the covariance, but only an estimate of its expected value, i.e. its mean.

3.1.3.1 State Dynamics and Measurement Models

The state dynamics and measurement models are the deterministic forms of the corresponding SLM filter models given in [13], viz.

$$\dot{\underline{x}}(t) = F(t)\,\underline{x}(t) + \underline{u}(t) \tag{3.3}$$

$$\dot{\underline{z}}(t) = \underline{h}(\underline{x}(t)) + \underline{v} \tag{3.4}$$

Here, $\underline{x}$ and $\underline{z}$ are the state and measurement vectors, F is the matrix of linearized state dynamics partials, $\underline{h}(x)$ is the non-linear measurement model, and $\underline{u}$ and $\underline{v}$ are the state and measurement noise vectors.

The state vector $\underline{x}$ consists of 5 misalignment vectors which correspond to the 15 states modeled in the dynamics of the SLM filter; these are the *x, y,* and *z* components of

the misalignments in the AFE body to M50 inertial transformation, $\underline{x}_{ial} = \begin{bmatrix} \alpha_{x\text{-}ial} \\ \alpha_{y\text{-}ial} \\ \alpha_{z\text{-}ial} \end{bmatrix}$

the static components of the misalignments in the star tracker 1 to AFE body transformation (payload bay static bending), $\underline{x}_{sb} = \begin{bmatrix} \alpha_{x\text{-}sb} \\ \alpha_{y\text{-}sb} \\ \alpha_{z\text{-}sb} \end{bmatrix}$

the precession of the AFE IMU axes with respect to the AFE body frame, due to gyro bias drift, $\underline{x}_{gd} = \begin{bmatrix} \omega_{x\text{-}gd} \\ \omega_{y\text{-}gd} \\ \omega_{z\text{-}gd} \end{bmatrix}$

the misalignments in the star tracker 1 to star tracker 2 transformation, $\underline{x}_{st} = \begin{bmatrix} \alpha_{x\text{-}st} \\ \alpha_{y\text{-}st} \\ \alpha_{z\text{-}st} \end{bmatrix}$

the dynamic components of the misalignments in the star tracker 1 to AFE body transformation (payload bay dynamic bending), $\underline{x}_{db} = \begin{bmatrix} \alpha_{x\text{-}db} \\ \alpha_{y\text{-}db} \\ \alpha_{z\text{-}db} \end{bmatrix}$

The dynamics of these state variables, assumed to be quasi-time-invariant and linear, are given by

$$F = \begin{bmatrix} 0 & & {}_I M_B(t) & & \\ & 0 & & 0 & \\ & & 0 & & \\ & 0 & & 0 & \\ & & & & F_{db} \end{bmatrix} \tag{3.5}$$

where ${}_I M_B(t)$ is the AFE body to inertial transformation,

$$F_{db} = \begin{bmatrix} 1/\tau_1 & & 0 \\ & 1/\tau_2 & \\ 0 & & 1/\tau_3 \end{bmatrix} \qquad (3.6)$$

and τ_i is the time constant associated with the *ith* element of the dynamic bending ECRV. The effect of ${}_I M_B(t)$ in the dynamics matrix is to map the gyro drift onto the inertial misalignment. As long as the orbiter maneuvers are performed slowly, the elements of this matrix will vary slowly in comparison to the SLM processor time step. Thus, F may be assumed to be quasi-time-invariant and linear.

The state noise is given by

$$\underline{u} = \begin{bmatrix} \underline{u}_{ial} \\ \underline{u}_{sb} \\ \underline{u}_{gd} \\ \underline{u}_{st} \\ \underline{u}_{db} \end{bmatrix} \qquad (3.7)$$

where $\underline{u}_{ial} = \underline{u}_{sb} = \underline{u}_{gd} = \underline{u}_{st} = [0\ 0\ 0]^T$, and $\underline{u}_{db}$ has the form of the noise portion of the ECRV, or

$$\underline{u}_{db} = \begin{bmatrix} \sigma_{db_1}\sqrt{1 - e^{-2\Delta t/\tau_1}} \\ \sigma_{db_2}\sqrt{1 - e^{-2\Delta t/\tau_2}} \\ \sigma_{db_3}\sqrt{1 - e^{-2\Delta t/\tau_3}} \end{bmatrix} \eta(0,1) \qquad (3.8)$$

where

$\eta(0,1)$	= a normal random variable, with zero mean and unit variance,
σ_{db_i}	= the magnitude, or rms value, of the *ith* element of the ECRV.

Although equation (3.4) is in vector form, the measurements are processed singly. These measurements consist of the components of star lines-of-sight. By linearizing matrix expansions of the misalignments $\underline{x}$ about the identity matrix, it is possible to reduce the non-linear measurement model to a linear form, as shown in [14]. The resulting scalar equation is

$$\delta q = \underline{b}_i^T \underline{x} \qquad (3.9)$$

where δq is the measurement residual and $\underline{b}_i^T$ is a row vector, dependent on which component of the line-of-sight is being considered, and which star tracker is being used. Note that $\underline{b}_i^T$ is also a function of the current estimate of the misalignments.

The measurement noise is applied via the scalar equation,

$$v = \sigma_{st}^2\, \eta(0,1) \qquad (3.10)$$

in which σ_{st}^2 is the variance associated with star tracker measurements.

3.1.3.2 Initial Conditions

Assuming the mean of $\{\boldsymbol{x}(0)\} = \boldsymbol{0}$, the filter's best initial estimate of the state vector is $\hat{\boldsymbol{x}}(t) = \boldsymbol{0}$, so that

$$E_0 = E[\,(\boldsymbol{x}(0) - \boldsymbol{0})\,(\boldsymbol{x}(0) - \boldsymbol{0})^T\,] = E[\,\boldsymbol{x}(0)\ \boldsymbol{x}(0)^T\,] \tag{3.11}$$

Therefore, a vector generated from the initial error covariance is equivalent to a realization of the initial value of the random function $\boldsymbol{x}(t)$, the function describing the environment state. Such a vector can be generated as follows.

First, the eigenvalues and eigenvectors of the initial covariance matrix, E_0, are found using

$$E_0\Xi = \Xi\Lambda \tag{3.12}$$

where

$$\Lambda = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_{15} \end{bmatrix} \tag{3.13}$$

$$\Xi = \begin{bmatrix} \xi_1\ \xi_2 \cdots \xi_{15} \end{bmatrix} \tag{3.14}$$

and the λ_i and ξ_i are the eigenvalues and eigenvectors of E_0, respectively. Now, the λ_i correspond to the uncorrelated mean square errors, which are equivalent to 1σ errors since the mean of $\{\boldsymbol{x}(0)\} = \boldsymbol{0}$. Also, the ξ_i indicate the directions of the correlations between these errors. Therefore, an initial 1σ state vector, $\boldsymbol{x}_0$, can be determined by

$$\boldsymbol{x}_0 = \Xi\, diag(\Lambda)^{1/2} \tag{3.15}$$

Here, $diag(\Lambda)$ indicates a vector containing the diagonal elements of Λ.

3.1.3.3 Attitude Timeline Model

The attitude timeline model is used to provide attitude information to the SLM algorithm which would be supplied by the orbiter's IMU at AFE power-up (for the coarse alignment), and by the AFE's IMU during the Star Line Maneuver itself. As such, this model takes the place of IMU models for both the orbiter and the AFE. This model uses a series of 2nd-order Taylor expansions of small angle rotations about the identity matrix to model each maneuver as follows:

$${}_{new}M_{old}(t) = I - \omega\, dt\, \boldsymbol{a}^{x} + \frac{1}{2}(\omega\, dt)^2\,[\,\boldsymbol{a}\,\boldsymbol{a}^T - I\,] \tag{3.16}$$

where

${}_{new}M_{old}(t)$	= approximate rotation matrix from attitude at time t to attitude at next time step $t + dt$,
$\underline{a}$	= axis about which maneuver is performed,
$\underline{a}^x$	= "cross-product" matrix of above,
ω	= maneuver rate, and
dt	= simulation time step.

Then, if ${}_IM_B(t)$ is the body-to-inertial rotation matrix at time t,

$$ {}_IM_B(t + dt) = {}_{new}M_{old}(t)\, {}_IM_B(t) \tag{3.17}$$

3.2 RESULTS

Several verification analyses were accomplished with the simulation. Three studies were performed: a baseline test (referred to as *SLM_BSLN*), comparisons to a run supplied by CSDL [15] (referred to as *RH_BSLN*), and verifications of parametric studies performed by CSDL.

The baseline case, *SLM_BSLN*, is a variation of the CSDL-supplied run, *RH_BSLN*. Both scenarios use both of the Space Shuttle's star trackers, and use three measurement periods, which are separated by two 90° rotations by the orbiter. In the baseline run, the orbiter has an initial attitude in which the *Z* body axis is aligned with the *Z* axis of the M50 frame, and the *X* and *Y* body axes are rotated by -10.6° from the *X* and *Y* inertial axes, respectively. After collecting one average measurement from each star tracker, the shuttle executes a 90° rotation about the *Z* body axis at a rate of 0.2 degrees/second. Another set of measurements is taken, which is followed by a 90° rotation about the minus *X* inertial axis at the same maneuver rate. The final data take period then occurs. The input loads for this case are given in table 3-1, and a schematic of its attitude timeline is shown in figure 3-1.

Table 3-1: Input Loads for Baseline Case, *SLM_BSLN*

Maneuvers:		
$_I M_{BSTS}(t=0) =$	$\begin{bmatrix} 0.9829 & 0.1840 & 0 \\ -.1840 & 0.9829 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Maneuver axes:		
1st maneuver	[0 0 1]	
2nd maneuver	[-.1840 0.9829 0]	
Maneuver sizes:		
1st maneuver	90°	
2nd maneuver	90°	
Maneuver rate:	0.2 degrees/second	
Measurements:		
No. of star trackers used:	2	
Star lines-of-sight:	*- Z* star tracker	*-Y* star tracker
1st data take period	[0 0 -1]	[0 -1 0]
2nd data take period	[0 0 -1]	[1 0 0]
3rd data take period	[0 -1 0]	[1 0 0]
Measurement variance:	225 arcsec^2	
Error Sources:		
$\sigma_{ial} =$	[82 82 82]	arcsec
$\sigma_{isb} =$	[1200 1200 1200]	arcsec
$\sigma_{db} =$	[24 24 24]	arcsec
$\sigma_{gd} =$	[0.01 0.01 0.01]	degrees/hour
$\sigma_{st1} =$	[60 60 60]	arcsec
$\sigma_{st2} =$	[60 60 60]	arcsec
$\tau_{db} =$	[400 400 400]	sec
Initial Environment State Vector:		
IMU misalignment	[1310.00 -982.20 2096.54]	arcsec
Payload bay static misalignment	[1498.24 -619.37 2018.74]	arcsec
Gyro bias drift rate	[0.01 0.01 0.00]	deg/hr
Star tracker misalignment	[-6.60 18.20 1.25]	arcsec
Payload bay dynamic misalignment	[23.06 15.50 -18.87]	arcsec
Other:		
$_{BSTS} M_{BAFE} = I$		
Correlation modeling: On		

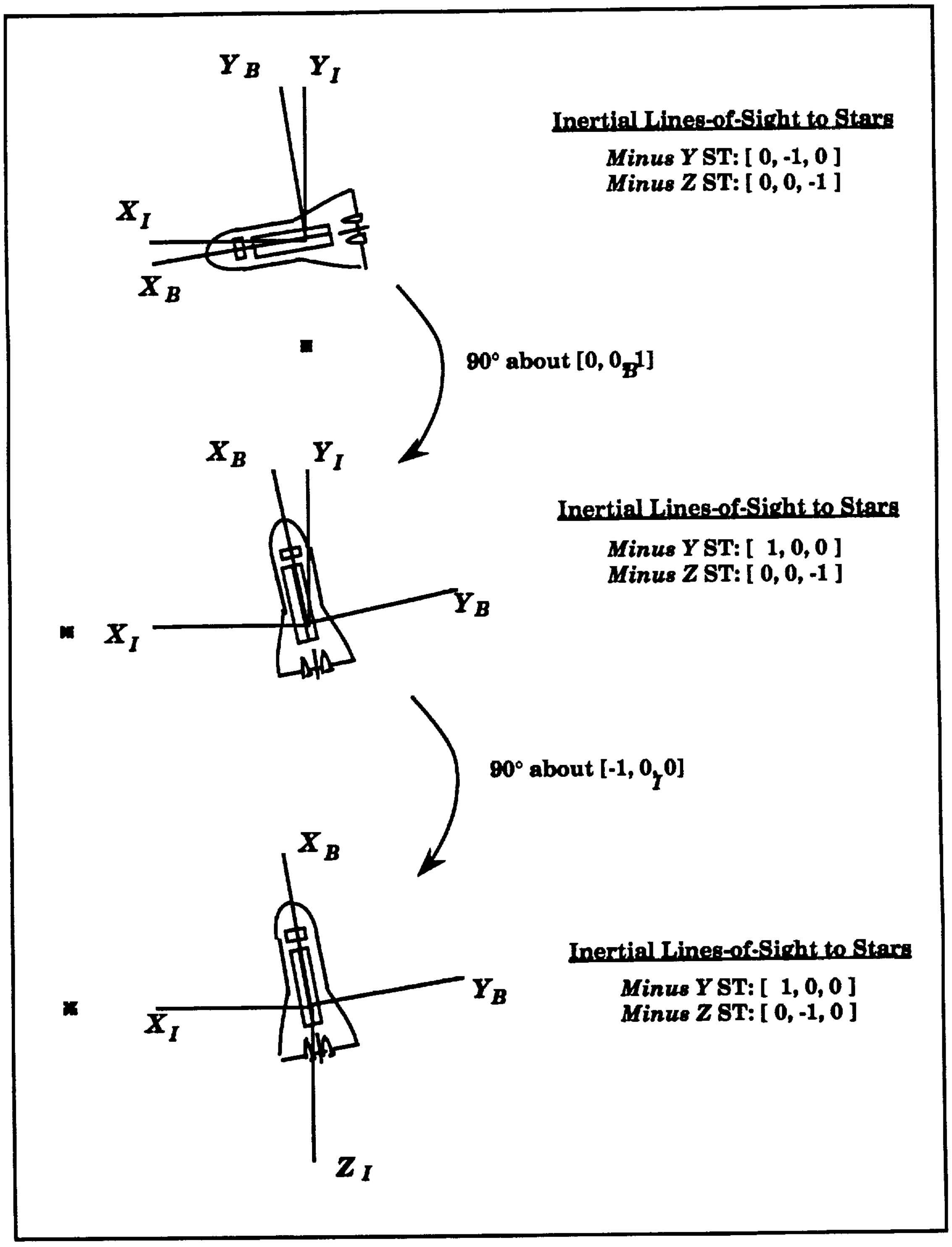

Y_B
Y_I
X_I
X_B
Inertial Lines-of-Sight to Stars
Minus Y ST: [0, -1, 0]
Minus Z ST: [0, 0, -1]
90° about [0, 0, 1]$_B$
X_B
Y_I
Y_B
X_I
Inertial Lines-of-Sight to Stars
Minus Y ST: [1, 0, 0]
Minus Z ST: [0, 0, -1]
90° about [-1, 0, 0]$_I$
X_B
Y_B
X_I
Z_I
Inertial Lines-of-Sight to Stars
Minus Y ST: [1, 0, 0]
Minus Z ST: [0, -1, 0]

Figure 3-1. - Orbiter Measurement Attitudes and Maneuvers for *SLM_BSLN*
(Star in -*Z* ST not shown)

The errors in final IMU alignment for *SLM_BSLN* are shown in table 3-2. Here, filter rms errors are found from the square root of the diagonal elements of the final filter covariance, and true errors are the differences between the filter's estimates and the environment's values for the final alignment states. The root sum squared, or rss, of the errors in each axis is also shown.

Table 3-2: *SLM_BSLN*, Final IMU Alignment Errors (arcsec)

	AXIS 1	AXIS 2	AXIS 3	RSS OF AXES
FILTER RMS (1σ) ERRORS	22.87	17.74	22.66	36.75
TRUE ERRORS	14.37	-9.89	-30.00	34.70

Next, *RH_BSLN,* the CSDL-supplied comparison case was examined. The major differences between this case and *SLM_BSLN* are in orbiter attitudes. The approximate initial attitude in this run is as follows: the shuttle's *Z* axis is aligned with the M50 minus *X* axis, and the *X* and *Y* body axes are rotated by -10.6° from the minus *Z* and minus *Y* inertial axes, respectively. The second data collection attitude is reached by a 90° maneuver about the minus *Z* body axis, which is followed by a 90° rotation about the *Z* inertial axis to reach the final attitude. The input loads for this case are given in table 3-3, and figure 3-2 depicts the sequence of attitudes. Note that a different set of star pairs is necessarily required for this maneuver sequence. The results for this case are shown in table 3-4. The results of running the same case using the CSDL simulation are shown in table 3-5.

Table 3-3: Input Loads for *RH_BSLN*

<u>Maneuvers:</u>

$$_{I}M_{BSTS}(t=0) = \begin{bmatrix} 0.0431 & -.0310 & -.9986 \\ 0.1654 & -.9855 & 0.0378 \\ -.9853 & -.1668 & -.0373 \end{bmatrix}$$

Maneuver axes:	
1st maneuver	[0.0431 -.0310 -0.9986]
2nd maneuver	[0.1654 -.9855 0.0378]
Maneuver sizes:	
1st maneuver	90°
2nd maneuver	90°
Maneuver rate:	0.2 degrees/second

<u>Measurements:</u>

No. of star trackers used: 2

Star lines-of-sight:

	- *Z* star tracker	-*Y* star tracker
1st data take period	[1 0 0]	[0 1 0]
2nd data take period	[1 0 0]	[0 0 1]
3rd data take period	[0 1 0]	[0 0 1]

Measurement variance: 225 arcsec^2

<u>Error Sources:</u>

σ_{ial} = [82 82 82] arcsec

σ_{isb} = [1200 1200 1200] arcsec

σ_{db} = [24 24 24] arcsec

σ_{gd} = [0.01 0.01 0.01] degrees/hour

σ_{st1} = [60 60 60] arcsec

σ_{st2} = [60 60 60] arcsec

τ_{db} = [400 400 400] sec

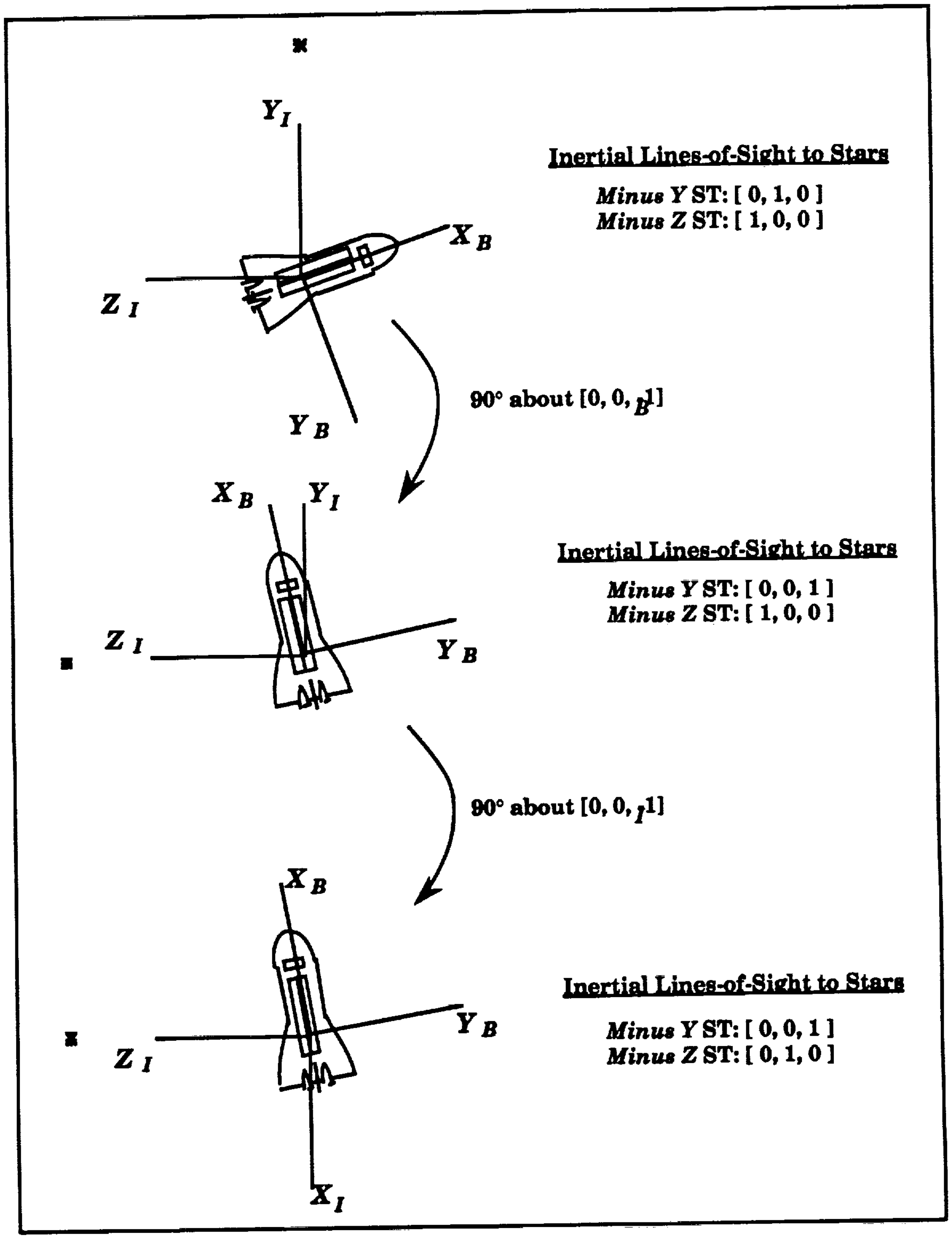
<u>Initial Environment State Vector:</u>

IMU misalignment	[243.87	10.32	1848.03]	arcsec
Payload bay static misalignment	[-1732.60	-302.73	-239.65]	arcsec
Gyro bias drift rate	[-0.01	-0.01	0.00]	deg/hr
Star tracker misalignment	[-155.13	78.90	-71.03]	arcsec
Payload bay dynamic misalignment	[27.37	32.06	-23.99]	arcsec

<u>Other:</u>

$_{BSTS}M_{BAFE} = I$

Correlation modeling: On



Y_I
Inertial Lines-of-Sight to Stars
Minus Y ST: [0, 1, 0]
Minus Z ST: [1, 0, 0]
X_B
Z_I
90° about [0, 0, $_B$1]
Y_B
X_B
Y_I
Inertial Lines-of-Sight to Stars
Minus Y ST: [0, 0, 1]
Minus Z ST: [1, 0, 0]
Z_I
Y_B
90° about [0, 0, $_I$1]
X_B
Inertial Lines-of-Sight to Stars
Y_B
Minus Y ST: [0, 0, 1]
Minus Z ST: [0, 1, 0]
Z_I
X_I

Figure 3-2. - Orbiter Measurement Attitudes and Maneuvers for *RH_BSLN*
(Star in *-Z* ST not shown)

Table 3-4: *RH_BSLN* Using NCAD sim, Final IMU Alignment Errors (arcsec)

	AXIS 1	AXIS 2	AXIS 3	RSS OF AXES
FILTER RMS (1σ) ERRORS	22.54	17.78	22.85	36.69
TRUE ERRORS	-2.89	-24.27	-18.51	46.89

Table 3-5: *RH_BSLN* Using CSDL sim, Final IMU Alignment Errors (arcsec)

	AXIS 1	AXIS 2	AXIS 3	RSS OF AXES
FILTER RMS (1σ) ERRORS	19.53	14.63	18.72	30.76
TRUE ERRORS	-11.28	3.55	-11.19	16.28

A comparison of the data in tables 3-4 and 3-5 indicates the degree of difference between the NCAD and CSDL simulations. It should be noted that small differences exist between the orbiter attitude timeline and initial environment state used in the generation of these results.

The final set of verification runs compares parametric studies of magnitudes and time constants for an ECRV used to model a portion of the filter dynamics (viz. equation (3.6)). The input loads for these studies are shown in table 3-6. In figure 3-3, the results using the CSDL high fidelity simulation for the ECRV magnitude study are compared to the results obtained using the NCAD simulation. Similarly, figure 3-4 presents a comparison of the ECRV time constant studies. In the diagrams, the rss of the filter rms final errors is represented on the vertical axis. Also, *ref* denotes data derived from plots in CSDL presentations, and *sim* represents results from the current simulation. It should be noted that an attitude sequence different from the previous cases was used in these comparisons. In particular, the magnitude of the second orbiter maneuver was 60°, rather than 90°.

Table 3-6: Input Loads for Parametric Studies

Maneuvers:

$$ {}_{I}M_{BSTS}(t = 0) = \begin{bmatrix} 0.0431 & -.0310 & -.9986 \\ 0.1654 & -.9855 & 0.0378 \\ -.9853 & -.1668 & -.0373 \end{bmatrix} $$

Maneuver axes:	
1st maneuver	[0.0431 -.0310 -0.9986]
2nd maneuver	[0.1654 -.9855 0.0378]
Maneuver sizes:	
1st maneuver	90°
2nd maneuver	60°
Maneuver rate:	0.2 degrees/second

Measurements:

No. of star trackers used:	2	
Star lines-of-sight:		
	- *Z* star tracker	-*Y* star tracker
1st data take period	[1 0 0]	[0 1 0]
2nd data take period	[1 0 0]	[0 0 1]
3rd data take period	[0.5000 0.8660 0]	[0 0 1]
Measurement variance:	225 arcsec^2	

Error Sources:

σ_{ial} = [82 82 82] arcsec

σ_{sb} = [1200 1200 1200] arcsec

σ_{db} = [* * *] arcsec

σ_{gd} = [0.01 0.01 0.01] degrees/hour

σ_{st1} = [60 60 60] arcsec

σ_{st2} = [60 60 60] arcsec

τ_{db} = [** ** **] sec

* Range: $6 \le \sigma_{db} \le 192$ arcsec

** Range: $4 \le \tau_{db} \le 4000$ sec

Initial Environment State Vector†:

IMU misalignment	[243.87	10.32	1848.03]	arcsec
Payload bay static misalignment	[-1732.60	-302.73	-239.65]	arcsec
Gyro bias drift rate	[-0.01	-0.01	0.00]	deg/hr
Star tracker misalignment	[-155.13	78.90	-71.03]	arcsec
Payload bay dynamic misalignment	[27.37	32.06	-23.99]	arcsec

† As a result of being derived from the initial covariance, the initial state varies when σ_{db} is parametrically varied. The state vector shown is for a nominal value of σ_{db}.

Other:

${}_{BSTS}M_{BAFE} = I$

Correlation modeling: On

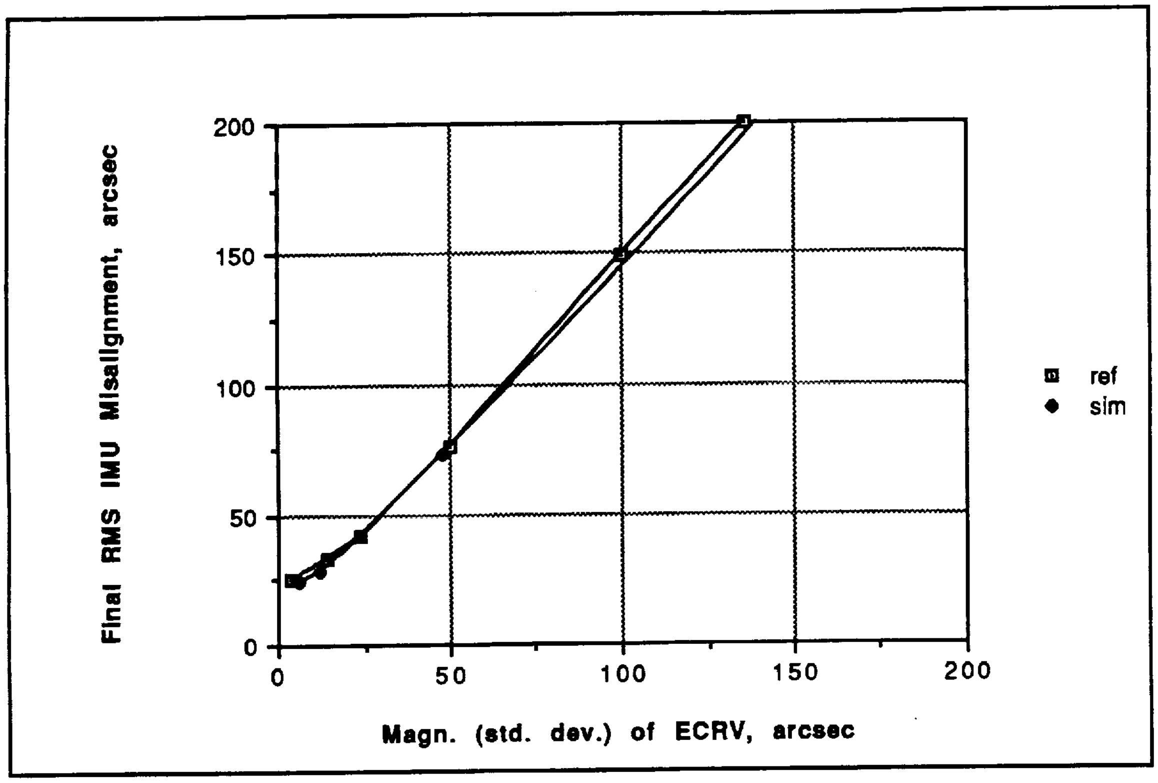


Figure 3-3.- Comparison of ECRV Magnitude Parametric Analyses

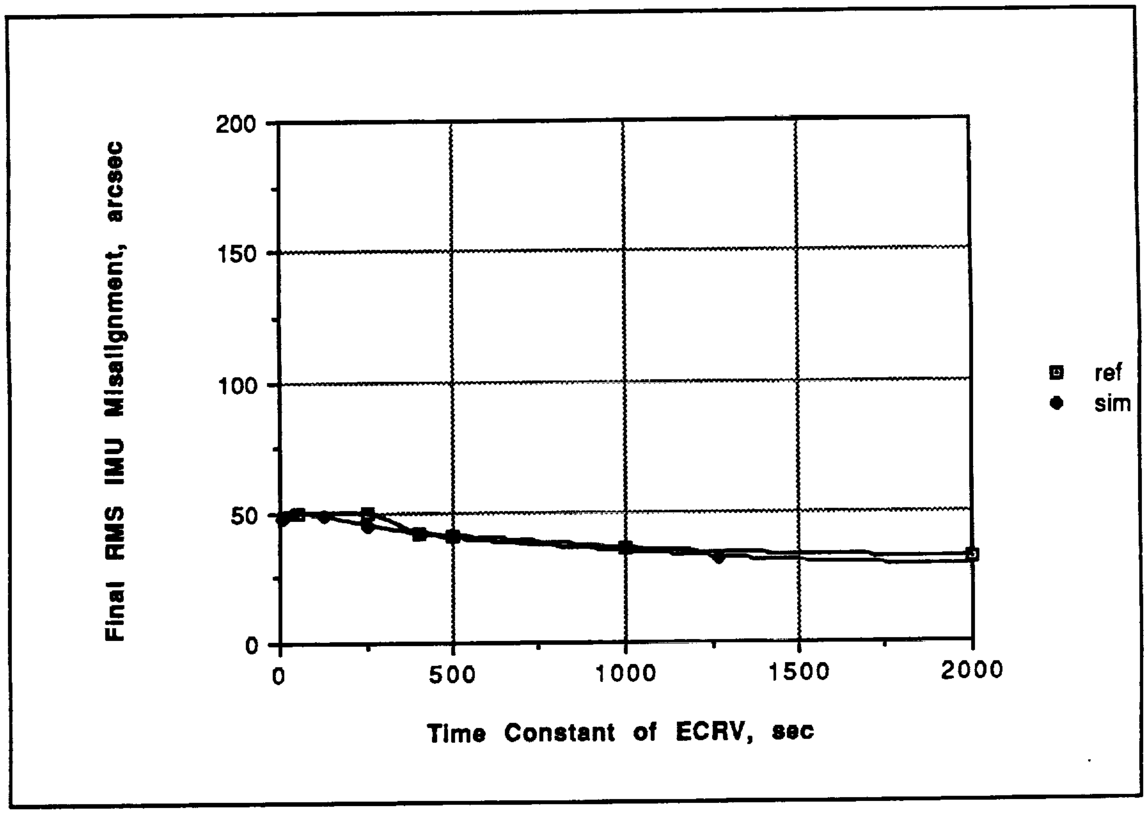


Figure 3-4.- Comparison of ECRV Time Constant Parametric Analyses

3.3 DISCUSSION

For the present simulation, good agreement is found when the final filter rms IMU alignment errors of *SLM_BSLN* (table 3-2) and *RH_BSLN* (table 3-4) are compared. Thus, the simulation's implementation of the filter is shown to be convergent and consistent for two different attitude and star pair sequences. When *RH_BSLN* run using the CSDL simulation (table 3-5) is considered, good comparisons in filter rms 1σ IMU alignment errors are also obtained. The small observed differences in filter rms errors could be expected, since the high fidelity CSDL simulation contains more accurate models and fewer approximations. The larger differences in true errors could be expected, since the initial environment state vector used in the CSDL simulation was different.

Furthermore, good agreement is found when the ECRV parametric studies performed using the NCAD simulation are compared to the CSDL results (figures 3-3 and 3-4). Again, small differences in the plots can be ascribed to the differences in the fidelity of the simulations.

While the cases examined above do not cover the range of analyses performed by CSDL, they include three different attitude sequences, 7 different star pairs, and 13 different runs, a reasonable subset for verification purposes. The close agreements found in these verification tests support the CSDL assertion that the SLM can be used to accurately align the AFE's IMU. The results also indicate that this technique is relatively insensitive to some implementation changes.

4.0 MONTE CARLO ANALYSIS

A statistical evaluation of the SLM algorithm was performed using the Monte Carlo technique along with the previously described simulation program. Three hundred ninety-nine initial error state vectors were generated using a normal distribution function, which had a zero mean, and the transformed eigenvalues of the initial filter covariance matrix as variances. These initial states were next used as inputs for the simulation program to generate a sample of the infinite population of SLM processor outputs. Estimates of the means and standard deviations of the final values of the true errors and filter rms values were then computed. Ninety-five percent confidence intervals were also generated for these statistics. Using this data, the following hypotheses were tested:

I. The filter is an unbiased and consistent estimator of the environment state vector.
II. The filter is an unbiased and consistent estimator of the root mean square (rms) errors associated with its state vector estimate.

Both hypotheses were found to be true, indicating that previous performance analysis [9] provided reliable indications of filter performance under all expected possible conditions.

4.1 METHOD

4.1.1 Initial Conditions

Random initial environment state vectors were derived from the filter covariance matrix. As shown in Section 3.1.3.2,

$$\boldsymbol{x}_0 = \Xi \, diag(\Lambda)^{1/2} \tag{4.1}$$

gives an initial 1σ state vector. Thus, a random initial state vector can be generated with

$$\boldsymbol{x}_0 = \Xi \, diag(\Lambda)^{1/2} \cdot \eta(0,1) + \mu_{\boldsymbol{x}_0} \tag{4.2}$$

where

$\eta(0,1)$	=	a random variable having a normal distribution with zero mean and unit variance, and
$\mu_{\boldsymbol{x}_0}$	=	the mean of $\boldsymbol{x}_0 = \boldsymbol{0}$

As long as no biases are introduced through propagation of initial vectors generated by this method, the resulting ensemble of random state trajectories will retain a zero mean.

4.1.2 Simulation Processing

The initial error state vector defined above is propagated and updated using simulated measurements in an identical fashion to the method previously described in section 3. The input loads used for these runs are identical to those given in table 3-5, except that nominal values were used for the ECRV parameters, and a random initial environment state vector was used. For Monte Carlo cases, the vector given in table 3-6 is the 1σ state vector discussed in section 4.1.1. As previously, the rms error estimates from the SLM filter's covariance matrix and the true errors, $\underline{\varepsilon}(t)$, were saved every 5 seconds during the simulation.

4.1.3 Post-Processing

A *Matlab* script, *mc_post*, was written to post-process the Monte Carlo data (a listing is included in Appendix B). This program reads from all Monte Carlo cases the final filter rms error estimates and the final true errors. It then calculates $\bar{x}_{\varepsilon}$ and s_{ε}, the sample mean and standard deviation of the true errors, and $\bar{x}_{rms}$ and s_{rms}, the sample mean and standard deviation of the filter rms error estimates. The following standard formulas [16] are used for these computations:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.3}$$

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{4.4}$$

In these formulas, n is the number of Monte Carlo cases.

Due to the finite nature of the sample size on which these estimates are based, some indication of the quality or reliability of these values is desired. One method for arriving at such an indication is to define an interval about the estimate within which the true value has a given probability of lying. Such an interval is called a confidence interval. This concept is described as follows by Allen [16]:

> The idea of a confidence interval is very similar to that of an error limit in numerical analysis. If we calculate a value x and know that the error in the calculation does not exceed δ (where $\delta > 0$), then we know the true value lies between $x - \delta$ and $x + \delta$. In the case of an estimator, we are dealing with a random variable, so we cannot predict with certainty that the true value θ of the parameter is within any finite interval. However, we can choose a high probability, such as 0.95 (95%), and then construct an interval, called a *confidence interval*, such that the probability that the true value of θ lies in the interval is 0.95. This is usually stated in the form of a $100(1 - \alpha)\%$ confidence interval where α, sometimes called the "level of significance," is the probability of error.

Allen gives several theorems which describe the computation of confidence intervals. Allen's formulas are given below. (N.B. : these formulas may also be found in a standard math handbook, such as [17]).

The 100(1 - α)% confidence interval for an estimate of a mean is given by

$$\bar{x} \pm \frac{z_{\alpha/2}\,\sigma}{\sqrt{n}} \tag{4.5}$$

where σ is the true standard deviation (assumed to be known here), and z_α is defined to be the largest value of z such that $Pr[Z>z] = \alpha$, where Z is a standard normal random variable. (For a 95% confidence interval $\alpha = 0.05$, and $z_{\alpha/2} = 1.96$.) If σ is unknown, the expression above should be modified:

$$\bar{x} \pm \frac{t_{\alpha/2}\,s}{\sqrt{n}} \tag{4.6}$$

where $t_{\alpha/2}$ is defined by $Pr[T>t_{\alpha/2}] = \alpha/2$, and T has a Student-t distribution with n-1 degrees of freedom. However, as $n \to \infty$, a Student-t distribution approaches a standard normal distribution. Therefore, s may be substituted for σ in equation (4.5) for $n \geq 30$ [16].

The 100(1 - α)% confidence interval for a variance estimate is given by

$$\frac{(n-1)s^2}{\chi^2_{\alpha/2}} \leq \sigma^2 \leq \frac{(n-1)s^2}{\chi^2_{1-\alpha/2}} \tag{4.7}$$

where χ^2_α is determined by a chi-squared distribution with n-1 degrees of freedom.

4.2 RESULTS

The sample mean and standard deviation of the true steady-state errors, $\bar{x}_\varepsilon$ and s_ε, and $\bar{x}_{rms}$ and s_{rms}, the sample mean and standard deviation of the filter steady-state rms error estimates, are given in tables 4-1 and 4-2, based on sampling 399 simulation cases. In addition, 95% confidence intervals are given for each parameter.

Table 4-1: 95% Confidence Intervals for Mean and Standard Deviation of True Errors, $n = 399$

	$\bar{x}_\varepsilon$, arcsec					s_ε, arcsec		
IMU	-1.62	±	3.06	29.20	≤	31.22	≤	33.55
Inertial	-0.79	±	2.04	19.51	≤	20.87	≤	22.42
Misalign	-1.55	±	2.43	23.15	≤	24.75	≤	26.60
PLB to	-1.28	±	2.38	22.66	≤	24.24	≤	26.04
ST1 Static	0.79	±	2.62	25.02	≤	26.76	≤	28.75
Misalign	-5.11	±	7.97	76.05	≤	81.32	≤	87.38
Gyro	0.0001	±	0.0010	0.0096	≤	0.0103	≤	0.0111
Drift	0.0001	±	0.0010	0.0094	≤	0.0100	≤	0.0108
Rate	0.0005	±	0.0009	0.0090	≤	0.0096	≤	0.0103
ST1 to	1.60	±	1.25	11.92	≤	12.74	≤	13.69
ST2	5.73	±	7.38	70.43	≤	75.31	≤	80.92
Misalign	5.63	±	7.57	72.24	≤	77.25	≤	83.01
PLB to	0.17	±	2.26	21.58	≤	23.07	≤	24.79
ST1 Dyn	0.80	±	2.08	19.86	≤	21.24	≤	22.82
Misalign	0.73	±	2.11	20.12	≤	21.51	≤	23.11

Table 4-2: 95% Confidence Intervals for Mean and Standard Deviation of Filter RMS Error Estimates, $n = 399$

	$\bar{x}_{rms}$, arcsec					s_{rms}, arcsec		
IMU	29.03	±	0.14	1.36	≤	1.46	≤	1.57
Inertial	19.37	±	0.10	0.91	≤	0.97	≤	1.04
Misalign	22.88	±	0.11	1.07	≤	1.15	≤	1.23
PLB to	24.26	±	0.12	1.14	≤	1.22	≤	1.31
ST1 Static	26.39	±	0.13	1.24	≤	1.32	≤	1.42
Misalign	83.22	±	0.41	3.90	≤	4.17	≤	4.49
Gyro	0.0100	±	0.0000	0.0005	≤	0.0005	≤	0.0005
Drift	0.0099	±	0.0000	0.0005	≤	0.0005	≤	0.0005
Rate	0.0098	±	0.0000	0.0005	≤	0.0005	≤	0.0005
ST1 to	12.60	±	0.06	0.59	≤	0.63	≤	0.68
ST2	76.91	±	0.38	3.61	≤	3.86	≤	4.14
Misalign	77.01	±	0.38	3.61	≤	3.86	≤	4.15
PLB to	23.38	±	0.12	1.10	≤	1.17	≤	1.26
ST1 Dyn	21.99	±	0.11	1.03	≤	1.10	≤	1.18
Misalign	22.12	±	0.11	1.04	≤	1.11	≤	1.19

4.3 DISCUSSION

4.3.1 State Vector Estimation

In the simulation used for this study, the environment model consisted simply of the deterministic forms of the state dynamics and measurement models used by the Kalman filter. Thus, if the noise terms in the environment were zeroed, the steady-state error vector ε_{ss} would be zero in a properly functioning filter. However, when the noise terms are included, the various steady-state error vectors observed in many trials of the simulation exhibit the characteristics of a random variable. If the mean of this random variable matches the mean of the environment noise models, then the filter is an *unbiased* estimator of the environment state vector. If the standard deviation of this random variable is sufficiently small, then the filter is a *consistent* estimator, since it can be expected to converge to the same estimates each time.

The major noise source present in the state dynamics models is the ECRV used to model payload bay dynamic bending. Because of correlations in the error covariance matrix, this noise term is included in the IMU inertial misalignment state and the payload bay static bending state. The magnitude of the noise portion of this ECRV was 24 arcsec/axis, 1σ, with a zero mean. Thus, the true mean and standard deviation of the steady-state errors, μ_{ε} and σ_{ε}, should be on the order of 0 and 24 arcsec, respectively, for the IMU inertial and payload bay misalignment portions of the error vector. The sampled values for these parameters, shown in table 4-1, closely match these predictions, within 95% confidence limits. One exception is the third component of payload bay static bending, where an unobservable star tracker misalignment forms the lower error limit.

The gyro drift rate and star tracker misalignment portions of the state dynamics are modeled as constants with random initial values. The distribution functions for these variables have a zero mean and standard deviations of 0.01 deg/sec/axis for gyro drift rate and 85 arcsec/axis for star tracker mounting misalignment (root sum squared of 60 arcsec/axis/star tracker). No measurements were made which could provide information about gyro drift rate, and no information can be gained in the direction of the star tracker boresight axes [9]. Hence, the uncertainty in the initial conditions of the gyro drift and two of the star tracker misalignment axes are unresolvable by the filter, and their initial random distributions will be reflected in the steady-state error statistics, as shown by table 4-1. The remaining component of star tracker misalignment is resolved to approximately 15 arcsec, the noise level associated with the measurement model.

Since all components of the mean of the true error vector were zero (to within the 95% confidence intervals), the filter was found to be an unbiased estimator of the simulated environment model. In addition, the filter was found to be consistent in its estimates, to within 95% confidence intervals surrounding the variances imposed by noise terms in the environment model. Hypothesis I is therefore confirmed.

4.3.2 Estimation of Root Mean Square Errors

A Kalman filter maintains its own estimate of the accuracy of its state vector estimate by means of a covariance matrix. This matrix contains estimates of mean square state errors along its main diagonal and correlation estimates in its off-diagonal terms. As stated previously, the square roots of the diagonal terms, or the root mean square errors, are equivalent to the standard deviations associated with the state estimate. Therefore, unbiased steady-state rms error estimates should correspond to the standard deviations associated with observations of the ensemble of true error state vectors. Furthermore, consistent rms error estimates should have a small standard deviation.

A comparison of the sample mean of the filter rms error estimates in table 4-2 with the true error standard deviations in table 4-1 shows close agreement. Only the three components of IMU Inertial Misalignment fall outside the 95% confidence intervals of s_{ε} by a few hundredths of an arcsecond. The filter can therefore be qualified as an unbiased estimator of the rms errors, verifying the first statement of Hypothesis II.

The sample standard deviations of the filter rms error estimates, also shown in table 4-2, are uniformly small in comparison to the sample means. In addition, these values are on the order of one-half the size of the confidence intervals surrounding s_{ε}. From this result, it can be concluded that the filter consistently estimated the rms errors; thus, Hypothesis II is verified.

4.3.3 Comparison of Results to Modified Monte Carlo Analysis

Finally, the mean and 1σ data in tables 4-1 and 4-2 augment the modified Monte Carlo performance analysis presented in [9]. In the modified Monte Carlo approach, initial error vectors are scattered uniformly within a 3.5σ error ellipse. In contrast, the standard Monte Carlo technique places 67% of the initial error vectors within a 1σ error ellipse. Thus, a larger sample of the 1σ region is examined with the standard technique so that, in general, its mean and 1σ estimates are more reliable than those of the modified method.

In [9], the modified Monte Carlo method was used only to verify that no final IMU alignment errors fell outside the 3σ range predicted by previous linear covariance analysis. It was judged that too few cases were run to produce accurate statistics. Such statistics are given by tables 4-1 and 4-2. The sample standard deviation of IMU inertial misalignment in table 4-1, averaged over the three axes, is 25.61 arcsec, which agrees well with the linear covariance analysis figure of about 25 arcsec. The data in table 4-2 can be analogized to the results of running 399 linear covariance analyses in order to see the effects of random residuals on the covariance updates. As these data show, such effects are quite small.

5.0 CONCLUSIONS AND RECOMMENDATIONS

Verification analyses in this report have shown that a simplified simulation of the SLM developed by the NCAD is convergent and consistent for two different attitude and star pair sequences. Also, good agreement was found with a simulation check case provided by the CSDL. In addition the NCAD simulation was used to verify parametric analyses [2], [7], and [9] performed by Draper of an ECRV used to model payload bay dynamic bending.

Monte Carlo analysis has shown that the SLM algorithm can accurately and reliably estimate a large sample of the possible expected inertial misalignment biases present in attitude information transferred from Space Shuttle IMUs to the AFE IMU. The SLM algorithm has also been shown to accurately and reliably estimate root mean square errors associated with its misalignment estimates. These conclusions support and verify previous linear covariance error analysis and modified Monte Carlo analysis [9].

The reader should note that in the all of these investigations, the SLM Kalman filter was only required to estimate biases which were of the genre expected by its designers. This shortcoming was a result of using environment dynamics and measurement models which closely resembled the filter's models. Most components of the true environment are well understood, and these are included in the filter's models when judged significant by the designer. However, the payload bay dynamic bending component, which results from day/night thermal cycling of the orbiter's longerons, is a poorly understood phenomenon which has never been directly measured. As the sensitivity analysis in section 2 and [2], [7], and [9] showed, even a correct model is quite sensitive to parameter variations. In [13], this modeling issue has been only partially addressed, since only parametric variations within the nominal model were investigated. No firm requirement for more extensive study of this issue exists, since there is good confidence among the members of the AFE GN&C community that the ECRV model of dynamic bending is adequate. Nevertheless, as the AFE GN&C system design schedule permits, analyses to indicate the filter's robustness to more diverse modeling errors would further enhance confidence in the SLM filter design.

Another topic not addressed in this report are problems associated with properly integrating the SLM algorithm with the star tracker data provided by the Space Shuttle GN&C computers. The relatively inflexible architecture of the orbiter's onboard software and hardware interfaces makes the properly time-tagged data difficult to acquire at the required rates. This problem is augmented if telemetry is used to transfer the data between the orbiter and the AFE, as is currently envisioned for SLM ground processing. Individuals from the operations, software design, and GN&C design communities at CSDL, JSC, and MSFC are currently pursuing these issues.

6.0 REFERENCES

1. Daly, Kevin C.: Alternate Attitude Update Techniques for the Inertial Upper Stage. Charles Stark Draper Laboratory Report No. CSDL-R-1723, September, 1984.

2. Tao, Y.C.: Alignment of AFE IMU in Shuttle Payload Bay using the Shuttle Star Tracker. Charles Stark Draper Laboratory Presentation, August 11, 1987.

3. Ruiz, Michael: Aeroassist Flight Experiment Mission Overview. NASA - Johnson Space Center Presentation, February 28, 1990.

4. Strahan, Alan: Personal Communication. NASA - Johnson Space Center, January 8, 1991.

5. McHenry, Leroy: Personal Communication. NASA - Johnson Space Center, January 2, 1991.

6. Kreimendahl, Frank A.: AFE Mission Covariance Analysis Results. Charles Stark Draper Laboratory Memorandum No. EGB-89-155, July 28, 1989.

7. Tao, Y.C.: AFE-IMU Alignment Process: Comparison of 1 Tracker/2 Tracker Performance. Charles Stark Draper Laboratory Presentation, June, 1988.

8. Treder, A., Norris, R., and Ruprecht, R.: Inflight Alignment of Payload Inertial Reference from Shuttle Navigation System. Advances in the Astronautical Sciences, Volume 57, 1985.

9. Hain, Roger: Star Line Maneuver Performance Analysis. Charles Stark Draper Laboratory Memorandum No. EGB-90-143, May 7, 1990.

10. Meldahl, K. L., Norris, R. E., and Clarke, J. P.: Flight Data Evaluation of a Technique for Payload Inertial Reference Alignment Transfer from the Shuttle Navigation System. Proceedings of the National Technical Meeting, Institute of Navigation, Anaheim, California, January 20-23, 1987.

11. Bogner, Anthony: An Alternate Approach to Monte Carlo Analysis. Charles Stark Draper Laboratory Memorandum No. OTVJ-89-045, September 11, 1989.

12. Hain, Roger.: SLM Dynamic Bending Mismodeling Sensitivity. Charles Stark Draper Laboratory Memorandum, June 14, 1990.

13. Spratlin, K., Fuhry, D., and Hain, R.: AFE G&N Detailed Requirements Document, Volume I, Section 4.14, Charles Stark Draper Laboratory Report No. CSDL-R-2134, May 15, 1989.

14. Hain, Roger.: Derivation of Major Elements of the SLM Algorithm. Charles Stark Draper Laboratory Memorandum No. EGB-90-144, March 15, 1990.

15. Hain, Roger.: SLM Test Run. Charles Stark Draper Laboratory Memorandum, June 23, 1989.

16. Allen, Arnold O.: Probability, Statistics, and Queueing Theory. Academic Press, Inc. Orlando, Florida, 1978.

17. Beyer, William H., Editor: CRC Standard Mathematical Tables, 26th Edition. CRC Press, Boca Raton, Florida, 1984.

APPENDIX A - THE PROGRAM *SLM SIM*

PROGRAM FLOW OVERVIEW

In order to give an overview of the simulation structure, the functional relationships of some of its modules are shown below. Each block corresponds to a *Matlab* script or function which is described in the sequel. Inner blocks are called by outer blocks. The page number on which a listing of these modules begins is given in parentheses after the module title.

SLM SIM (A-4)

- Declare global variables

    SIM EXEC (A-6)

    - Read i-loads (standard deviations of error sources, constant transformation matrices, etc.)
    - Coarse align:

    $${}_{I}M_{BAFE} = {}_{I}M_{BSTS}\ {}_{BSTS}M_{BAFE}$$

        SLM INIT (A-12)

        - Initialize filter dynamics and covariance
        - Incorporate filter state vector into filter transformation matrices, e.g.

        $${}_{I}M_{IMU} = I - [\mathbf{x}_{ial}]^{\times}$$

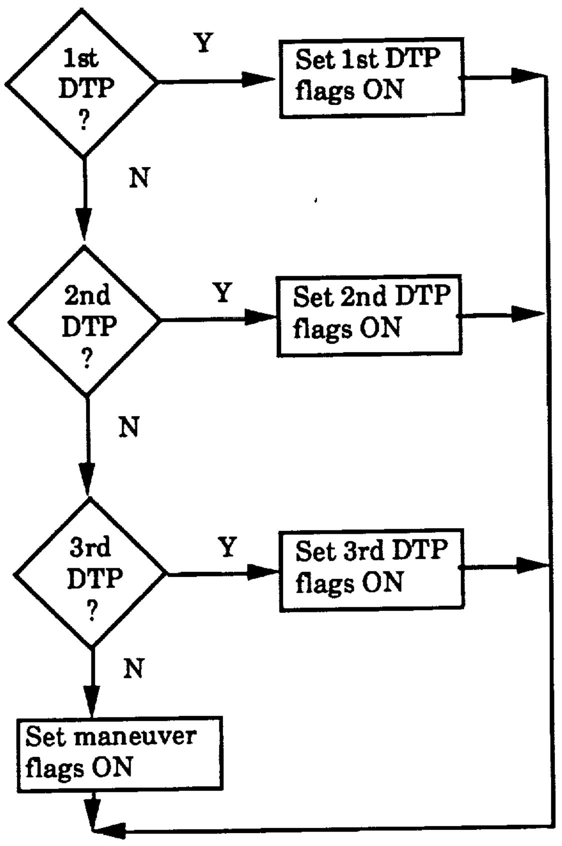
        ENV INIT (A-15)

        - Initialize filter dynamics (same model as filter)
        - Incorporate environment state vector into environment transformation matrices

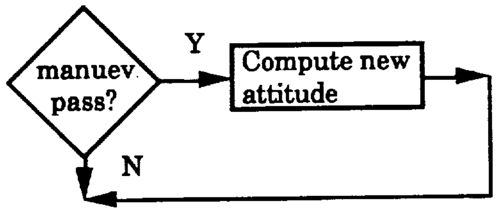
    - while T_SIM ≤ T_END,

        SIM SEQUENCER (A-17)

        - Zero control flags



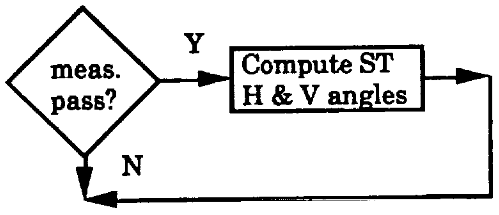
<u>ENV CONTROL</u> (A-19)

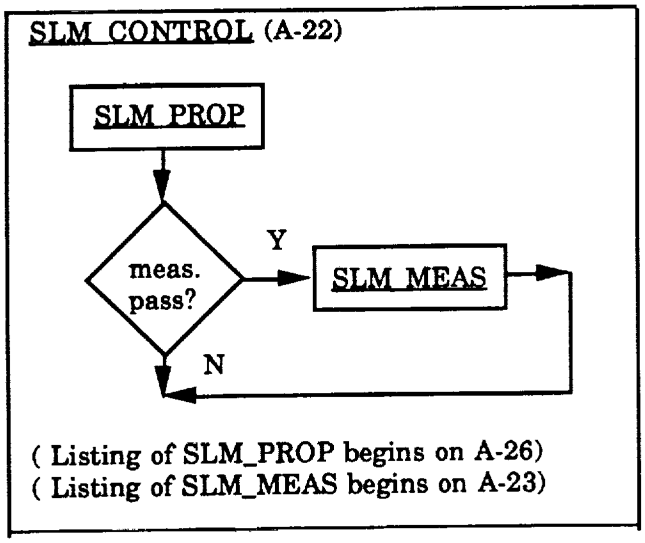


- Propagate environment state:

$$x_{i+1} = \Phi x_i + w$$

- Incorporate environment state vector into environment transformation matrices





(Listing of SLM_PROP begins on A-26)
(Listing of SLM_MEAS begins on A-23)

```
        • Save plotting data

        • T_SIM = T_SIM + DT_SIM

    • End while
```

PROGRAM LISTING

Following are lists of the *Matlab* scripts and functions which compose the simulation program. The modules are listed in the order in which they are called. For information about the conventions and standard functions of *Matlab*, refer to the *Matlab User's Guide*, available from *The Mathworks, Inc.* 20 North Main St., Suite 250, Sherborn, MA 01770.

A capitalization convention is used in the program to indicate global (i.e. shared or common) variables. There is no analogous provision in *Matlab* for named common blocks. Note that the only purpose of the main script, *slm_sim*, is to declare these global variables, and then to call the executive function, *sim_exec*.

```
%-----------------------------------------------------------------------

%...STAR LINE MANEUVER SIMULATION SCRIPT

%...Version of 17 Apr 90

%...Purpose: main script for SLM simulation; initializes global variables
%            and iloads

%-----------------------------------------------------------------------

     clear
     pack

%...DECLARE GLOBAL VARIABLES

%...SLM processor

     global  E  F_FILT  X_FILT  M_IMU_TO_I_FILT  M_ST_MISALIGN_FILT ...
             M_BAFE_TO_I_FILT  M_BAFE_TO_ST1_FILT  M_BAFE_TO_ST2_FILT ...
             DT_SLM  M_ST1_TO_BAFE  ALIGN_CORRELATED

%...Environment model processor

     global  F_ENV  X_ENV  M_IMU_TO_I_ENV  M_SB_MISALIGN_ENV ...
             M_ST_MISALIGN_ENV  M_BAFE_TO_ST1_ENV  M_BAFE_TO_ST2_ENV ...
             M_BAFE_TO_I_ENV

%...I-loads used by both of above processors

     global  SIG_STS_IAL  SIG_SB  SIG_DB_IC  SIG_DYN_BEND  SIG_GD  SIG_ST1 ...
             SIG_ST2  TAU_DB  ST_MEAS_VAR  M_ST1_TO_ST2  M_BAFE_TO_BSTS ...
             M_BSTS_TO_ST1  M_BSTS_TO_ST2  U_STAR  D2R  SEC_2_RAD  DPH_2_RPS

%...Coarse alignment

     global  M_BSTS_TO_I

%...IMU model

     global M_BAFE_TO_I  MAN_RATE  MAN_AXIS  T_MAN

%...ST model

     global  V  H  ST_ID  STAR_ID

%...Sim sequencing and timing

     global  T_SIM  DT_SIM  T_START_MEAS  T_END  T_SIM_START ...
             STAR_PAIRS  ST_ID_ARRAY  MAN_AXES

%...Plot arrays

     global  PLOT_FREQ  FILT_RMS_ERRS  TRUE_ERRS  T_PLOT  PLOT_ARRAY_SIZE

%...Names of data files
```

```
     global  DIARY_NAME  MAT_FILE_NAME

%...BEGIN SIMULATION

     sim_exec

%...END SLM_SIM
%-------------------------------------------------------------------------
```

```
function sim_exec

%------------------------------------------------------------------------

%...STAR LINE MANEUVER SIMULATION EXECUTIVE

%...Version of  2 May 90

%------------------------------------------------------------------------

%...I-loads

     iloadslm;      % Original baseline case
%     ilod_rh0;      % R. Hain inital attitude and stars; comparison case
%     ilod_rh4;      % R. Hain initial attitude and stars; 60 deg 2nd maneuv.
%     ilods_mc;      % Monte Carlo case: R. Hain baseline case, with
                    % 60 degree second maneuver

%...Comment out the following if this is not a Monte Carlo case

%     load case_number;     % if this is first run, make sure to reset!
%     case_number = case_number + 1;
%     save case_number;
%     case_string = int2str(case_number);
%     eval(['delete ', DIARY_NAME, case_string,'.dat' ])
%     eval(['diary ', DIARY_NAME, case_string,'.dat' ])

%...Open diary file and print header

%     eval(['delete ', DIARY_NAME ])
%     eval(['diary ', DIARY_NAME ])
     stime = fix(clock);
     fprintf('\n STAR LINE MANEUVER SIMULATION \n')
     COMMENT
%     case_number                       % for Monte Carlo only
     fprintf('\n Sim started at ')
     fprintf('%2.0f:%2.0f:%2.0f,',stime(4),stime(5),stime(6))
     fprintf('%2.0f/%2.0f/%4.0f \n',stime(2),stime(3),stime(1))
     hold off

%...Coarse align

     M_BAFE_TO_I = M_BSTS_TO_I * M_BAFE_TO_BSTS;

%...Initialize SLM algorithm

     slm_init;

%...Initialize Environment

     env_init;

%...Print initial environment and filter state vectors, and initial
%   filter covariance matrix rms errors

     xe = X_ENV ./SEC_2_RAD;
     xf =X_FILT ./SEC_2_RAD;
     ee = diag(E) .^0.5 ./SEC_2_RAD;

    fprintf('\n INIT ENV STATE \n')
    fprintf('  imu misalignment %10.2f %10.2f %10.2f \n', xe(1),xe(2),xe(3) )
    fprintf('  plb static misal %10.2f %10.2f %10.2f \n', xe(4),xe(5),xe(6) )
    fprintf('  gyro drift rate  %10.2f %10.2f %10.2f \n', xe(7),xe(8),xe(9) )
    fprintf('  st1 to st2 misal %10.2f %10.2f %10.2f \n', xe(10),xe(11),xe(12) )
    fprintf('  plb dyn bend mis %10.2f %10.2f %10.2f \n', xe(13),xe(14),xe(15) )

    fprintf('\n INIT FILT STATE \n')
    fprintf('  imu misalignment %10.2f %10.2f %10.2f \n', xf(1),xf(2),xf(3) )
    fprintf('  plb static misal %10.2f %10.2f %10.2f \n', xf(4),xf(5),xf(6) )
    fprintf('  gyro drift rate  %10.2f %10.2f %10.2f \n', xf(7),xf(8),xf(9) )
    fprintf('  st1 to st2 misal %10.2f %10.2f %10.2f \n', xf(10),xf(11),xf(12) )
    fprintf('  plb dyn bend mis %10.2f %10.2f %10.2f \n', xf(13),xf(14),xf(15) )

    fprintf('\n INIT FILT COV - RMS ERRORS \n')
    fprintf('  imu misalignment %10.2f %10.2f %10.2f \n', ee(1),ee(2),ee(3) )
    fprintf('  plb static misal %10.2f %10.2f %10.2f \n', ee(4),ee(5),ee(6) )
    fprintf('  gyro drift rate  %10.2f %10.2f %10.2f \n', ee(7),ee(8),ee(9) )
    fprintf('  st1 to st2 misal %10.2f %10.2f %10.2f \n', ee(10),ee(11),ee(12) )
    fprintf('  plb dyn bend mis %10.2f %10.2f %10.2f \n', ee(13),ee(14),ee(15) )

%...Begin simulation

     T_SIM = T_SIM_START;
     first_pass_plot = 1;
     sim_done = 0;
     while T_SIM <= T_END,
          sim_sequencer
          if T_SIM == T_END, sim_done=1;, end
          slm_sim_plotter(first_pass_plot,sim_done)
          if first_pass_plot==1, first_pass_plot=0;, end
          T_SIM = T_SIM + DT_SIM;
     end

     eval(['save ', MAT_FILE_NAME ])                    % use for single-case
%     eval(['save ', MAT_FILE_NAME, case_string, ...  % use for Monte-Carlo
%          ' FILT_RMS_ERRS TRUE_ERRS T_PLOT'])          % "    "     "
     diary off

%...End sim_exec

%-----------------------------------------------------------------------
```

```
function iloadslm
%-----------------------------------------------------------------------

%...STAR LINE MANEUVER ILOADS

%...Purpose:  to initialize constant variables and matrices for the
%             SLM simulation

%...Version of: 02 May 90

%-----------------------------------------------------------------------

%...Run Specific text matrices

     MAT_FILE_NAME = 'orig_bsln' ;
     DIARY_NAME = 'orig_bsln.dat' ;
     COMMENT = 'Baseline Case';

%...Conversion constants

     D2R = pi./180;
     SEC_2_RAD = D2R./3600;
     DPH_2_RPS = D2R./3600;

%...Sim timing parameters

     T_SIM_START = 0;
     DT_SIM = 1;
     DT_SLM = 1;
     T_START_MEAS = 5;

%...Maneuver parameters

     man_size = [ 90  90] .*D2R ;
     MAN_RATE = 0.2 .*D2R ;
     T_MAN(1) = 1./MAN_RATE.*man_size(1) + T_START_MEAS ;
     T_MAN(2) = 1./MAN_RATE.*man_size(2) + T_MAN(1) ;
%    T_END = 10;
     T_END = T_MAN(2) + 50.*DT_SIM;
     MAN_AXES = [ 0.0000  0.0000  1.0000;
                 -0.1840  0.9829  0.0000;
                  0       0       0      ]' ;
     star_table;

%...Measurement parameters

     STAR_PAIRS = [ 110 111;
                    110 112;
                    111 112;
                      0   0];
     ST_ID_ARRAY = [ 1 2;
                     1 2;
                     1 2;
                     0 0];
```

```
%...STS inertia (not currently used)

%     I_STS = [ 959576.     3146.   -252978.
&                 3146. 7178685.      -778.
&             -252978.     -778.   7529715. ];

%...Initial STS attitude

      M_BSTS_TO_I = [ 0.9829  0.1840  0
                     -0.1840  0.9829  0
                        0       0     1 ];

%...Transformation matrices

      M_BAFE_TO_BSTS = eye(3);
      m_stsnb_to_st1 = [-.0056491  .9994101 -.0338744
                         .9894338  .0006786 -.1449833
                        -.1448747 -.0343355 -.9888540 ];
      m_stsnb_to_st2 = [-.9662658 -.1833851  .1808317
                        -.1839513  .0000000 -.9829353
                         .1802558 -.9830411 -.0337339 ]';
      c_stsnb_p_ang = cos( 10.6 .*D2R);
      s_stsnb_p_ang = sin( 10.6 .*D2R);
      m_bsts_to_stsnb = [ c_stsnb_p_ang     0     s_stsnb_p_ang
                               0            1          0
                         -s_stsnb_p_ang     0     c_stsnb_p_ang ];
      M_BSTS_TO_ST1 = m_stsnb_to_st1 * m_bsts_to_stsnb ;
      M_BSTS_TO_ST2 = m_stsnb_to_st2 * m_bsts_to_stsnb ;
      M_ST1_TO_BAFE = (M_BSTS_TO_ST1 * M_BAFE_TO_BSTS)' ;
      M_ST1_TO_ST2 = M_BSTS_TO_ST2 * M_BSTS_TO_ST1' ;

%...Expected 1-sigma values of error sources

      SIG_STS_IAL = [   82;    82;    82 ] .* SEC_2_RAD ;
      SIG_SB      = [ 1200; 1200; 1200 ] .* SEC_2_RAD ;
      SIG_DB_IC   = [   24;    24;    24 ] .* SEC_2_RAD ;
      SIG_DYN_BEND= [   24;    24;    24 ] .* SEC_2_RAD ;
      SIG_GD      = [  .01;   .01;   .01 ] .* DPH_2_RPS ;
      SIG_ST1     = [   60;    60;    60 ] .* SEC_2_RAD ;
      SIG_ST2     = [   60;    60;    60 ] .* SEC_2_RAD ;
      TAU_DB      = [  400;   400;   400 ] ;
      ST_MEAS_VAR = 225 .* SEC_2_RAD.^2 ;

%...Correlation modeling flag

      ALIGN_CORRELATED = 1;

%...Initial Environment State Vector

%...If random state derived from filter covariance, set to 1:

      DERIVE_IC_FROM_COV = 1;
      SEED = 20174;

%...Alternate initial state vectors

%     x_sts_ial = 82 .* rand(3,1)
%     x_sb = 1200 .* rand(3,1)
%     x_db_ic = 24 .* rand(3,1)
%     x_st1 = 60 .* rand(3,1)
%     x_st2 = 60 .* rand(3,1)
%     x(1:3) = (x_sts_ial.^2 + x_sb.^2 + x_db_ic.^2).^.5;
%     x(4:6) = (x_sb.^2 + x_st1.^2).^.5;
%     x(7:9) = [0.01; 0.01; 0.01];
%     x(10:12) = (x_st1.^2 + x_st2.^2).^.5;
%     x(13:15) = x_db_ic

%     X_ENV = x' .*SEC_2_RAD;

%...Plot parameters

      PLOT_FREQ = 1/5;

%...End iloads

%-------------------------------------------------------------------------
```

```
function star_table

     U_STAR(:,110) = [ 0.0000; 0.0000;-1.0000];
     U_STAR(:,111) = [ 0.0000;-1.0000; 0.0000];
     U_STAR(:,112) = [ 1.0000; 0.0000; 0.0000];
     U_STAR(:,113) = [ 0.5000; 0.0000; 0.8660];
     U_STAR(:,114) = [ 0.0395;-0.0343;-0.9986];
     U_STAR(:,115) = [ 0.0343;-0.2225;-0.9743];
     U_STAR(:,116) = [ 0.0343; 0.9550;-0.2945];
     U_STAR(:,117) = [ 0.1834;-0.9830; 0.0000];
     U_STAR(:,118) = [ 0.9830; 0.1803; 0.0337];
     U_STAR(:,119) = [ 0.9830; 0.0609; 0.1730];
```

```
function slm_init

%------------------------------------------------------------------

%...STAR LINE MANEUVER FILTER INITIALIZATION

%...Version of 2 Mar 90
%   Modified:  5 Mar 90
%              8 Mar 90 (correct rot. sense of m_bsts_to_stsnb)

%...Purpose: to initialize the filter dynamics and covariance matrices

%------------------------------------------------------------------

%...Initialize system dynamics

    F_FILT = zeros(15,15) ;
    F_FILT(13:15,13:15) = diag( -1 ./TAU_DB );
    X_FILT = zeros(15,1) ;

%...Initialize covariance

    sig_afe_ial = (SIG_STS_IAL.^2 + SIG_SB.^2 + SIG_DB_IC.^2).^0.5 ;
    sig_plb = (SIG_SB.^2 + SIG_ST1.^2).^0.5 ;
    sig_st = (SIG_ST1.^2 + SIG_ST2.^2).^0.5 ;

    E = zeros(15,15);
    E(1:3,1:3) = diag(sig_afe_ial.^2) ;
    E(4:6,4:6) = diag(sig_plb.^2) ;
    E(7:9,7:9) = diag(SIG_GD.^2) ;
    E(10:12,10:12) = diag(sig_st.^2) ;
    E(13:15,13:15) = diag(SIG_DB_IC.^2) ;

%...If correlations are modeled, initialize off-diagonal elements of E

    if ALIGN_CORRELATED,

%...Init correlation between IMU misalign and plb static bending

        E(1:3,4:6) = M_BAFE_TO_I * diag(SIG_SB.^2) ;
        E(4:6,1:3) = E(1:3,4:6)' ;

%...Init correlation bet IMU misalign and plb dynamic bending

        E(1:3,13:15) = M_BAFE_TO_I * diag(SIG_DB_IC.^2) ;
        E(13:15,1:3) = E(1:3,13:15)' ;

%...Init correlation bet plb static bending and star tracker misalign

        E(4:6,10:12) = M_ST1_TO_BAFE * diag(SIG_ST1.^2) * M_ST1_TO_ST2' ;
        E(10:12,4:6) = E(4:6,10:12)' ;

    end

%...Initialize misalignment transformation matrices

    slm_update_transf;
```

```
%...End slm_init

%------------------------------------------------------------------------
```

```
function slm_update_transf

%-----------------------------------------------------------------------

%...STAR LINE MANEUVER TRANSFORMATION MATRIX UPDATE

%...Version of 5 Mar 90

%...Purpose: to update the time-varying transformation matrices
%            which are derived from the state vector of misalignments

%-----------------------------------------------------------------------

%...Using 1st-order expansion, approximate misalignment transf's

     M_IMU_TO_I_FILT = eye(3) - xmat( X_FILT(1:3) ) ;
     m_sb_misalign = M_BSTS_TO_ST1 * ( eye(3) - xmat( X_FILT(4:6) ) )' ;
     M_ST_MISALIGN_FILT = eye(3) - xmat( X_FILT(10:12) ) ;
     m_db_misalign = ( eye(3) - xmat( X_FILT(13:15) ) )' ;

%...Update misalignment transformations

     M_BAFE_TO_I_FILT = M_IMU_TO_I_FILT' * M_BAFE_TO_I ;
     M_BAFE_TO_ST1_FILT = m_sb_misalign * m_db_misalign * M_BAFE_TO_BSTS ;
     M_BAFE_TO_ST2_FILT = M_ST_MISALIGN_FILT * M_ST1_TO_ST2 * ...
                          M_BAFE_TO_ST1_FILT ;

%...End slm_update_transf

%-----------------------------------------------------------------------
```

```
function env_init

%-----------------------------------------------------------------------

%...STAR LINE MANEUVER ENVIRONMENT INITIALIZATION

%...Version of 2 Mar 90
%   Modified:  5 Mar 90
%              5 Apr 90 (to base random number seed on clock)
%              9 Apr 90 (change from svd to eig)

%-----------------------------------------------------------------------


%...Initialize env system dynamics matrix

     F_ENV = zeros(15,15) ;
     F_ENV(13:15,13:15) = diag( -1 ./TAU_DB );

%...Initialize static misalignment transformation matrices using
%   1st-order Taylor expansion

     temp = eye(3) - xmat( X_ENV(4:6) ) ;% + ...
%                   0.5*( X_ENV(4:6) * X_ENV(4:6)' - eye(3) );
     M_SB_MISALIGN_ENV = M_BSTS_TO_ST1 * temp' ;
     M_ST_MISALIGN_ENV = eye(3) - xmat( X_ENV(10:12) ) ;% + ...
%                   0.5*( X_ENV(10:12) * X_ENV(10:12)' - eye(3) );

     env_update_transf;

%...End env_init

%-----------------------------------------------------------------------
```

```
function env_update_transf

%------------------------------------------------------------------------

%...STAR LINE MANEUVER ENVIRONMENT TRANSFORMATION MATRIX UPDATE

%...Version of 2 Mar 90
%   Modified:  5 Mar 90

%...Purpose: to update the time-varying transformation matrices
%            which are derived from the state vector of misalignments

%------------------------------------------------------------------------

%...Using 2nd-order expansion, approximate misalignment transf's

     M_IMU_TO_I_ENV = eye(3) - xmat( X_ENV(1:3) ) ;% + ...
%                    0.5*( X_ENV(1:3) * X_ENV(1:3)' - eye(3) );
     m_db_misalign = ( eye(3) - xmat( X_ENV(13:15) ) )' ;% + ...
%                    0.5*( X_ENV(13:15) * X_ENV(13:15)' - eye(3) ) )' ;

%...Update misalignment transformations

     M_BAFE_TO_I_ENV    = M_IMU_TO_I_ENV' * M_BAFE_TO_I ;
     M_BAFE_TO_ST1_ENV = M_SB_MISALIGN_ENV * m_db_misalign * ...
                          M_BAFE_TO_BSTS ;
     M_BAFE_TO_ST2_ENV = M_ST_MISALIGN_ENV * M_ST1_TO_ST2 * ...
                          M_BAFE_TO_ST1_ENV ;

%...End env_update_transf

%------------------------------------------------------------------------
```

```
function sim_sequencer

%------------------------------------------------------------------------

%...SLM SIMULATION SEQUENCER

%...Version of 17 Apr 90

%------------------------------------------------------------------------

%...Zero control flags

     env_meas_this_pass = 0;
     env_att_prop_this_pass = 0;
     slm_meas_this_pass = 0;

     if T_SIM < T_START_MEAS,
          MAN_AXIS = [0;0;0];

%...1st measurement interval

     elseif T_SIM == T_START_MEAS,
          env_meas_this_pass = 1;
          slm_meas_this_pass = 1;
          STAR_ID = STAR_PAIRS(1,:)
          ST_ID = ST_ID_ARRAY(1,1)

     elseif T_SIM == T_START_MEAS + DT_SIM,
          env_meas_this_pass = 1;
          slm_meas_this_pass = 1;
          STAR_ID = STAR_PAIRS(1,:)
          ST_ID = ST_ID_ARRAY(1,2)
          MAN_AXIS = MAN_AXES(:,1)

%...2nd measurement interval

     elseif T_SIM == T_MAN(1),
          env_meas_this_pass = 1;
          slm_meas_this_pass = 1;
          STAR_ID = STAR_PAIRS(2,:)
          ST_ID = ST_ID_ARRAY(2,1)

     elseif T_SIM == T_MAN(1) + DT_SIM,
          env_meas_this_pass = 1;
          slm_meas_this_pass = 1;
          STAR_ID = STAR_PAIRS(2,:)
          ST_ID = ST_ID_ARRAY(2,2)
          MAN_AXIS = MAN_AXES(:,2)

%...3rd measurement interval

     elseif T_SIM == T_MAN(2),
          env_meas_this_pass = 1;
          slm_meas_this_pass = 1;
          STAR_ID = STAR_PAIRS(3,:)
          ST_ID = ST_ID_ARRAY(3,1)

     elseif T_SIM == T_MAN(2) + DT_SIM,
          env_meas_this_pass = 1;
          slm_meas_this_pass = 1;
          STAR_ID = STAR_PAIRS(3,:)
          ST_ID = ST_ID_ARRAY(3,2)

%...If not measuring, then maneuver

     else
          env_att_prop_this_pass = 1;

     end

%...Call env and slm controllers with sequencing flags

     env_control(env_meas_this_pass,env_att_prop_this_pass)
%    M_BAFE_TO_I_ENV
%    if ST_ID ==1,
%         M_BAFE_TO_ST1_ENV
%    elseif ST_ID ==2,
%         M_BAFE_TO_ST2_ENV
%    end
     slm_control(slm_meas_this_pass)
%    M_BAFE_TO_I_FILT
%    if ST_ID ==1,
%         M_BAFE_TO_ST1_FILT
%    elseif ST_ID ==2,
%         M_BAFE_TO_ST2_FILT
%    end
%   xf =X_FILT*180*3600/pi;
    diary off
    fprintf('\n T_SIM = %6.2f  \n', T_SIM)
    diary on
%   fprintf(' FILT STATE - ARC SEC \n')
%   fprintf('  imu misalignment %10.2g %10.2g %10.2g \n', xf(1),xf(2),xf(3) )
%   fprintf('  plb static misal %10.2g %10.2g %10.2g \n', xf(4),xf(5),xf(6) )
%   fprintf('  gyro drift rate  %10.2g %10.2g %10.2g \n', xf(7),xf(8),xf(9) )
%   fprintf('  st1 to st2 misal %10.2g %10.2g %10.2g \n', xf(10),xf(11),xf(12) )
%   fprintf('  plb dyn bend mis %10.2g %10.2g %10.2g \n', xf(13),xf(14),xf(15) )

%...End sim_sequencer

%-----------------------------------------------------------------------
```

```
function env_control(env_meas_this_pass,env_att_prop_this_pass)

%-----------------------------------------------------------------

%...SLM ENVIRONMENT MODEL CONTROLLER

%...Version of 7 Mar 90
%   Modified  16 Mar 90
%              3 Apr 90 (to put env state prop before env meas)
%                       (also to fix coding error - F_ENV for F)
%             11 Apr 90 (to add state noise)

%-----------------------------------------------------------------

%...If this is a maneuver pass, compute new sts attitude

    if env_att_prop_this_pass,

%       M_BSTS_TO_I = imu_model( M_BSTS_TO_I );
        M_BSTS_TO_I = l_rate_att_prop( M_BSTS_TO_I, MAN_AXIS, ...
                          MAN_RATE, DT_SIM);
        M_BAFE_TO_I = M_BSTS_TO_I * M_BAFE_TO_BSTS ;

    end

%...Propagate env state vector and update transf matrices

    rand('normal')
    F_ENV(1:3,7:9) = M_BAFE_TO_I_ENV ;
    phi = eye(15) + F_ENV*DT_SIM + 0.5*F_ENV*DT_SIM^2 ;
    X_ENV = phi * X_ENV ;
    sig_db_env = [50;50;50]./3600.*pi./180;
    state_noise = sqrt( diag(eye(3) - phi(13:15,13:15)^2 ) ) ...
                  .* sig_db_env .* rand(3,1) ;
    X_ENV(13:15) = X_ENV(13:15) + state_noise;
    env_update_transf

%...If this is a measurement pass, compute star tracker offset angles

    if env_meas_this_pass,

        [ V(1) H(1) ] = env_st_model( U_STAR(:, STAR_ID(1) ), ...
                        M_BAFE_TO_I_ENV, M_BAFE_TO_ST1_ENV );
        [ V(2) H(2) ] = env_st_model( U_STAR(:, STAR_ID(2) ), ...
                        M_BAFE_TO_I_ENV, M_BAFE_TO_ST2_ENV )
        T_MEAS = T_SIM

    end

%...End env_control

%-----------------------------------------------------------------
```

```
function [m_b_to_i_new] = l_rate_att_prop(m_b_to_i_old, e_axis, w, dt)

%-----------------------------------------------------------------------

%...LOW RATE ATTITUDE PROPAGATION FUNCTION

%...Version of 28 Feb 90
%   Modified:  12 Mar 90

%...Purpose: to update the body to inertial transformation matrix
%            as if an actual attitude state were being propagated
%            during a maneuver.  Since this method uses a small angle
%            approximation to update the current transformation matrix,
%            the product w*dt should be sufficiently small to obtain
%            whatever required accuracy is desired.

%...Inputs:                  Descr.:
%          m_b_to_i_old      Current transformation, body to inertial
%          e_axis            Eigen-axis about which maneuver occurs;
%                            specify in body-axis frame
%          w                 Maneuver rate, in radians/time unit
%          dt                Time step, in units compatible with w

%...Output:
%          m_b_to_i_new      Updated body-to-inertial transf. matrix

%-----------------------------------------------------------------------

%...Construct cross-product matrix of eigenaxis

     e = e_axis ;
     ecross = [  0    -e(3)  e(2)
                e(3)    0   -e(1)
               -e(2)  e(1)    0   ];

%...Approximate rotation matrix using 2nd-order Taylor expansion

     wdt = w*dt;
     m = eye(3) - wdt*ecross + 0.5*wdt^2*( e*e' - eye(3) );

     m(1,:) = m(1,:)/norm(m(1,:));
     m(2,:) = m(2,:)/norm(m(2,:));
     m(3,:) = m(3,:)/norm(m(3,:));
     m_old_to_new = m;

%...Compute new body to inertial transformation

     m_b_to_i_new = m_b_to_i_old * m_old_to_new';

%...End l_rate_att_prop

%-----------------------------------------------------------------------
```

```
function [v, h] = env_st_model( u_i, m_b_to_i, m_b_to_st )

%-----------------------------------------------------------------------

%...STAR LINE MANEUVER ENVIRONMENT STAR TRACKER MODEL

%...Version of 2 Mar 90
%   Modified   3 Apr 90 (to include meas noise)

%...Purpose: to simulate the output average v and h angles from the
%            sts star trackers

%...Inputs:            Descr.:                       Source(s):
%         u_i          Inertial position of star
%         m_b_to_i     Body-to-inertial transformation
%         m_b_to_st    Body-to-star tracker transf

%...Outputs:
%         v            Vertical offset of star from center
%                      of star tracker field of view
%         h            Horizontal offset

%-----------------------------------------------------------------------

%...Transform star line of sight to star tracker frame

     m_i_to_st = m_b_to_st * m_b_to_i' ;
     u_st = m_i_to_st * u_i;

%...Compute v and h angles

     v = atan( -u_st(1) / u_st(3) );
     h = atan(  u_st(2) / u_st(3) );

%...Add measurement noise

     sig_st = sqrt(ST_MEAS_VAR);
     rand('normal');
     t = clock;
     rand('seed',t(6)*7348)
     fprintf('\n Seed for meas noise is %8.2f \n', rand('seed'))
     bias_st = 0;
     v = v + sig_st .* rand + bias_st;
     h = h + sig_st .* rand + bias_st;

%...End env_st_model

%-----------------------------------------------------------------------
```

```
function slm_control(slm_meas_this_pass)

%----------------------------------------------------------------------

%...SLM EVENT CONTROLLER

%...Version of 16 Mar 90
%   Modified   19 Mar 90
%               9 Apr 90  (to correlate filt and env propagation)

%----------------------------------------------------------------------

%...Measurement passes

     if slm_meas_this_pass,

          slm_meas

     end

%...Prop filt state on measurement and manuever passes

     slm_prop

%...End slm_control

%----------------------------------------------------------------------
```

```
function slm_meas
%-----------------------------------------------------------------
%...STAR LINE MANEUVER MEASUREMENT PROCESSING AND FILTER UPDATE
%...Version of 19 Mar 90


%-----------------------------------------------------------------
%...Two passes are performed, 1st for V angle, then for H angle meas

     for i = 1:2,

%...Filter's estimate for star los in appropriate st frame

          u_st_est = M_BAFE_TO_ST1_FILT * M_BAFE_TO_I_FILT' * ...
                     U_STAR( :, STAR_ID( ST_ID ) );
          if ST_ID == 2,
               u_st_est = M_ST_MISALIGN_FILT * M_ST1_TO_ST2 * u_st_est;
          end
          u_st_est = u_st_est / norm( u_st_est )

%...Construct from V and H angles the measured los

          u_st_meas = [-tan( V(ST_ID) ); tan( H(ST_ID) ); 1];
          u_st_meas = u_st_meas / norm(u_st_meas)

%...Measurement geometry vectors (1st 2 cols of b)

          b = calc_b_vec

%...Measurement residual

          del_q = u_st_meas(i) - u_st_est(i)

%...Mean-squared residual and residual test ratio (w/ 6-sigma edit)

          eb = E * b(:,i);
          ms_residual = b(:,i)' * eb + ST_MEAS_VAR
          r = abs(del_q) / (6 * sqrt(ms_residual) )

%...Update covariance (and resymmetrize) and update state

          if r < 1,
               w = eb / ms_residual;
               w'    % display weighting vector
               E = E - w * eb';
               E = E - tril(E) + triu(E)';
               X_FILT = X_FILT + w * del_q;
               slm_update_transf
          end

     xe = X_ENV ./SEC_2_RAD;
     xf =X_FILT ./SEC_2_RAD;
     ee = diag(E) .^0.5 ./SEC_2_RAD;
     fprintf('\n ENV STATE \n')
     fprintf('  imu misalignment %10.2f %10.2f %10.2f \n', xe(1),xe(2),xe(3) )
     fprintf('  plb static misal %10.2f %10.2f %10.2f \n', xe(4),xe(5),xe(6) )
     fprintf('  gyro drift rate  %10.2f %10.2f %10.2f \n', xe(7),xe(8),xe(9) )
     fprintf('  st1 to st2 misal %10.2f %10.2f %10.2f \n', xe(10),xe(11),xe(12) )
     fprintf('  plb dyn bend mis %10.2f %10.2f %10.2f \n', xe(13),xe(14),xe(15) )

     fprintf('\n FILT STATE \n')
     fprintf('  imu misalignment %10.2f %10.2f %10.2f \n', xf(1),xf(2),xf(3) )
     fprintf('  plb static misal %10.2f %10.2f %10.2f \n', xf(4),xf(5),xf(6) )
     fprintf('  gyro drift rate  %10.2f %10.2f %10.2f \n', xf(7),xf(8),xf(9) )
     fprintf('  st1 to st2 misal %10.2f %10.2f %10.2f \n', xf(10),xf(11),xf(12))
     fprintf('  plb dyn bend mis %10.2f %10.2f %10.2f \n', xf(13),xf(14),xf(15))

     fprintf('\n FILT COV - RMS ERRORS \n')
     fprintf('  imu misalignment %10.2f %10.2f %10.2f \n', ee(1),ee(2),ee(3) )
     fprintf('  plb static misal %10.2f %10.2f %10.2f \n', ee(4),ee(5),ee(6) )
     fprintf('  gyro drift rate  %10.2f %10.2f %10.2f \n', ee(7),ee(8),ee(9) )
     fprintf('  st1 to st2 misal %10.2f %10.2f %10.2f \n', ee(10),ee(11),ee(12) )
     fprintf('  plb dyn bend mis %10.2f %10.2f %10.2f \n', ee(13),ee(14),ee(15) )

      end

%...End slm_meas

%------------------------------------------------------------------------
```

```
function [b] = calc_b_vec

%-----------------------------------------------------------------------

%...STAR LINE MANEUVER MEASUREMENT GEOMETRY VECTOR COMPUTATION

%...Version of 19 Mar 90

%-----------------------------------------------------------------------

%...Cross-product matrices of star los in some useful frames

     u_i_cross = xmat( U_STAR( :, STAR_ID( ST_ID ) ) );
     u_b = M_BAFE_TO_I_FILT' * U_STAR( :, STAR_ID( ST_ID ) );
     u_b_cross = xmat( u_b );

%...Include st1-to-st2 bending effects if this is a st2 meas pass

     if ST_ID == 2,
          u_st2_cross = xmat( M_BAFE_TO_ST2_FILT * u_b );
          b(10:12,1:3) = u_st2_cross';
          m_b_to_st = M_BAFE_TO_ST2_FILT;
     else,
          b(10:12,1:3) = zeros(3,3);
          m_b_to_st = M_BAFE_TO_ST1_FILT;
     end

%...Compute remaining portion of b-matrix

     b(1:3,1:3) = ( m_b_to_st * M_BAFE_TO_I_FILT' * u_i_cross )';
     b(4:6,1:3) = - ( m_b_to_st * u_b_cross )';
     b(7:9,1:3) = zeros(3,3);
     b(13:15,1:3) = b(4:6,1:3);

%...End calc_b_vec

%-----------------------------------------------------------------------


function x = xmat(v)

x = [  0    -v(3)   v(2)
      v(3)    0    -v(1)
     -v(2)   v(1)    0   ];

%...End cross
```

```
function slm_prop

%-------------------------------------------------------------------

%...STAR LINE MANEUVER PROPAGATION FUNCTION

%...Version of 19 Mar 90

%-------------------------------------------------------------------

%...Update the portion of the filter dynamics matrix which
%   transforms the gyro drift rate vector into an IMU alignment
%   perturbation

     F_FILT(1:3,7:9) = M_BAFE_TO_I_FILT;

%...Approximate state transition matrix

     phi = eye(15) + F_FILT*DT_SLM + 0.5 * F_FILT^2 * DT_SLM^2;

%...Propagate filter state vector

     X_FILT = phi*X_FILT;

%...Construct random noise portion of the ECRV which models
%   dynamic bending

     q_db = diag( (eye(3) - phi(13:15,13:15)^2 ) * SIG_DYN_BEND.^2 );

%...Propagate filter covariance

     E = phi * E * phi' ;
     E(13:15,13:15) = E(13:15,13:15) + q_db ;

%...Update transformation matrices

     slm_update_transf

%...End propagation

%-------------------------------------------------------------------
```

```
function slm_sim_plotter(first_pass, sim_done)

%-----------------------------------------------------------------

%...STAR LINE MANEUVER PLOTTING FUNCTION

%...Version of 20 Mar 90
%   Modified:  26 Mar 90

%-----------------------------------------------------------------

%...Initialize plot variables if 1st pass

     if first_pass,
          PLOT_ARRAY_SIZE = T_END./DT_SIM.*PLOT_FREQ + 1;
          FILT_RMS_ERRS = zeros(15, PLOT_ARRAY_SIZE);
          TRUE_ERRS = zeros(15, PLOT_ARRAY_SIZE);
          T_PLOT = zeros(1, PLOT_ARRAY_SIZE);
     end

%...Fill plot arrays

     if rem(T_SIM,1/PLOT_FREQ) == 0,
          i = T_SIM.*PLOT_FREQ + 1;
          FILT_RMS_ERRS(:,i) = diag(E).^0.5 ./SEC_2_RAD;
          TRUE_ERRS(:,i) = (X_ENV - X_FILT) ./SEC_2_RAD;
          T_PLOT(i) = T_SIM;
     end

%...Perform plotting if sim is finished

     if sim_done,
          for i = 1:PLOT_ARRAY_SIZE,
               ial_filt_rss(:,i) = norm(FILT_RMS_ERRS(1:3,i));
               sb_filt_rss(:,i) = norm(FILT_RMS_ERRS(4:6,i));
               gd_filt_rss(:,i) = norm(FILT_RMS_ERRS(7:9,i));
               st_filt_rss(:,i) = norm(FILT_RMS_ERRS(10:12,i));
               db_filt_rss(:,i) = norm(FILT_RMS_ERRS(13:15,i));
               ial_err_rss(:,i) = norm(TRUE_ERRS(1:3,i));
               sb_err_rss(:,i) = norm(TRUE_ERRS(4:6,i));
               gd_err_rss(:,i) = norm(TRUE_ERRS(7:9,i));
               st_err_rss(:,i) = norm(TRUE_ERRS(10:12,i));
               db_err_rss(:,i) = norm(TRUE_ERRS(13:15,i));
          end

          ftime = fix(clock);
          fprintf('\n Sim finished at ')
          fprintf('%2.0f:%2.0f:%2.0f, ',ftime(4),ftime(5),ftime(6))
          fprintf('%2.0f/%2.0f/%4.0f \n',ftime(2),ftime(3),ftime(1))
          fprintf('\n Hit any key to begin plotting... \n')
          pause


          plot(T_PLOT,ial_filt_rss,'.',T_PLOT,ial_err_rss,'-')
          title('Filter RMS, True Errors')
          xlabel('time, sec')
          ylabel('RSS IMU Alignment, arcsec')
          pause
```

```
        plot(T_PLOT,sb_filt_rss,'.',T_PLOT,sb_err_rss,'-')
        title('Filter RMS, True Errors')
        xlabel('time, sec')
        ylabel('RSS PLB Static Bending, arcsec')
        pause


        plot(T_PLOT,gd_filt_rss,'.',T_PLOT,gd_err_rss,'-')
        title('Filter RMS, True Errors')
        xlabel('time, sec')
        ylabel('RSS Gyro Drift Rate, deg/hr')
        pause


        plot(T_PLOT,st_filt_rss,'.',T_PLOT,st_err_rss,'-')
        title('Filter RMS, True Errors')
        xlabel('time, sec')
        ylabel('RSS ST1 to ST2 misalign, arcsec')
        pause


        plot(T_PLOT,db_filt_rss,'.',T_PLOT,db_err_rss,'-')
        title('Filter RMS, True Errors')
        xlabel('time, sec')
        ylabel('RSS PLB Dyn Bending, arcsec')
        pause

%       plot(T_PLOT,ial_filt_rss)
%       title('Filter RMS Misalign Errors')
%       xlabel('time, sec')
%       ylabel('RSS IMU Alignment, arcsec')
%       pause

%       plot(T_PLOT,sb_filt_rss)
%       title('Filter RMS Misalign Errors')
%       xlabel('time, sec')
%       ylabel('RSS PLB Static Bending, arcsec')
%       pause

%       plot(T_PLOT,gd_filt_rss)
%       title('Filter RMS Misalign Errors')
%       xlabel('time, sec')
%       ylabel('RSS Gyro Drift Rate, deg/hr')
%       pause

%       plot(T_PLOT,st_filt_rss)
%       title('Filter RMS Misalign Errors')
%       xlabel('time, sec')
%       ylabel('RSS ST1 to ST2 misalign, arcsec')
%       pause

%       plot(T_PLOT,db_filt_rss)
%       title('Filter RMS Misalign Errors')
%       xlabel('time, sec')
%       ylabel('RSS PLB Dyn Bending, arcsec')
%       pause
%         plot(T_PLOT,ial_err_rss)
%         title('True Errors')
%         xlabel('time, sec')
%         ylabel('RSS IMU alignment, arcsec')
%         pause

%         plot(T_PLOT,sb_err_rss)
%         title('True Errors')
%         xlabel('time, sec')
%         ylabel('RSS PLB Static Bending, arcsec')
%         pause

%         plot(T_PLOT,gd_err_rss)
%         title('True Errors')
%         xlabel('time, sec')
%         ylabel('RSS Gyro Drift Rate, deg/hr')
%         pause

%         plot(T_PLOT,st_err_rss)
%         title('True Errors')
%         xlabel('time, sec')
%         ylabel('RSS ST1 to ST2 misalign, arcsec')
%         pause

%         plot(T_PLOT,db_err_rss)
%         title('True Errors')
%         xlabel('time, sec')
%         ylabel('RSS PLB Dyn Bending, arcsec')
%         pause

     end

%...End slm_sim_plotter

%----------------------------------------------------------------------
```

```
%...SLM post-sim plotting script
%...User must load in the proper data file, or comment out the following:

     load slm_mats.mat

     clg
     subplot(221)
     semilogy(t_plot,FILT_RMS_ERRS(1,:),'--',t_plot,abs(TRUE_ERRS(1,:)),':')
     xlabel('time, sec')
     ylabel('IAL1')
     subplot(222)
     semilogy(t_plot,FILT_RMS_ERRS(2,:),'--',t_plot,abs(TRUE_ERRS(2,:)),':')
     xlabel('time, sec')
     ylabel('IAL2')
     subplot(223)
     semilogy(t_plot,FILT_RMS_ERRS(3,:),'--',t_plot,abs(TRUE_ERRS(3,:)),':')
     xlabel('time, sec')
     ylabel('IAL3')
     text(0.6, 0.3, 'Filt RMS Errors  --', 'sc')
     text(0.6, 0.2, 'True Errors      : ', 'sc')
     pause

     clg
     subplot(221)
     semilogy(t_plot,FILT_RMS_ERRS(4,:),'--',t_plot,abs(TRUE_ERRS(4,:)),':')
     xlabel('time, sec')
     ylabel('PLB1')
     subplot(222)
     semilogy(t_plot,FILT_RMS_ERRS(5,:),'--',t_plot,abs(TRUE_ERRS(5,:)),':')
     xlabel('time, sec')
     ylabel('PLB2')
     subplot(223)
     semilogy(t_plot,FILT_RMS_ERRS(6,:),'--',t_plot,abs(TRUE_ERRS(6,:)),':')
     xlabel('time, sec')
     ylabel('PLB3')
     text(0.6, 0.3, 'Filt RMS Errors  --', 'sc')
     text(0.6, 0.2, 'True Errors      : ', 'sc')
     pause

     clg
     axis([0 1000 -3 -1])
     subplot(221)
     semilogy(t_plot,FILT_RMS_ERRS(7,:),'--',t_plot,abs(TRUE_ERRS(7,:)),':')
     xlabel('time, sec')
     ylabel('GDR1')
     subplot(222)
     semilogy(t_plot,FILT_RMS_ERRS(8,:),'--',t_plot,abs(TRUE_ERRS(8,:)),':')
     xlabel('time, sec')
     ylabel('GDR2')
     subplot(223)
     semilogy(t_plot,FILT_RMS_ERRS(9,:),'--',t_plot,abs(TRUE_ERRS(9,:)),':')
     xlabel('time, sec')
     ylabel('GDR3')
     text(0.6, 0.3, 'Filt RMS Errors  --', 'sc')
     text(0.6, 0.2, 'True Errors      : ', 'sc')
     axis;
     pause
```

```
        clg
        subplot(221)
        semilogy(t_plot,FILT_RMS_ERRS(10,:),'--',t_plot,abs(TRUE_ERRS(10,:)),':')
        xlabel('time, sec')
        ylabel('STM1')
        subplot(222)
        semilogy(t_plot,FILT_RMS_ERRS(11,:),'--',t_plot,abs(TRUE_ERRS(11,:)),':')
        xlabel('time, sec')
        ylabel('STM2')
        subplot(223)
        semilogy(t_plot,FILT_RMS_ERRS(12,:),'--',t_plot,abs(TRUE_ERRS(12,:)),':')
        xlabel('time, sec')
        ylabel('STM3')
        text(0.6, 0.3, 'Filt RMS Errors  --', 'sc')
        text(0.6, 0.2, 'True Errors      : ', 'sc')
        pause

        clg
        subplot(221)
        semilogy(t_plot,FILT_RMS_ERRS(13,:),'--',t_plot,abs(TRUE_ERRS(13,:)),':')
        xlabel('time, sec')
        ylabel('DYN1')
        subplot(222)
        semilogy(t_plot,FILT_RMS_ERRS(14,:),'--',t_plot,abs(TRUE_ERRS(14,:)),':')
        xlabel('time, sec')
        ylabel('DYN2')
        subplot(223)
        semilogy(t_plot,FILT_RMS_ERRS(15,:),'--',t_plot,abs(TRUE_ERRS(15,:)),':')
        xlabel('time, sec')
        ylabel('DYN3')
        text(0.6, 0.3, 'Filt RMS Errors  --', 'sc')
        text(0.6, 0.2, 'True Errors      : ', 'sc')

%...End slm_post
```

APPENDIX B - THE PROGRAM *MC POST*

The *Matlab* script below implements the equations given in Section 4.1.3 of this report. Inputs are *N*, the number of Monte Carlo cases, and *conf_limit*, which may be either 0.95 or 0.99. Defaults are assigned in the script, but the user may change these values via the keyboard while the script is running.

The program requires that two arrays containing, along their rows, the true errors and filter rms errors (as defined in the report) for each case. Only the last column of the array is required. The arrays should be stored in a *Matlab* data file (*.mat* file). The naming convention for these files is *mc_case$.mat*, where *$* should be replaced by the case number.

Note that case number 136 is excluded in the version below. This case was a repetition of case 135.

PROGRAM LISTING

```
%--- SLM Monte Carlo Simulation Post Processor -------------------------------

     N = 400
     conf_limit = .95
     keyboard
     Tru_Errs_All_Cases = zeros(N,15);
     Filt_RMS_All_Cases = zeros(N,15);

     for i=1:N,
          if i==136,i=i+1;end;
          eval(['load mc_case',int2str(i)]);
          Tru_Errs_All_Cases(i,:) = TRUE_ERRS(:,162)';
          Filt_RMS_All_Cases(i,:) = FILT_RMS_ERRS(:,162)';
     end

%...Sample mean and std dev

     tru_errs_est_mean = mean( Tru_Errs_All_Cases )
     tru_errs_est_std_dev = std( Tru_Errs_All_Cases )
     filt_rms_est_mean = mean( Filt_RMS_All_Cases )
     filt_rms_est_std_dev = std (Filt_RMS_All_Cases )
     pause
     clc

%...Confidence Intervals

     if conf_limit==.950,
          z = 1.960;
     elseif conf_limit==.990,
          z = 2.576;
     else,
          fprintf('Input value of conf_limit not allowable - ')
          fprintf('Re-enter')
          keyboard
     end
```

```
chi2_lo = gen_chi2(N-1,z);
chi2_up = gen_chi2(N-1,-z);

conf_int_xbar_tru = z.*tru_errs_est_std_dev./sqrt(N)
lo_conf_lim_sdev_tru = sqrt( (N-1).*(tru_errs_est_std_dev.^2)./chi2_lo )
up_conf_lim_sdev_tru = sqrt( (N-1).*(tru_errs_est_std_dev.^2)./chi2_up )

conf_int_xbar_filt = z.*filt_rms_est_std_dev./sqrt(N)
lo_conf_lim_sdev_filt = sqrt( (N-1).*(filt_rms_est_std_dev.^2)./chi2_lo )
up_conf_lim_sdev_filt = sqrt( (N-1).*(filt_rms_est_std_dev.^2)./chi2_up )
```

NASA
National Aeronautics and Space Administration

REPORT DOCUMENTATION PAGE

1. Report No. TM 102174	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle Simulation and Analyses of the Aeroassist Flight Experiment Attitude Update Method	5. Report Date June 1991
	6. Performing Organization Code EG 42
7. Author(s) J. R. Carpenter	8. Performing Organization Report No. S-623
9. Performing Organization Name and Address Navigation, Control, and Aeronautics Division National Aeronautics and Space Administration Johnson Space Center	10. Work Unit No.
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered Technical Memorandum
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract
A method which will be used to update the alignment of the Aeroassist Flight Experiment's Inertial Measuring Unit is simulated and analyzed. This method, the Star Line Maneuver, uses measurements from the Space Shuttle Orbiter star trackers along with an extended Kalman filter to estimate a correction to the attitude quaternion maintained by an Inertial Measuring Unit in the Orbiter's payload bay. This quaternion is corrupted by onorbit bending of the Orbiter payload bay with respect to the Orbiter navigation base, which is incorporated into the payload quaternion when it is initialized via a direct transfer of the Orbiter attitude state. The method of updating this quaternion is examined through verification of baseline cases and Monte Carlo analyses using a simplified simulation. The simulation uses nominal state dynamics and measurement models from the Kalman filter as its real world models, and is programmed on a Microvax minicomputer using Matlab, an interactive matrix analysis tool. Results are presented which confirm and augment previous performance studies, thereby enhancing confidence in the Star Line Maneuver design methodology. Additional analyses are suggested to characterize the method's robustness to more diverse real world models, and to the architecture of the data interfaces between the Aeroassist Flight Experiment and the Space Shuttle.

17. Key Words (Suggested by Author(s)) Space Shuttle, Aeroassist Flight Experiment, Star Line Maneuver, Inertial Measuring Unit, star tracker, payload bay bending, Kalman filter, Matlab, Monte Carlo analysis	18. Distribution Statement Unlimited Subject Category 17

19. Security Classification (of this report) Unclassified	20. Security Classification (of this page) Unclassified	21. No. of pages	22. Price

For sale by the National Technical Information Service, Springfield, VA 22161-2171