AUBURN UNIVERSITY

AEROSPACE

ENGINEERING

TERMINAL GUIDANCE AND NAVIGATION FOR COMET AND ASTEROID RENDEZVOUS

FINAL REPORT

Prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GEORGE C. MARSHALL SPACE FLIGHT CENTER

Under Contract NAS8-30248

June 15, 1975

ENGINEERING EXPERIMENT STATION
AUBURN UNIVERSITY
AUBURN, ALABAMA 36830

Aerospace Engineering Department
Auburn University
Auburn, Alabama 36830

TERMINAL GUIDANCE AND NAVIGATION FOR COMET AND ASTEROID RENDEZVOUS

FINAL REPORT

Prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GEORGE C. MARSHALL SPACE FLIGHT CENTER

Under Contract NAS8-30248

June 15, 1975

Arthur G. Bennett
Arthur G. Bennett, Associate Professor
Project Director

Robert G. Pitts
Robert G. Pitts, Professor and Head
Department of Aerospace Engineering

ABSTRACT

A terminal guidance and navigation scheme developed in earlier work is modified and evaluated for a solar electric propulsion rendezvous mission to comet Encke. The scheme is intended for autonomous, on board use. The guidance algorithm is based in optimal control theory and minimizes the time integrated square of thrust acceleration. The navigation algorithm employs a modified Kalman filter set in measurement variables. Random sequences were generated to simulate measurement errors and the evaluation was conducted with detailed numerical computations which include actual motions of spacecraft and comet. The evaluations showed that the scheme attains rendezvous and maintains station after rendezvous within less than 10 km for estimated "best" measurements and within less than 100 km for estimated "worst" measurements. The measurements required are angles, range and range rate. Angles and range appear to be absolutely necessary. Range rate is not as strong a measurement type and further modifications of the filter will quite probably allow a scheme that does not require the rate measurements.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLE

FIGURES

TABLE

1.0 INTRODUCTION

In previous work on the comet and asteroid navigation and guidance problem, an optimal control theory guidance algorithm was devised.[1] An evaluation of this algorithm combined with a Kalman filter method for navigation showed that onboard guidance and navigation is possible with reasonable limits on measurement errors.[2] But, this evaluation did not fully establish practicality of the onboard approach. Questions of how many measurement types are required and the effects of range of measurement accuracy were not answered. And, the particular coordinate system employed (range and direction cosines) gave unacceptable singularities near coordinate planes. Also, there was the not unusual question of management of state estimate divergence caused by the use of a linear filter with a nonlinear system.

The objective of the work presented here was to carry out further algorithm development to provide first answers to the questions above. Results were to include the precision attainable with different instrument types and accuracies and the onboard computer requirements necessary to implement the scheme.

After initiation of the work, it became evident that the conversion of the previously developed GANDER guidance and navigation computer program to a non-singular coordinate system would be a larger task than originally planned. The divergence question also presented unforeseen difficulties. It was therefore agreed to reduce work on less important tasks. Specifically, thrust errors were simulated by statistical terms in the filter equations and not by a second method wherein thrust errors would be statistically developed and handled as new state variables. Similarly, the use of available angular measurements to obtain improved estimates of target ephemeris was not investigated. These refinements, properly done, can only improve results. Since successful rendezvous was obtained without these refinements, the decision to drop them was correct.

Identification of specific instruments, their accuracies and power requirements could not be carried out because the requisite data has yet to be developed in the frequency ranges of interest for onboard radar in free space. A request to do this data development could not be funded, so these questions remain open.

2.0 REFORMULATION OF THE GUIDANCE AND NAVIGATION SCHEME

A first step in the work reported here was to eliminate the difficulties encountered with the range and direction cosine coordinates used previously in the Kalman filter. The difficulty arose because of the basic relation among direction cosines that the sum of their squares is unity. Near a coordinate plane one of the direction cosines approaches zero and small numerical differences such as arise from measurement errors can lead, in numerical computation, to imaginary values for this cosine. If this happens, the computer becomes most upset and reports in displeasure with a long string of error statements. There are ways to get around this problem, but in doing so some of the information in the data is lost. It is better to employ coordinates in which the singularity trouble does not arise. Plain rectangular coordinates would solve this problem, but would introduce a nonlinear relation between measurement variables and filter variables. It was one of our basic objectives to avoid this nonlinearity.

Coordinates that can be related to measurement variables in a one-to-one fashion are simple spherical coordinates, and it is these coordinates that were decided upon. A singularity in the various functions arising in the algorithms can arise only on the polar axis of the spherical coordinates. This singularity is not only less likely to occur, but is more easily managed if it should become necessary.

2.1 System Dynamics and the Guidance Algorithm

We here repeat the equations of motion and guidance algorithm as previously developed.[1,2]

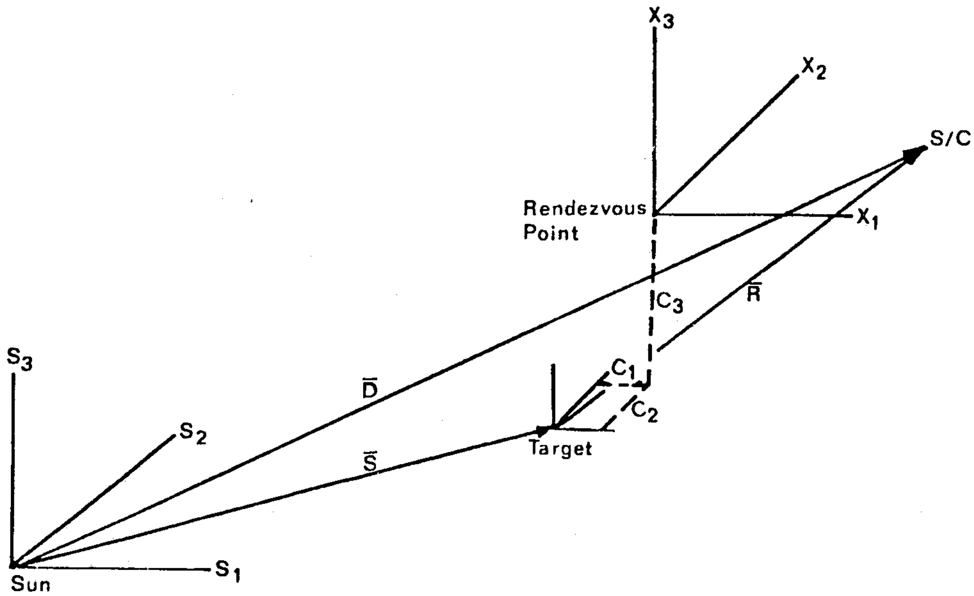


Figure 1. Rendezvous Geometry.

Appropriate to onboard guidance, the equations of motion are set in a coordinate frame fixed relative to the target as shown in Figure 1. In rectangular coordinates centered at the rendezvous point, the equations of motion, neglecting the gravitational attraction of the target, are

$$
\begin{aligned}
\dot{x}_1 &= x_4 \\
\dot{x}_2 &= x_5 \\
\dot{x}_3 &= x_6 \\
\dot{x}_4 &= F_1 + \frac{GM}{S^3}\,[S_1 - D_1(S/D)^3] \\
\dot{x}_5 &= F_2 + \frac{GM}{S^3}\,[S_2 - D_2(S/D)^3] \\
\dot{x}_6 &= F_3 + \frac{GM}{S^3}\,[S_3 - D_3(S/D)^3]
\end{aligned}
\tag{1}
$$

where the subscripts indicate components in the corresponding coordinate directions. M is the mass of the sun and G is the universal gravitational constant. From the optimal control theory guidance law, the control forces, F_1, F_2, F_3, which constitute the guidance algorithm, are

$$
\begin{aligned}
F_1 &= [\frac{6}{\tau_o^2}\,(1 - \frac{2\tau}{\tau_o})]x_{10} + [\frac{2}{\tau_o}(1 - \frac{3\tau}{\tau_o})]x_{40} \\
F_2 &= [\frac{6}{\tau_o^2}\,(1 - \frac{2\tau}{\tau_o})]x_{20} + [\frac{2}{\tau_o}(1 - \frac{3\tau}{\tau_o})]x_{50} \\
F_3 &\quad [\frac{6}{\tau_o^2}\,(1 - \frac{2\tau}{\tau_o})]x_{30} + [\frac{2}{\tau_o}(1 - \frac{3\tau}{\tau_o})]x_{60}
\end{aligned}
\tag{2}
$$

where x_{10}, x_{20}, etc., are the initial conditions and $\tau = T_F - T$ is the time to go with T_F = final rendezvous time and T = running time.

Note that the equations above are in rectangular coordinates. These coordinates are employed for the precision integration made to construct the comparison trajectory for the evaluation to be made. Transformation to the spherical coordinates for the measurement and filter variables are made in the navigation algorithm.

2.2 The Navigation Algorithm in Spherical Coordinates.

As in previous work a procedure devised by Mehra[3] was chosen to handle nonlinearities of the state-measurement transformation. In this procedure, the filtering is carried out in measurement variable coordinates where the observation transformation is linear.

The state of the system, $\underline{x} = (x_1 x_2 x_3 x_4 x_5 x_6)^T$, is transformed from rectangular coordinates used for the guidance problem to spherical coordinates, $\underline{y} = (R\lambda\beta\dot{R}\dot{\lambda}\dot{\beta})^T$ as defined in Figure 2. Measurements of, for example, range, range rate, and angles are a subset of the spherical coordinates. For this reason the y variables are called measurement variables.

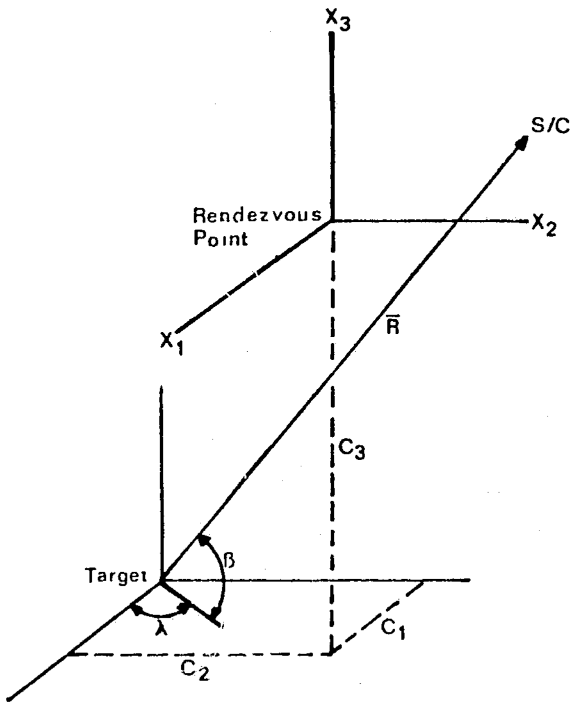


Figure 2. Transformation Geometry.

The transformation from rectangular coordinates, centered at the rendezvous point, to the measurement variables, $\underline{y} = g(\underline{x})$, is

$$\begin{aligned}
R &= [(x_1+C_1)^2 + (x_2+C_2)^2 + (x_3+C_3)^2]^{\frac{1}{2}} \\
\lambda &= \text{ARCTAN } [(x_2+C_2)/(x_1+C_1)] \\
\beta &= \text{ARCTAN}\left[\frac{x_3 + C_3}{[(x_1+C_1)^2 + (x_2+C_2)^2]^{\frac{1}{2}}}\right] \\
\dot{R} &= [(x_1+C_1)x_4 + (x_2+C_2)x_5 + (x_3+C_3)x_6]/R \\
\dot{\lambda} &= [(x_1+C_1)x_5 - (x_2+C_2)x_4]/[(x_1+C_1)^2 + (x_2+C_2)^2] \\
\dot{\beta} &= \frac{[(x_1+C_1)^2+(x_2+C_2)^2]x_6 - (x_3+C_3)[(x_1+C_1)x_4+(x_2+C_2)x_5]}{[(x_1+C_1)^2 + (x_2+C_2)^2]^{\frac{1}{2}} R^2}
\end{aligned} \tag{3}$$

The inverse transformation, $\underline{x} = \ell(\underline{y})$ is

$$\begin{aligned}
x_1 &= R \cos \lambda \cos \beta - C_1 \\
x_2 &= R \cos \beta \sin \lambda - C_2 \\
x_3 &= R \sin \beta - C_3 \\
x_4 &= \dot{R} \cos \lambda \cos \beta - R\dot{\beta} \sin \beta \sin \lambda + R\dot{\lambda} \cos \beta \sin \lambda \\
x_5 &= \dot{R} \cos \beta \sin \lambda - R\dot{\beta} \sin \beta \sin \lambda + R\dot{\lambda} \cos\beta \cos\lambda \\
x_6 &= \dot{R} \sin \beta + R\dot{\beta} \cos \beta
\end{aligned} \tag{4}$$

It was assumed that the most general set of measurements that can be made is range, angles λ and β, and range rate. It was assumed that angular rates $\dot{\lambda}$ and $\dot{\beta}$ cannot be measured directly.

The filter process proceeds as follows. Starting with an estimate $\hat{\underline{x}}_{k/k}$ at time T_k, and the spirit of the linear Kalman theory, a state estimate $\hat{\underline{x}}_{k+1/k}$ at time T_{k+1} is obtained by a linear extrapolation through the state transition matrix for the linearized system

$$\hat{\underline{x}}_{k+1/k} = \phi_{k+1/k} \hat{\underline{x}}_{k/k} \tag{5}$$

where

$$\phi_{k+1/k} = \begin{bmatrix} \alpha_{k+1/k} & 0 & 0 & \beta_{k+1/k} & 0 & 0 \\ 0 & \alpha_{k+1/k} & 0 & 0 & \beta_{k+1/k} & 0 \\ 0 & 0 & \alpha_{k+1/k} & 0 & 0 & \beta_{k+1/k} \\ \gamma_{k+1/k} & 0 & 0 & \delta_{k+1/k} & 0 & 0 \\ 0 & \gamma_{k+1/k} & 0 & 0 & \delta_{k+1/k} & 0 \\ 0 & 0 & \gamma_{k+1/k} & 0 & 0 & \delta_{k+1/k} \end{bmatrix} \tag{6}$$

and,

$$\begin{aligned} \alpha_{k+1/k} &= \frac{\tau_{k+1/k}}{\tau_{k/k}} \left[3\left(\frac{\tau_{k+1/k}}{\tau_{k/k}}\right)-2\left(\frac{\tau_{k+1/k}}{\tau_{k/k}}\right)^2\right] \\ \beta_{k+1/k} &= \tau_{k+1/k} \left[\frac{\tau_{k+1/k}}{\tau_{k/k}} - \left(\frac{\tau_{k+1/k}}{\tau_{k/k}}\right)^2\right] \\ \gamma_{k+1/k} &= -\frac{6}{\tau_{k/k}} \left[\left(\frac{\tau_{k+1/k}}{\tau_{k/k}}\right) - \left(\frac{\tau_{k+1/k}}{\tau_{k/k}}\right)^2\right] \\ \delta_{k+1/k} &= 3\left(\frac{\tau_{k+1/k}}{\tau_{k/k}}\right) - 2\left(\frac{\tau_{k+1/k}}{\tau_{k/k}}\right)^2 \end{aligned} \tag{7}$$

with time to go given by

$$\begin{aligned} \tau_{k/k} &= T_F - T_k \\ \tau_{k+1/k} &= T_F - T_{k+1} \end{aligned} \tag{8}$$

Another method used to form the first state estimate $\hat{x}_{k+1/k}$ is to directly integrate (fourth-order Runge-Kutta) the state estimate

$$\hat{\underline{x}}_{k+1/k} = \hat{\underline{x}}_{k/k} + \int_{T_k}^{T_{k+1}} \hat{\dot{\underline{x}}}\, dt \tag{9}$$

With new computing equipment, this integration can be done onboard.

After the first estimate $\underline{x}_{k+1/k}$ is formed, we then transfer to measurement variables with Equations (3) in the form

$$\underline{y}_{k+1/k} = g\ (\underline{x}_{k+1/k}) \qquad \text{(nonlinear)} \tag{10}$$

The best estimate of the measurement variables at T_{k+1} is then given by Kalman's relation

$$\hat{\underline{y}}_{k+1/k+1} = \hat{\underline{y}}_{k+1/k}\ K_{k+1/k}\ (\underline{z}_{k+1} - H\hat{\underline{y}}_{k+1/k}) \tag{11}$$

where H is a rectangular matrix of ones and zeros that picks from $\hat{\underline{y}}_{k+1/k}$ those elements that correspond to the actual measurements, which, in this case, is

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{12}$$

$\underline{z}_{k+1}$ is the actual measurement vector, and $K_{k+1/k}$ is the Kalman gain matrix (yet to be calculated). After filtering, transformation is effected back to rectangular coordinates using Equations (4) in the form

$$\underline{x}_{k+1/k+1} = \ell(\underline{y}_{k+1/k+1}) \qquad \text{(nonlinear)} \tag{13}$$

The Kalman gain is calculated by

$$K_{k+1/k} = M_{k+1/k}\ H^T (HM_{k+1/k}\ H^T + R_{k+1})^{-1} \tag{14}$$

where R_{k+1} is a square diagonal matrix of the variances of the measurement errors, and $M_{k+1/k}$ is a matrix consisting of the transferred covariance of measurement variables calculated by

$$M_{k+1/k} = \psi_{k+k/k}\ M_{k/k}\ \psi^T{}_{k+1/k} \tag{15}$$

where $M_{k/k}$ is the measurement variables covariance matrix at T_k and $\psi_{k+1/k}$ is an equivalent "transition matrix" for the measurement variables; i.e.,

$$\hat{\underline{y}}_{k+1/k} \simeq \psi_{k+1/k}\ \hat{\underline{y}}_{k/k} \tag{16}$$

Mehra observed that ψ can be constructed by calculating

$$\psi_{k+1/k} = \left(\frac{\partial \underline{y}_{k+1/k}}{\partial \underline{y}_{k/k}}\right)\left(\frac{\partial \underline{y}_{k/k}}{\partial \underline{x}_{k/k}}\right)\left(\frac{\partial \underline{x}_{k/k}}{\partial \underline{y}_{k+1/k}}\right)$$

or

$$\psi_{k+1/k} = \left(\frac{\partial g}{\partial \underline{x}}\right)_{k+1/k}\ \phi_{k+1/k}\ \left(\frac{\partial \ell}{\partial \underline{y}}\right)_{k/k} \tag{17}$$

The matrices $(\partial \ell/\partial \underline{y})_{k/k}$ and $\phi_{k+1/k}$ are available from the best estimate of state at $T_{k/k}$, and $(\partial g/\partial \hat{\underline{x}})_{k+1/k}$ is formed by using the first estimate at $T_{k+1/k}$. The matrices $(\partial \ell/\partial \hat{\underline{y}})$ and $(\partial g/\partial \hat{\underline{x}})$, in columns, are

$$
\begin{gathered}
\left(\frac{\partial \ell}{\partial y_1}\right) = \begin{bmatrix} \cos\lambda \ \cos\beta \\ \sin\lambda \ \cos\beta \\ \sin\beta \\ -\dot{\beta} \ \cos\lambda \ \sin\beta - \dot{\lambda}\sin\lambda \ \cos\beta \\ \dot{\lambda} \ \cos \lambda\cos \ \beta - \dot{\beta}\sin\lambda \ \sin\beta \\ \dot{\beta} \ \cos \ \beta \end{bmatrix} \\
\left(\frac{\partial \ell}{\partial y_2}\right) = \begin{bmatrix} -R \ \sin\lambda \ \cos\beta \\ R \ \cos\lambda \ \cos\beta \\ 0 \\ -\dot{R} \ \sin\lambda \ \cos\beta + R\dot{\beta} \ \sin\lambda \ \sin\beta - R\dot{\lambda} \ \cos\lambda \ \cos\beta \\ -\dot{R} \ \cos\lambda \ \cos\beta - R\dot{\lambda} \ \sin\lambda \ \cos\beta - R\dot{\beta} \ \cos\lambda \ \sin\beta \\ 0 \end{bmatrix} \\
\left(\frac{\partial \ell}{\partial y_3}\right) = \begin{bmatrix} -R \ \cos\lambda \ \sin\beta \\ -R \ \sin\lambda \ \sin\beta \\ R \ \cos\beta \\ -\dot{R} \ \cos\lambda \ \sin\beta - R\dot{\beta} \ \cos\lambda \ \cos\beta + R\dot{\lambda} \ \sin\lambda \ \sin\beta \\ -\dot{R} \ \sin\lambda \ \sin\beta - R\dot{\lambda} \ \cos\lambda \ \sin\beta - R\dot{\beta} \ \sin\lambda \ \cos\beta \\ \dot{R} \ \cos \ \beta - R\dot{\beta} \ \sin\beta \end{bmatrix} \\
\left(\frac{\partial \ell}{\partial y_4}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \cos\lambda \ \cos\beta \\ \sin\lambda \ \cos\beta \\ \sin\beta \end{bmatrix} \\
\left(\frac{\partial \ell}{\partial y_5}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -R \ \sin\lambda \ \cos\beta \\ R \ \cos\lambda \ \cos\beta \\ 0 \end{bmatrix} \\
\left(\frac{\partial \ell}{\partial y_6}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -R \ \cos\lambda \ \sin\beta \\ -R \ \sin\lambda \ \sin\beta \\ R \ \cos\beta \end{bmatrix}
\end{gathered} \tag{18}
$$

$$\left(\frac{\partial g}{\partial x_1}\right) = \begin{bmatrix} P_1/R \\ -P_2/Q^2 \\ -(P_1P_3)/(QR^2) \\ (P_4R^2-VP_1)/R^3 \\ (P_5Q^2-2WP_1)Q^4 \\ (P_1P_6-P_3P_4)/(QR^2) - (2QP_1P_6)/R^4+(2AP_1P_3)/(QR^4)+(P_1P_3A)/(Q^3R^2) \end{bmatrix}$$

$$\left(\frac{\partial g}{\partial x_2}\right) = \begin{bmatrix} P_2/R \\ P_1/Q^2 \\ -(P_2P_3)/(QR^2) \\ (R^2P_5-VP_2)/R^3 \\ -(P_4Q^2+2WP_2)/Q^4 \\ (P_2P_6-P_3P_5)/(QR^2)-(2QP_2P_6)/R + (2AP_2P_3)/(QR^4)+(P_2P_3A)/(Q^3R^2) \end{bmatrix}$$

$$\left(\frac{\partial g}{\partial x_3}\right) = \begin{bmatrix} P_3/R \\ 0 \\ Q/R^2 \\ (P_6R^2-VP_3)/R^3 \\ 0 \\ -A/(QR^2)-(2QP_3P_6)/R^4+(2AP_3{}^2)/(QR^4) \end{bmatrix}$$

$$\left(\frac{\partial g}{\partial x_4}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ P_1/R \\ -P_2/Q^2 \\ -(P_1P_3)/(QR^2) \end{bmatrix} \qquad (19)$$

$$\left(\frac{\partial g}{\partial x_5}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ P_2/R \\ P_1/Q^2 \\ -(P_2P_3)/(QR^2) \end{bmatrix}$$

$$\left(\frac{\partial g}{\partial x_6}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ P_3/R \\ 0 \\ Q/R^2 \end{bmatrix}$$

where

$$
\begin{aligned}
P_1 &= x_1 + C_1 & Q &= [P_1^{\,2} + P_2^{\,2}]^{\frac{1}{2}} \\
P_2 &= x_2 + C_2 & R &= [P_1^{\,2} + P_2^{\,2} + P_3^{\,2}]^{\frac{1}{2}} \\
P_3 &= x_3 + C_3 & A &= P_1P_4 + P_2P_5 \\
P_4 &= \dot{x}_1 = x_4 & V &= P_1P_4 + P_2P_5 + P_3P_6 \\
P_5 &= \dot{x}_2 = x_5 & W &= P_1P_5 - P_2P_4 \\
P_6 &= \dot{x}_3 = x_6 & &
\end{aligned}
\tag{20}
$$

All that remains is to propagate the covariance to time T_{k+1} and this is done by the relation

$$M_{k+1/k+1} = (I - K_{k+1/k}\, H)\, M_{k+1/k} \tag{21}$$

2.3 The Modified Guidance and Navigation Scheme

The procedure is illustrated in the block diagram in Figure 3. Starting at time $T_{k/k}$ an estimate of the state $\hat{\underline{x}}_{k/k}$ is presumed available for evaluation. The exact state $\underline{x}_k$ is also specified at this time. The estimate $\underline{x}_{k/k}$ is put into the guidance law to generate the thrusting required over the ensuing guidance interval. The full equations of motion of the comet are then integrated accurately to a time T_{k+1} and the result is then transformed to measurement variables and approximate noise added to simulate actual measurements $\underline{z}_{k+1}$. To represent onboard computations, the state $\hat{\underline{x}}_{k/k}$ is propagated to time T_{k+1} through the transition matrix $\phi_{k+1/k}$ (or integrated as shown by dotted line). The nonlinear transformation $g(\underline{x})$ to measurement variables is then made to give a first estimate $\hat{\underline{y}}_{k+1/k}$. The filtered estimate $\hat{\underline{y}}_{k+1/k+1}$ is then transformed nonlinearly by $\ell(\underline{y})$ to obtain the new state estimate $\hat{\underline{x}}_{k+1/k+1}$ and the process is repeated.

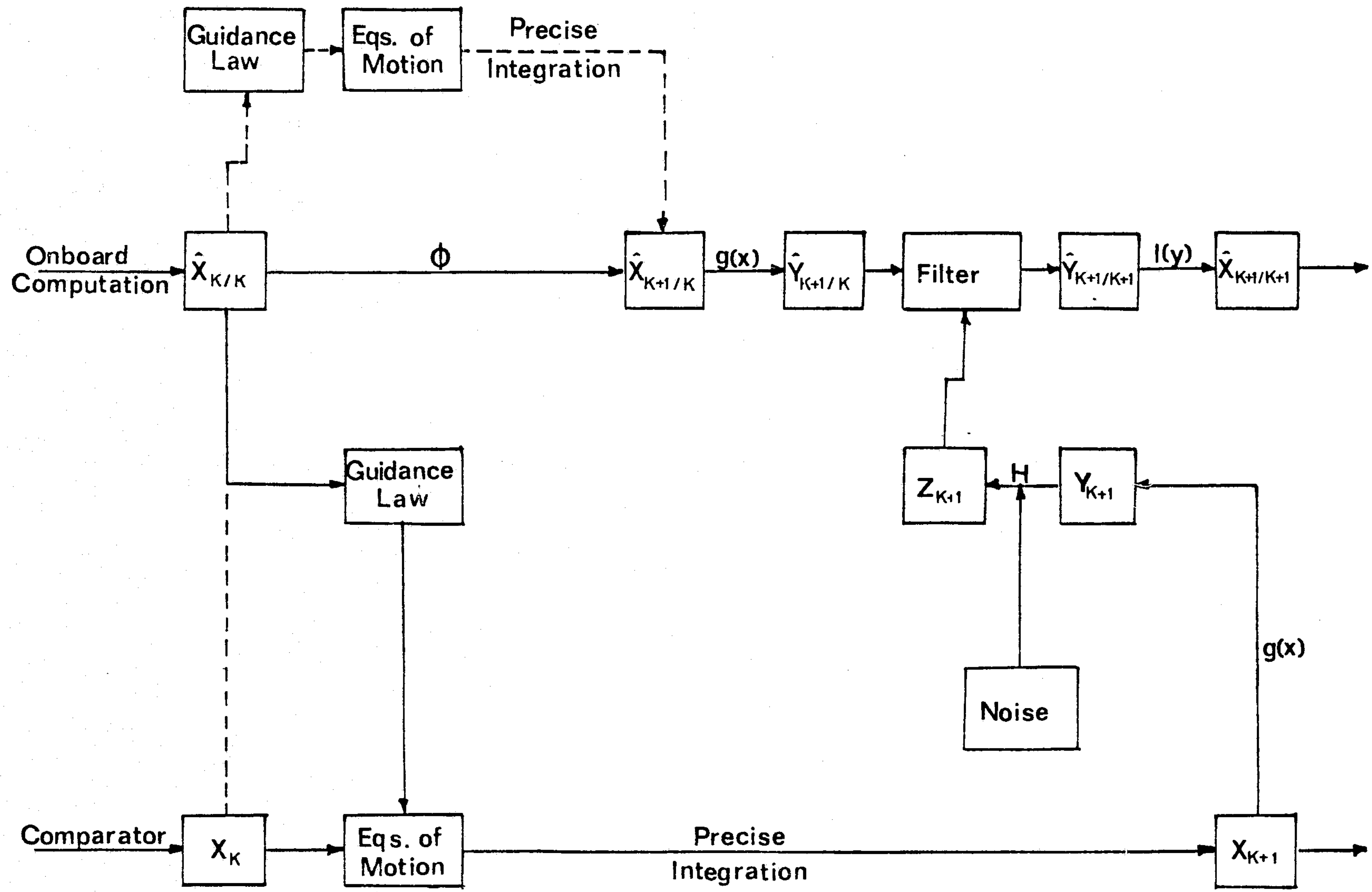


Figure 3. Evaluation Scheme.

3.0 ALGORITHM DEVELOPMENT

Having completed the reformulation in spherical coordinates, a second task is to reinvestigate the divergence problem. Proper choice of statistical quantities employed to set the filter is necessary, and work done to provide this proper choice is outlined in Section 3.1.

Other approaches to improvement of system performance through trajectory biasing and end point control are described in Sections 3.2 and 3.3.

3.1 Investigation of Filter Divergence

The Kalman filter theoretically produces an increasingly accurate estimate as additional data is processed. However, under actual operating conditions, error levels in the Kalman filter are often higher than predicted by theory. Errors can, in fact, increase continuously although additional data is being processed. No general way of handling this problem is available and approximations based on ideas from the theory and ad hoc procedures are the rule and this is the approach we take. To insure that the set of measurements employed are sufficient to reconstruct all states, system observability is investigated first.

3.1.1 System Observability

A system is observable if all states can be reconstructed from the set of available measurements. For the equations

$$\begin{aligned} \underline{y}_{k+1/k} &= \psi_{k+1/k}\, \underline{y}_{k/k} \\ \underline{z}_{k+1} &= H\, \underline{y}_{k+1} \end{aligned} \tag{22}$$

an augmented matrix derived by Kalman[4]

$$F = (H^T, \psi^T H^T, (\psi^T)^2 H^T, \ldots (\psi^T)^{n-1} H^T) \tag{23}$$

is formed. If the rank of the matrix F is n, where n is the order of the y vector, the system is observable.

A computer program written by Bullock and Fosha[5] is used to form the matrix F. The matrix is normalized at each step $(H^T, \psi^T H^T, \ldots)$ so that each column has unit length. The complete matrix is renormalized so each row has unit length (The normalization procedure prevents overflow in large problems). Then the square matrix FF^T is computed. If FF^T is nonsingular, the system is observable.

Since ψ is a function of time, the matrix F is also a function of time. Therefore, F must be computed at each guidance interval (or any multiple) because a new F is present at each step, and the degree of observability can change.

3.1.2 State Covariance Weighting Criteria

In nonlinear systems, the state error covariance matrix tends to reduce rapidly with respect to the actual state indeterminancies. This can be attributed to the fact that the filter believes the first measurement is highly accurate when it is not, and state estimates are biased incorrectly. To account for the reduction of the covariance matrix, a weighting procedure is employed.

Introducing state disturbance into the system, equation (16) becomes

$$\underline{y}_{k+1/k} = \psi_{k+1/k}\, \underline{y}_{k/k} + \Gamma\, \underline{w}_k \tag{24}$$

where Γ is the disturbance transition matrix, assumed to be identity, and $\underline{w}_k$ is the disturbance vector. The process $\{\underline{w}_k, k=0,1,\ldots\}$ is a Gaussian White sequence for which

$$E\,[\underline{w}_k] = 0 \;,\; h = 1,2,\ldots \tag{25}$$

Defining the positive definite matrix

$$Q_k = E[\underline{w}_k\;\underline{w}_k^{\,T}] \tag{26}$$

It can be shown (See, for example, Meditch[6])that the state error covariance matrix is given by the relation

$$M_{k+1/k} = \psi_{k+1/k}\; M_{k/k}\; \psi^T_{k+1/k} + Q_k \tag{27}$$

This weighting criteria aids in eliminating the severe reduction of the state covariance matrix and keeps it from becoming non-positive definite. However, proper weighting of measurements is not guaranteed because Q cannot be accurately determined, and errors can still exist.

3.1.3 Approximation of the Initial State Error Covariance Matrix

In a nonlinear system, good state knowledge at time k does not insure good knowledge of future state because of nonlinearities in the dynamical equations. Hence, the Kalman gain must weight the measurements sufficiently to compensate for the errors in the one-step propagation. Measurements are not weighted enough initially when the filter is initialized with state error covariance matrices that are too small. To aleviate this problem, a design procedure for determining a suitable initial state error covariance matrix is introduced.

Assuming a priori information for the range of values of the Kalman gain, a suitable initial state error covariance can be found eliminating the incorrect weighting of measurements. The only elements that are directly adjustable in the Kalman gain matrix are those which lie on the diagonal of the 6x4 gain matrix (to be shown later; these adjustable elements correspond directly to the first four diagonal elements of the state covariance matrix because of the special form of the observation matrix (discussed previously). Therefore, for given values of the Kalman gain, an initial state covariance matrix (four diagonal elements) can be found.

There exists a one-to-one relationship between the diagonal elements of the Kalman gain matrix and the first four diagonal elements of the state covariance matrix. A change in one element of the state covariance matrix results in a direct change of the corresponding element in the Kalman gain. This can be shown by performing the matrix operations. By precalculation, ψ is found to be approximately an identity matrix, so Equation (27) can be written

$$M_{k+1/k} \simeq M_{k/k} + Q_k \tag{28}$$

where $M_{k+1/k}$ is

$$M_{k+1/k} = \begin{bmatrix} M_{11} & & & & & \\ & M_{22} & & & & \\ & & M_{33} & & & \\ & & & M_{44} & & \\ & & & & M_{55} & \\ & & & & & M_{66} \end{bmatrix}$$

The Kalman gain relationship is given by

$$K_{k+1/k} = M_{k+1/k} H^T (HM_{k+1/k}H^T + R_{k+1})^{-1} \tag{29}$$

We write out Equation (29) in explicit matrix form,

$$\begin{bmatrix} K_{11} & & & 0 \\ & K_{22} & & \\ & & K_{33} & \\ 0 & & & K_{44} \end{bmatrix}_{6x4} = \begin{bmatrix} M_{11} & & & 0 \\ & M_{22} & & \\ & & M_{33} & \\ 0 & & & M_{44} \end{bmatrix}_{6x4} \begin{bmatrix} M_{11}+R_{11} & & & 0 \\ & M_{12}+R_{22} & & \\ & & M_{33}+R_{33} & \\ 0 & & & M_{44}+R_{44} \end{bmatrix}^{-1}_{4x4} \qquad (30)$$

Notice that only the first four diagonal elements of the state covariance matrix have any direct effect on the Kalman gain matrix because the elements M_{55} and M_{66} were eliminated in the calculations.

The Kalman gain matrix can be reduced to a 4x4 matrix since only the diagonal elements are considered. The diagonal elements of the 4x6 observation matrix correspond directly with the adjustable elements in the gain matrix, so H is taken to be a 4x4 identity matrix. The state covariance matrix is also reduced to a 4x4 matrix since only four elements can be directly obtained.

Now, an initial covariance matrix can be found insuring accurate weights for the measurements. Rewriting Equation (29), the Kalman gain is

$$K_{k+1/k} = M_{k+1/k} \left(M_{k+1/k} + R_{k+1}\right)^{-1} \qquad (31)$$

Elementary matrix manipulations of Equation (31) result in

$$M_{k+1/k} = \left(I - K_{k+1/k}\right)^{-1} K_{k+1/k} R_{k+1} \qquad (32)$$

Since ψ is approximately an identity matrix, the initial state covariance matrix can be calculated by

$$M_{k/k} \approx M_{k+1/k} - Q_k \tag{33}$$

The formulation calculates only those diagonal elements in the state covariance matrix which correspond to the actual measurements. But this is sufficient because the other diagonal elements in the state covariance matrix do not affect the diagonal elements in the Kalman gain matrix. However, the other diagonal elements do affect the off-diagonal terms in the $\psi M_{k/k} \psi^T$ calculation which, in turn, affects the fifth and sixth rows of the Kalman gain matrix. Therefore, the other diagonal elements must be chosen carefully. But this can only be done through numerical experiments.

With this formulation, a range of values for the state covariance matrix corresponding to the values of the Kalman gain matrix can be calculated, and, through simulation, the desired state covariance matrix can be determined. However, since the unmeasured states are not directly weighted, divergence can still occur.

3.2 Trajectory Biasing

In the comet rendezvous problem, the optimal control algorithm tends to fly the spacecraft directly toward the desired rendezvous point. If the rendezvous point lies directly between the comet and spacecraft, as illustrated in Figure 4, the measured angles change very little in the initial stages of flight and can cause poor estimation of the angular rates (unmeasured states). Poor estimates of the angular rates can cause the velocity of the spacecraft to be in error and accurate rendezvous cannot be attained. To eliminate this problem, trajectory biasing is investigated.

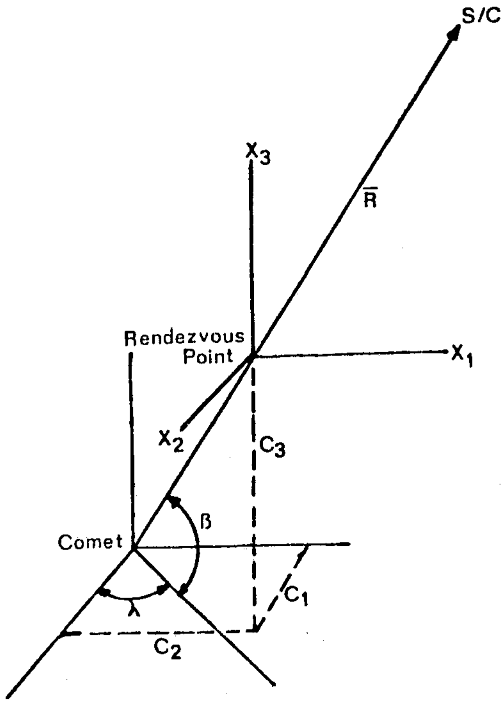


Figure 4. Possible Rendezvous Geometry.

Instead of flying directly to the desired rendezvous point, the spacecraft initially flies toward a point biased away from the target point. This point can be changed at each guidance step (or some multiple) until the biased point becomes the desired rendezvous point. Biasing the trajectory of the spacecraft in this manner gives larger angular change and, consequently, better estimation of angular rates.

With a better estimate of the unmeasured states, velocity estimates are more accurate, and more accurate rendezvous can be attained.

3.3 End point Control

The guidance algorithm singularity leads to a "blow up" of the procedure at rendezvous unless appropriate steps are taken. One approach would be to "freeze" the commands near rendezvous, but this is not a good way because after the rendezvous is accomplished, it must be maintained. That is, station keeping must be done. For this reason we looked for a modification that would transform the terminal approach algorithm into a station keeping algorithm while avoiding the singularity. One easy way to do this is simply to push time to go a predetermined amount as rendezvous is approached. A successive increase of time to go by a predetermined amount of time at each guidance step after a specified point will keep time to go large enough to avoid the singularity and automatically turns the approach algorithm into a station keeping algorithm. For deterministic runs and evaluations with very high precision data this worked well when implemented in the form of an increase of one guidance time interval at each succeeding step starting at the time of one interval before rendezvous. However, for cases of only moderately accurate data, this procedure still led to thrusts above those easily attainable with SEP (about 10^{-4}g). Several other methods of reducing end point thrust were considered, and a particularly simple one was found to work well. Instead of pushing one step, we chose an increase in time to go equal to

$$\left|\frac{F}{10^{-4}}\right| \Delta$$

Where,

Δ = guidance step size

10^{-4} = maximum SEP thrust (g's)

F = value of thrust in unmodified next interval.

Here, the symbol $|A|$ means round A to the next higher integer. Note that there are extensions of this technique that could be applied early in the terminal approach should such be required. However, this extension was not incorporated in the computer program as it was not necessary for the cases considered.

4.0 ALGORITHM EVALUATION

The full guidance and navigation algorithm of Section 2.0 was programmed for simulation on an IBM 370/150 digital computer. The simulation package includes a double precision, fourth-order Runge-Kutta integrator to provide an accurate comparison trajectory for evaluation of algorithm performance. A brief discussion of the program and user instructions are included in the Appendix.

4.1 Development Simulations

The starting point was chosen to be five days before rendezvous at a distance of 50,000 km from the target and a relative velocity of 20,000 km/day. Measurements are made and navigation computations made each one-tenth of a day. The rendezvous point is located 1732 km from the target with 1000 km displacement in each coordinate direction. In the first simulations, the initial state error covariance was assumed to be a diagonal matrix with terms of the order of 10^5 in units of km and days. This matrix is an identity matrix in the units of 10^5 km for distance and days for time as used in the computer program. A five percent error in each coordinate of the true initial state was chosen as the estimated position and velocity of spacecraft.

Simulations were made and gross filter divergence was found as shown in Figure 5. Terminal rendezvous errors were about 2000 km and 1,500 km/day (17.4 m/sec). Divergence began approximately four days from rendezvous. At this point, system observability was checked with the method of Section 3.1.1 to determine if the four measurements were

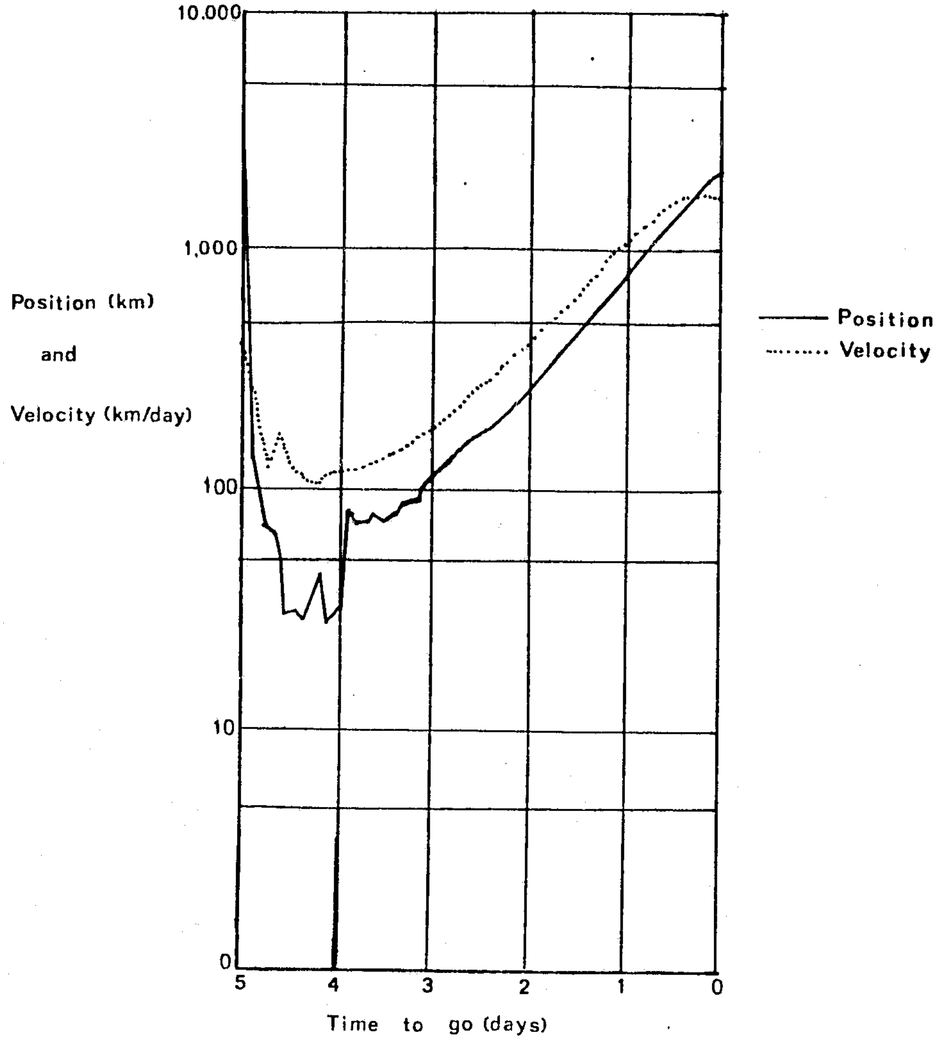


Figure 5. Navigation Errors for Initial Simulations.

sufficient to reconstruct all six states. The observability criteria was applied with solutions showing complete system observability throughout the mission. Hence, methods for correcting divergence were investigated.

One cause of divergence was the rapid reduction of the state error covariance matrix. In fact, simulations show the covariance matrix became numerically zero in the computer near rendezvous and completely ignored new measurements as they were made. A standard fix is to insert noise into the basic system equations as discussed in Section 3.1.2. A constant matrix Q was added to the state error covariance matrix to approximate this noise. Q was arbitrarily chosen diagonal with elements of the order of magnitude of 10. The magnitude of the diagonal elements in Q are approximately the same size as those in the state error covariance matrix when divergence started.

Figure 6 shows a marked improvement in the navigation errors with the addition of the Q matrix. Errors at rendezvous were approximately 11.5 km and 310 km/day (3.59 m/sec) with the spacecraft a distance of 28 km from rendezvous.

Continuing the divergence investigation the method of Section 3.1.3 was employed to estimate initial covariance. Assuming desired values for the diagonal elements of the Kalman gain matrix of about .95, an initial state error covariance matrix was found. Interestingly, results were very close to the previously assumed identity matrix and we therefore continued to use the identity matrix.

The principal errors were in the velocity estimates. To get a better angular data; that is, data with larger changes, the trajectory of the spacecraft was biased away from the final target point following the idea of Section 3.2. The new target point was chosen as a function of position by the equation

$$C_1 = \{[\hat{x}_1^2 + \hat{x}_2^2 + \hat{x}_3^2]^{\frac{1}{2}}/1.5\} + 1000$$

where C_1 is the standoff distance in the x_1 direction. The point changes each guidance step, and, upon a forced command, becomes the

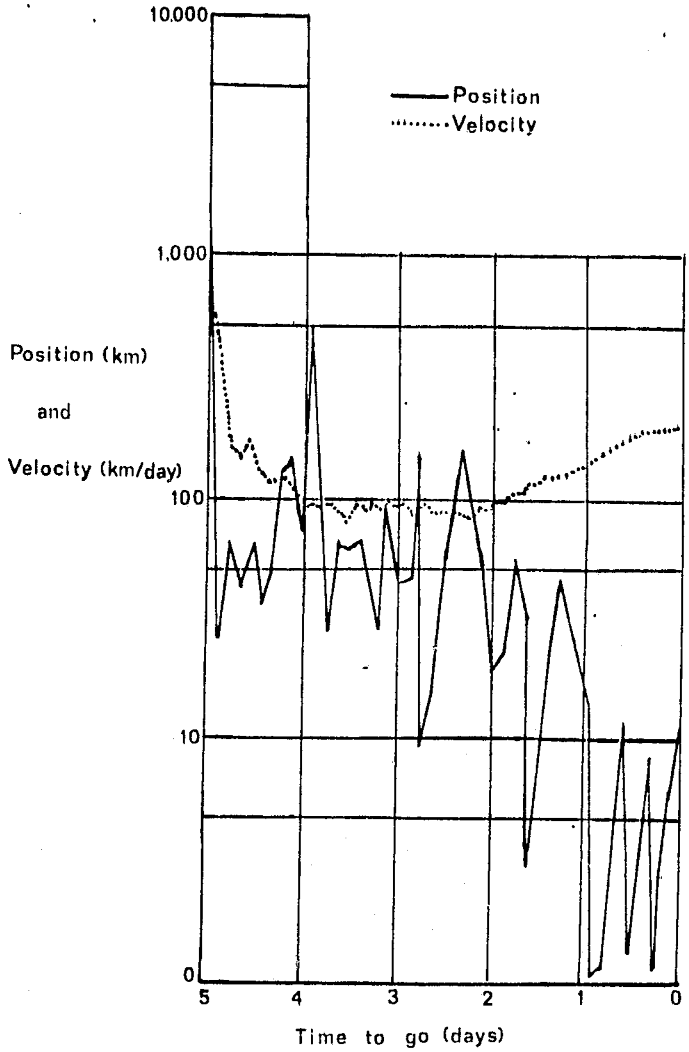


Figure 6. Navigation Errors with Q Matrix Added.

desired rendezvous point at time zero. Biasing in this manner allows the angles to change more in the early stages of flight and thus gives a more accurate estimate of the angular rates. Figure 7 shows the

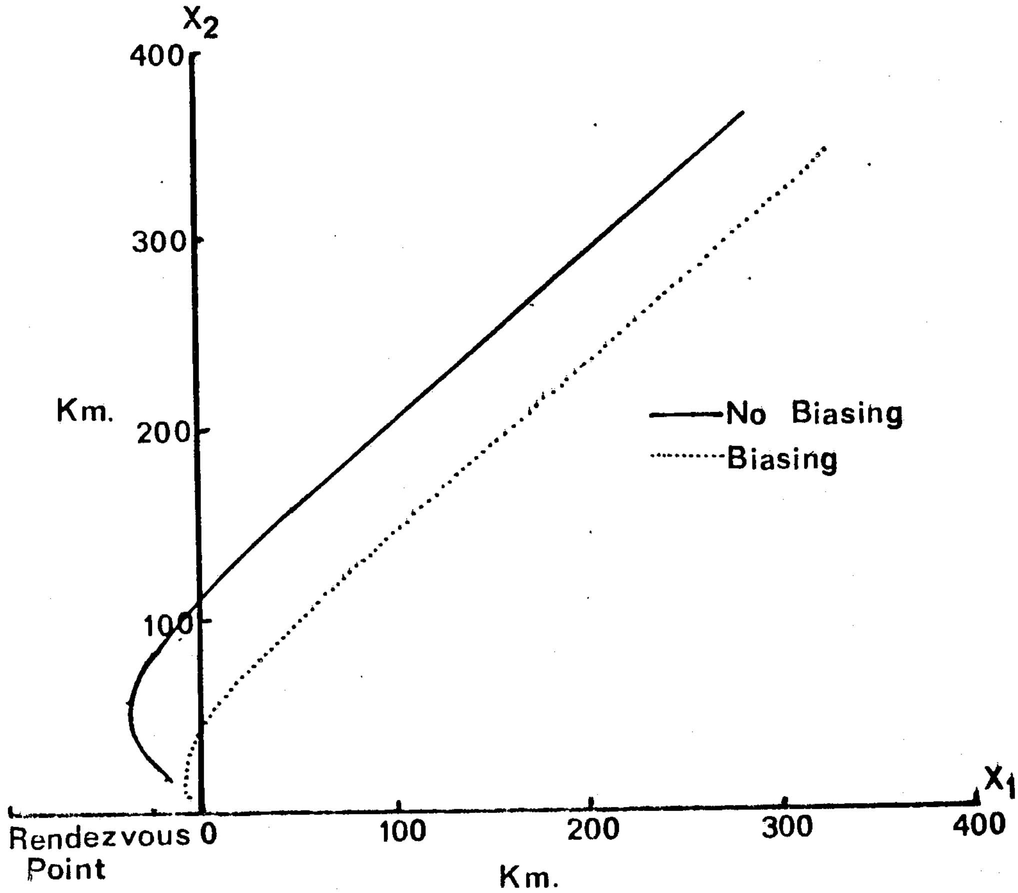


Figure 7. Comparison of Trajectories with and without Biasing within 1/2 Day to Go.

terminal geometry of typical biased and unbiased trajectories. A considerable improvement in accuracy was obtained. The improvement was sufficient to warrant undertaking the parametric study of the effects of measurement accuracy. Before conducting this study, the end point control scheme of Section 3.3 was incorporated into the program.

4.2 Generation of Parametric Data

Table 1 presents nine of the data error sets employed in the parametric study. The choice of error in range measurement standard deviation, σ_R, of 0.3 percent and 3.0 percent of range was made to bracket expected errors with realistic radar equipment. Specific information on what these errors might be is not available and these values were chosen after discussions with NASA radar electronics personnel. The choice of standard deviation of range rate measurements, $\sigma_{\dot{R}}$, of 100 km/day and 1,000 km/day (1.16 m/sec and 11.6 m/sec), was based on these same discussions. The standard deviation of 1.0 arc seconds on angle measurements was also a best estimate of what could be done with on board equipment. This value is three times larger than the value used by JPL for use of ground based data reduction of spacecraft TV.

TABLE 1. DATA SETS FOR PARAMETRIC STUDY
(Note: 1,000 km/day ≃ 11.6 m/sec.)

Data Set	Range, R	Angles λ & β (arc min.)	Range Rate, $\dot{R}$ (km/day)
A	.03156 R	1.08	10
B	↓	↓	100
C	.03 R	1.0	100
D	↓	↓	1,000
E	↓	↓	10,000
F	↓	↓	100,000
G	.003	↓	10
H	↓	↓	100
J	↓	↓	1,000

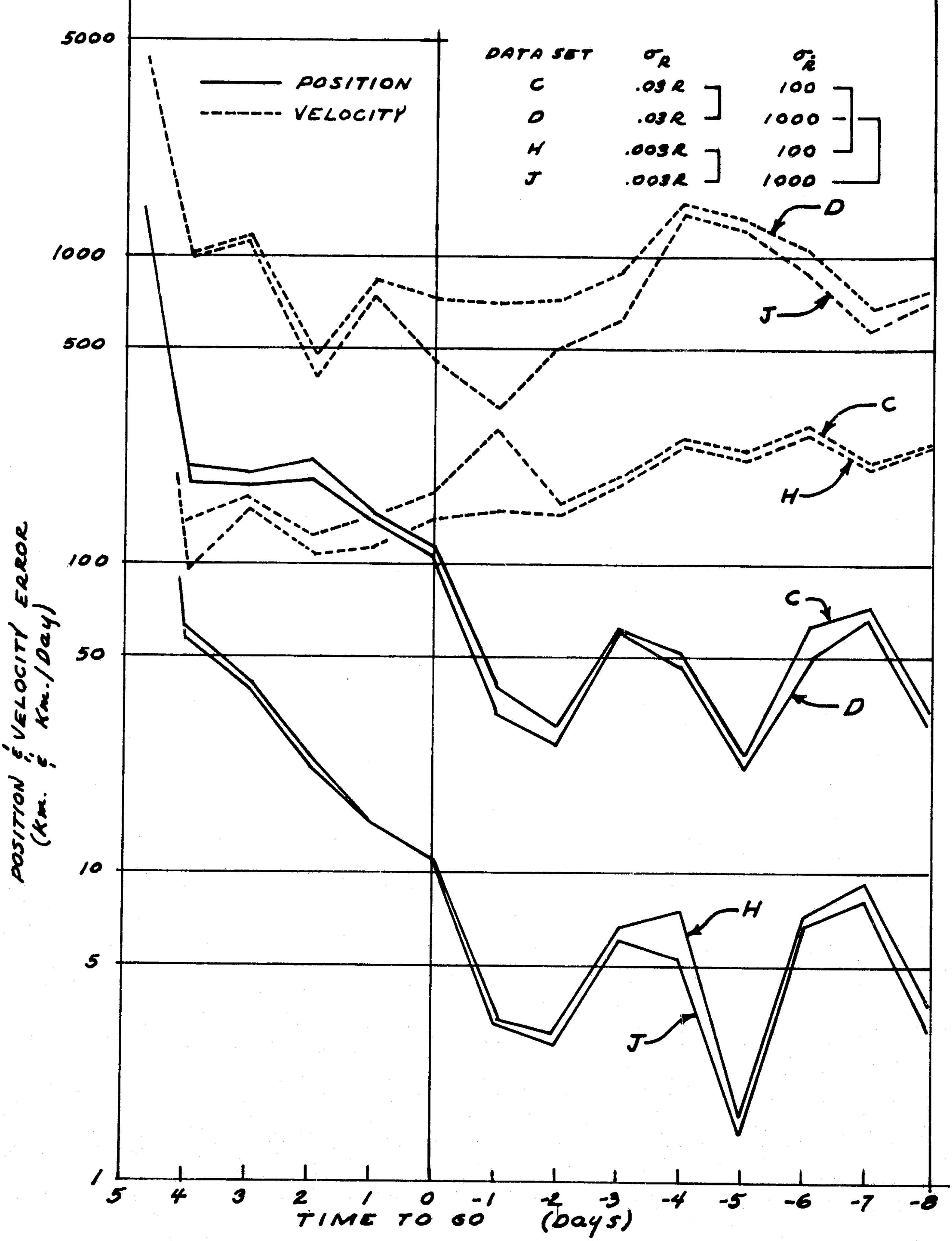
5.0 RESULTS

The parametric runs made with the data sets of Table 1 gave no cases of divergence even with the extreme range rate errors of sets E and F. And station was maintained after rendezvous for as long a period as run (up to 30 days after rendezvous). The results indicate that the rate measurements may not be required, but this requires additional investigation.

Results of four representative sets, C, D, H and J are shown in Figure 8. These sets bracket expected measurement errors as discussed in Section 4.2. In the samples, the same four sequences of random numbers was used to simulate the four measurements made (range, range-rate, and two angles) Other sequences gave results differing only in detail not accuracy levels. The plots give position and velocity errors only at intervals of one day because plots with the one-tenth day computation steps used in the simulations are cluttered and difficult to interpret. The values at each day were obtained by averaging over an interval about each plotted point.

One point is immediately obvious from Figure 8: The magnitude of the error in position depends almost completely on the accuracy of range measurements and the magnitude of the error in velocity depends almost completely on the accuracy of range rate measurement. For example, consider sets C and D. These sets have the same range measurement standard deviation (.03 R), but different rate standard deviation (100 km/day and 1000 km/day). The position errors for these two cases are very nearly the same, but the velocity errors differ by a factor

FIG. 8 ERRORS VS. TIME TO GO



of three to four. Similarly, sets C and H show essentially the same velocity errors and a difference in position error by a factor of nine or ten.

We are led to two conclusions applicable in the range of measurement errors considered: First, that reduction of range rate error by a factor of ten leads to an improvement in velocity estimation by a factor of three or four and has little effect on accuracy of position estimation. Second, that reduction in range error by a factor of ten leads to an improvement in position estimation by a factor of nine or ten and has little effect on accuracy of velocity estimation. The plots in Figure 8 show these effects add linearity. Stated another way: An improvement in range rate measurement accuracy produces only about one-third the improvement in velocity estimate while an improvement in range measurement accuracy leads to about the same improvement in position estimate. Note that the linearity of the relation for velocity breaks down and (fortunately) very large rate measurement errors do not lead to large velocity estimate errors. Range is clearly the more valuable data type.

Examination of the velocity error curves of Figure 8 shows what might seem to be a slight divergence after the nominal rendezvous time. However, the reduction in accuracy is caused by the elimination of trajectory biasing shortly before zero time to go. The ensuing motion is a cyclic motion near the rendezvous point. This is reflected in the position error curves. An appropriate biasing of the trajectory and thrust levels can undoubtedly smooth this out. This would be a point for study only later in guidance and navigation scheme development. As it is, the errors are not large and do not grow with time.

The work presented here has lost much of its sense of immediacy because of changes in the overall NASA space program. Missions to comets or other deep space bodies which will require autonomous navigation and guidance are a long way in the future. When the time comes, the results here may help to give a starting point for the detailed investigations and development that will then be required.

REFERENCES

1. Abercrombie, G. E., "Two Guidance Algorithms for Rendezvous Missions to Comets and Asteroids," Interim Report, Prepared under Contract NAS8-27664, Auburn University, December, 1972.

2. Bennett, A. G., "Guidance and Navigation Studies," Final Report, Part I, Prepared under Contract NAS8-27664, Auburn University, December, 1973.

3. Mehra, R. K., "A Comparison of Several Nonlinear Filters for Reentry Vehicle Tracking," IEEE Trans. on Auto Control, Vol. AC-16, No. 4, August, 1971, pp. 307-319.

4. Kalman, R. E., "On the General Theory of Control Systems," Proc. First Intern. Cong. IFAC, Moscow, 1960.

5. Bullock, T. E., and Fosha, C. E., "A General Purpose Fortran Program for Estimation, Control, and Simulation," Proceedings of the Eighth Annual IEEE Region II Convention, Huntsville, Alabama, November, 1969.

6. Meditch, J. S., Stochastic Optimal Linear Estimation and Control, McGraw-Hill, New York (1969), pp. 171-174.

7. Jazwinski, A. H., Stochastic Processes and Filtering Theory, Academic Press, New York (1970), pp. 301-305.

APPENDIX

COMPUTER SIMULATION PROGRAM

The computer program used is written in FORTRAN IV and used with the IBM 370/155. The program is a research tool, not a production routine. The steps in the simulaiton and the names of the subroutines that carry out these steps are as follows.

A fourth order Runge-Kutta subroutine, RUNKUT, is used to integrate the dynamic forces in GOFX$\not{0}$ over subintervals of length DT. At each time step (DELT), observations are made in OBSERV and the filter is used to predict the state in FILTER. The information generated is transferred to subroutine CYCLOT and terminal conditions are checked. Program sequencing and execution is controlled by subroutine CYCLE. Subroutine TARGET is used to generate the comet's position. NOISE is a dummy name for the functions URAND (Uniformly Distributed Random Numbers) and GRAND (Gaussian Distributed Random Numbers). Ten independent noise channels are shared by these two functions.

A. Subroutine Names and Descriptions

MAIN	Reads in system data and calls CYCLE.
CYCLE	Controls sequence of operation and transfer of data between XT (true state) and XP (predicted state).
RUNKUT	Fourth order Runge-Kutta integrator. Dynamics are provided by DOFX$\not{0}$ and guidance by GOFX$\not{0}$. Called by CYCLE. Performs N integrations of step size DT at each call.

Entries:

RKINIT called by CYCLE
Initialize internal variables and read in XT

DOFX∅ - Compute contribution of dynamics to true $\dot{\underline{x}}$. J is the index of the components of XT.

Entries:

DOFX∅: Compute data to be used by all components.

DOFX: Compute each component of the true state vector.

DXINIT: Initialize internal constants and read in comet data.

SPECIAL: Calls target for comet position.

GOEFX∅: - Same as DOEF∅ except XP is used as variable.

FILTER - User supplied algorithm to calculate XP. (Basic Extended Kalman filter used in present listing).

Entries:

FLINIT: Used to initialize arrays.

SPECIAL: Calls MINV, CONOBS, and can call INTGR.

OBSERV - Generates Z; may call GRAND.

CYCLOT - Outputs data and checks for end of run.

Entries:

TERMIN: Check for end conditions to be satisfied.

RECAP: If end conditions met outputs minimum normed distance, velocity, and associated times.

CYINIT: Initialize internal constants.

TARGET - Comet's position by solution of Kepler's equation.

MINV - Gaussian elimination inversion routine.

CONOBS - Determines degree of observability

SPECIAL - Calls MTRANS, MMUL, MXINV, and NORML

MATRIX — Subroutine name for group of entries.

Entries:

MMUL: Performs a special matrix multiplication for CONOBS.

MTRANS: Transposes matrices for CONOBS.

MXINV: Calls DINVER

NORML: Normalizes matrices for CONOBS.

DINVER - Matrix inversion routine using pivital condensation for determination of determinants used in CONOBS.

INTGR - Same as RUNKUT except uses XP instead of XT.

GRAND - Generates Gaussian distributed random noise with given mean (RMEAN) and standard deviation (STDDEV).

LSCNT - Noise channel number. Calls URAND.

URAND - Generates random numbers over the interval [o,1].

BLOCK DATA - Initializes seed numbers for URAND.

B. Variable Names and Definitions

XT - True state vector.

XP - Predicted state vector (loaded in GXINIT).

XE - Error in state.

Z - True observations.

ZP - Predicted observations.

ZE - Error in observations (residuals)

LØL - One BYTE logical array used to control sequencing of simulator.

LØE - One BYTE Logical array for use by error monitor (not implemented).

DT - Integration stepsize (true position).

DELT - Guidance update stepsize (predicted position).

N - DELT/DT (an integer).

TI - Integration start time.

TF - Integration end time.

ISLEN - Number of elements in state vector.

LOLEN - Number of elements in the observation vector.

C - Rendezvous stand-off distance.

A - Target semi-major axis.

RN - Target mean motion.

EPS - Target eccentricity.

EO - Target eccentric anomaly.

TSTAR - Guidance initiation time.

C. Input Data

Card Number	Contents	Format
1	Title Card	SA8
2	Run time logical flags (LØL)	80L1
3	Run time error flags (LØE)	80L1
4	N, ISLEN, IOLEN	8I3
5	TI, TF, DT, DELT	8F10.0
6	XT (Initial conditions)	8F10.0
7	A, RN, EPS, EO, TSTAR	8F10.0
8	C	8F10.0

D. Subroutine Initialization Entries

Subroutines and Line Numbers	Entry
GX INIT(J)	
38,39,40	Initial position error
41,42,43	Initial position error
FLINIT	
238-243	Initial state error covariance matrix
OVINIT	
90	Standard deviation of angular measurement error (EANG)
91, 92	Fix logical if statements for RMIN and RDMIN
OBSERV	
30-33	Mean, standard deviation and noise channel.
24-27	Range, angles, and range rate errors (RFRAF)

PROGRAM LISTING

MAIN

```
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL*1 L$L,L$E,LMON,LF,LT
      COMMON/V$RBLE/XT(6),XP(6),XE(6),Z(6),ZP(6),ZE(6)
      COMMON/T$MER/DELT,DT,TIME,TI,TF,N,ISLEN,IOLEN
      COMMON/SY$TEM/L$L(40),L$E(10)
      COMMON/M$NITR/LMON(20)
      COMMON/NOI$E/IRAN(10),DG(10),RFRAF(6)
      COMMON/OFFSET/C(3)
      COMMON/MOOCH/T6(4,4)
      DIMENSION B(4,6),RN(4,4)
      DATA LF,LT/F,T/
  502 FORMAT(80L1)
  503 FORMAT(5A8)
  504 FORMAT(8I3)
  505 FORMAT(8F10.0)
  602 FORMAT(1H0,80L1)
  603 FORMAT(1H0,5A8)
  604 FORMAT(1H0,'  INPUT CARD LIST')
  606 FORMAT(1H ,////)
  607 FORMAT(1H0,8I5)
  608 FORMAT(1H0,1P6D12.5)
      WRITE(6,604)
      READ(5,503)TITLE
      WRITE(6,603)TITLE
      READ(5,502)L$L
      WRITE(6,602)L$L
      READ(5,502)L$E
      WRITE(6,602)L$E
      READ(5,504)N,ISLEN,IOLEN
      WRITE(6,607)N,ISLEN,IOLEN
      READ(5,505)TI,TF,DT,DELT
      WRITE(6,608)TI,TF,DT,DELT
      WRITE(6,606)
      CALL CYCLE
      STOP
      END
      SUBROUTINE CYCLE
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL*1 L$L,L$E,LMON,LF,LT
      COMMON/V$RBLE/XT(6),XP(6),XE(6),Z(6),ZP(6),ZE(6)
      COMMON/T$MER/DELT,DT,TIME,TI,TF,N,ISLEN,IOLEN
      COMMON/SY$TEM/L$L(40),L$E(10)
      COMMON/M$NITR/LMON(20)
      COMMON/NOI$E/IRAN(10),DG(10),RFRAF(6)
      COMMON/OFFSET/C(3)
      COMMON/MOOCH/T6(4,4)
      DATA LF,LT/F,T/
      DIMENSION B(4,6),RN(4,4)
C    **** INITIALIZE SUBROUTINES ****
      CALL RKINIT
      DUM=DXINIT(1)
```

```
      DUM=GXINIT(1)
      DUM=URINIT(1)
      CALL CYINIT
      CALL FLINIT
      CALL OVINIT
      CALL INTINI
      DUM=DXININ(1)
      DUM=GXININ(1)
C     **** COMPUTE TRAJECTORIES ****
    1 CALL RUNKUT
      IF(L$L(2))GO TO 2
      CALL OBSERV
      CALL FILTER
      GO TO 3
    2 DO 4 I=1,ISLEN
    4 XP(I)=XT(I)
C     **** OUTPUT CYCLE DATA ****
    3 CALL CYCLOT
C     **** MONITOR SECTION ****
      CALL TERMIN
      IF(L$L(2))GO TO 5
      IF(L$L(3))GO TO 5
      IF(L$L(6))GO TO 1
      GO TO 7
    5 DO 6 I=1,ISLEN
    6 XP(I)=XT(I)
      IF(L$L(6))GO TO 1
C     **** OUTPUT SECTION ****
    7 CALL RECAP
      RETURN
      END
      SUBROUTINE RUNKUT
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL*1 L$L,L$E,LMON,LF,LT
      LOGICAL*1 LS1,LS2
      COMMON/V$RBLE/XT(6),XP(6),XE(6),Z(6),ZP(6),ZE(6)
      COMMON/T$MER/DELT,DT,TIME,TI,TF,N,ISLEN,IOLEN
      COMMON/SY$TEM/L$L(40),L$E(10)
      COMMON/M$NITR/LMON(20)
      COMMON/NOI$E/IRAN(10),DG(10),RFRAF(6)
      COMMON/OFFSET/C(3)
      DIMENSION XINT(6),SUM(6)
      DATA LF,LT/F,T/
      L$L(12)=LT
      YS=DSQRT(XP(1)**2+XP(2)**2+XP(3)**2)
      C(1)=1.0D-02+(YS/1.5D0)
      IF(TIME.GT.38.4) C(1)=1.0D-02
      WRITE(6,750)C
  750 FORMAT(1H0,7X,'C1',11X,'C2',11X,'C3'/4X,1P3D12.5/)
      DO 1 ICYCLE=1,N
      DO 33 I=1,ISLEN
```

```
 33 SUM(I)=0.0D0
    L$L(10)=LT
    DO 10 I1=1,4
    LS1=I1.EQ.2. OR.I1.EQ.3
    LS2=I1.EQ.4
    L$L(11)=I1.EQ.3
    F=F1
    FS=F5
    IF(LS1)F=F2
    IF(LS1)FS=F3
    IF(LS2)FS=F4
    TS=TIME+DT*FS
    DO 20 I=1,ISLEN
20  XINT(I)=XT(I)+FS*XINT(I)
    DO 31 I=1,NP1
    J=I-1
    IF(J.GT.0) GO TO 2
    DUM=DOFX$(J,TS)
    DUM=GOFX$(J,TS)
    L$L(12)=LF
    GO TO 31
2   XINT(J)=DT*(DOFX(J)+GOFX(J))
    SUM(J)=SUM(J)+F*XINT(J)
31  CONTINUE
    L$L(10)=LF
10  CONTINUE
    TIME=TIME+DT
    DO 11 I=1,ISLEN
11  XT(I)=XT(I)+SUM(I)
  1 CONTINUE
    RETURN
    ENTRY RKINIT
    READ(5,501)(XT(I),I=1,ISLEN)
    WRITE(6,601)(XT(I),I=1,ISLEN)
501 FORMAT(8F10.0)
601 FORMAT(1H0,1P8D12.5)
    F1=1.0D0/6.0D0
    F2=2.0D0*F1
    F3=1.0D0/2.0D0
    F4=1.0D0
    F5=0.0D0
    TIME=TI
    NP1=ISLEN+1
    DO 32 I=1,ISLEN
 32 XINT(I)=XT(I)
    RETURN
    END
    FUNCTION DOFX$(J,TS)
    IMPLICIT REAL*8(A-H,O-Z)
    LOGICAL*1 L$L,L$E,LMON,LF,LT
    COMMON/V$RBLE/XT(6),XP(6),XE(6),Z(6),ZP(6),ZE(6)
```

```
      COMMON/T$MER/DELT,DT,TIME,TI,TF,N,ISLEN,IOLEN
      COMMON/SY$TEM/L$L(40),L$E(10)
      COMMON/M$NITR/LMON(20)
      COMMON/NOI$E/IRAN(10),DG(10),RFRAF(6)
      COMMON/VAR1/A,RN,EPS,EO,TSTAR,DET
      COMMON/OFFSET/C(3)
      DIMENSION D(3),S(3)
      DATA LF,LT/F,T/
99601 FORMAT(1H0,I4,'IMPROPER INDEX *DOFX')
      IF(.NOT.L$L(11))CALL TARGET(S,TS)
      S2=0.0D0
      D2=0.0D0
      DO 1 I=1,3
      D(I)=S(I)+C(I)+XT(I)
      D2=D2+D(I)**2
 1    S2=S2+S(I)**2
      DN=DSQRT(D2)
      SN=DSQRT(S2)
      RATI=GM/(SN*SN*SN)
      RAT2=(SN/DN)**3
      DOFX$=0.0D0
      IF(TF-TIME.GT.DELT)RETURN
      IF(.NOT.L$L(10))RETURN
      WRITE(6,602)TIME,XT
  602 FORMAT(1H0,'END STATE ',2X,'TIME=',F10.3/1H ,1P6D12.5)
      RETURN
      ENTRY DOFX(J)
      GO TO (99999,99999,99999,99998,99998,99998),J
      WRITE(6,99601)J
      DOFX=0.0D0
      L$E(2)=LT
      RETURN
99999 DOFX=XT(J+3)
      RETURN
99998 DOFX=RATI*(S(J-3)-D(J-3)*RAT2)
      RETURN
      ENTRY DXINIT(J)
      READ(5,501)A,RN,EPS,EO,TSTAR
      EO=RN*TSTAR
      WRITE(6,601)A,RN,EPS,EO,TSTAR
  501 FORMAT(8F10.0)
  601 FORMAT(1H0,1P8D12.5)
      READ(5,501)C
      WRITE(6,601)C
      GM=9.90549D05
      DET=1.0D-3
      DXINIT=0.0D0
      RETURN
      END
      FUNCTION GOFX$(J,TS)
      IMPLICIT REAL*8(A-H,O-Z)
```

```
      LOGICAL*1 L$L,L$E,LMON,LF,LT
      COMMON/V$RBLE/XT(6),XP(6),XE(6),Z(6),ZP(6),ZE(6)
      COMMON/T$MER/DELT,DT,TIME,TI,TF,N,ISLEN,IOLEN
      COMMON/SY$TEM/L$L(40),L$E(10)
      COMMON/M$NITR/LMON(20)
      COMMON/NOI$E/IRAN(10),DG(10),RFRAF(6)
      COMMON/FORCE/F(3)
      COMMON/OFFSET/C(3)
      COMMON/XPROP/XPNEW(6)
      DATA LF,LT/F,T/
      DIMENSION XERR(6)
99601 FORMAT(1H0,I4,'IMPROPER INDEX *GOFX')
      IF(L$L(12))TIM1=TS
      TAU0=TF-TIM1
      TAU=TF-TS
      TRAT1=(6.0D0/TAU0**2)*(1.0D0-2.0D0*(TAU/TAU0))
      TRAT2=(2.0D0/TAU0)*(1.0D0-3.0D0*(TAU/TAU0))
      GOFX$=0.0D0
      RETURN
      ENTRY GOFX(J)
      GO TO (99999,99999,99999,99998,99998,99998),J
      WRITE(6,99601)J
      L$E(3)=LT
      GOFX=0.0D0
      RETURN
99999 GOFX=0.0D0
      RETURN
99998 F(J-3)=TRAT1*XP(J-3)+TRAT2*XP(J)
      GOFX=F(J-3)
      RETURN
      ENTRY GXINIT(J)
      WRITE(6,500)(XT(I),I=1,6)
  500 FORMAT(1H1,3X,'XTRUE INITIALLY'/4X,1P6D12.5/)
      XERR(1)=2.0D-02
      XERR(2)=1.5D-02
      XERR(3)=1.0D-02
      XERR(4)=7.0D-03
      XERR(5)=5.0D-03
      XERR(6)=4.0D-04
      WRITE(6,501)(XERR(I),I=1,6)
  501 FORMAT(1H0,3X,'INITIAL ERROR ON X'/4X,1P6D12.5/)
      DO 90001 I=1,ISLEN
90001 XP(I)=XT(I)+XERR(I)
      WRITE(6,776)(XP(I),I=1,ISLEN)
  776 FORMAT(1H0,3X,'XHAT INITIALLY'/4X,1P6D12.5/)
      DO 25 I=1,6
   25 XPNEW(I)=XP(I)
      GXINIT=0.0D0
      RETURN
      END
      SUBROUTINE FILTER
```

```
      IMPLICIT REAL*8(A-H,O-Z)
      REAL G,BQ,PSI
      LOGICAL*1 L$L,L$E,LMON,LF,LT
      COMMON/V$RBLE/XT(6),XP(6),XE(6),Z(6),ZP(6),ZE(6)
      COMMON/T$MER/DELT,DT,TIME,TI,TF,N,ISLEN,IOLEN
      COMMON/SY$TEM/L$L(40),L$E(10)
      COMMON/M$NITR/LMON(20)
      COMMON/NOI$E/IRAN(10),DG(10),RFRAF(6)
      COMMON/KALMAN/RM(6,6),FILT(6,4),U(6),RN(4,4),B(4,6)
      COMMON/OFFSET/C(3)
      COMMON/MOOCH/T6(4,4)
      COMMON/XPROP/XPNEW(6)
      COMMON/YTRUE/YTRUE(6)
      DIMENSION T1(6),YL(6),Y(6),T2(6,6),T3(6,6),T4(6,6),QP(
     *6,6),
     $T5(6,6),PHI(6,6)
      DIMENSION DY(6),QEXT(6,6),RU(1),DDY(6)
      DIMENSION BQ(4,6),PSI(6,6)
      DIMENSION G(6,1)
      DIMENSION LL(6),MM(6)
      DATA LF,LT/F,T/
      TAUK1=TF-TIME
      TAUK=TAUK1+DELT
      RAT=TAUK1/TAUK
      RAT2=RAT*RAT
      RAT3=RAT2*RAT
      A=3.0D0*RAT2-2.0D0*RAT3
      DIF=RAT-RAT2
      F=TAUK1*DIF
      D=3.0D0*RAT2-2.0D0*RAT
      E=-6.0D0*DIF/TAUK
      DO 31 I=1,3
      PHI(I,I)=A
      PHI(I,I+3)=F
      PHI(I+3,I)=E
      PHI(I+3,I+3)=D
   31 CONTINUE
C    PREDICT X STATE
      IF(L$L(9)) GO TO 8000
      DO 10 I=1,ISLEN
      T1(I)=0.0D0
      DO 10 J=1,ISLEN
   10 T1(I)=T1(I)+PHI(I,J)*XP(J)
      GO TO 8002
 8000 CONTINUE
      CALL INTGR
      DO 8001 I=1,ISLEN
 8001 T1(I)=XP(I)
 8002 CONTINUE
C    TRANSFORM TO Y SYSTEM
      P1=T1(1)+C(1)
```

```
      P2=T1(2)+C(2)
      P3=T1(3)+C(3)
      Q=DSQRT(P1**2+P2**2)
      R=DSQRT(P1**2+P2**2+P3**2)
      AU=P1*T1(4)+P2*T1(5)
      V=AU+P3*T1(6)
      W=((P1)*(T1(5)))-((P2)*(T1(4)))
      Y(1)=R
      Y(2)=DATAN2(P2,P1)
      Y(3)=DATAN2(P3,Q)
      Y(4)=V/R
      Y(5)=W/(Q**2)
      Y(6)=(Q**2*T1(6)-AU*P3)/(Q*R**2)
      CL=DCOS(Y(2))
      CB=DCOS(Y(3))
      SL=DSIN(Y(2))
      SB=DSIN(Y(3))
C     COMPUTE LEFT PARTIAL DERIVATIVE
      T2(1,1)=CL*CB
      T2(2,1)=SL*CB
      T2(3,1)=SB
      T2(4,1)=-Y(6)*CL*SB-Y(5)*SL*CB
      T2(5,1)=Y(5)*CL*CB-Y(6)*SL*SB
      T2(6,1)=Y(6)*CB
      T2(1,2)=-Y(1)*SL*CB
      T2(2,2)=Y(1)*CL*CB
      T2(4,2)=-Y(4)*SL*CB+Y(1)*Y(6)*SB*SL-Y(1)*Y(5)*CL*CB
      T2(5,2)=Y(4)*CL*CB-Y(1)*Y(5)*SL*CB-Y(1)*Y(6)*CL*SB
      T2(1,3)=-Y(1)*CL*SB
      T2(2,3)=-Y(1)*SL*SB
      T2(3,3)=Y(1)*CB
      T2(4,3)=-Y(4)*CL*SB-Y(1)*Y(6)*CL*CB+Y(1)*Y(5)*SL*SB
      T2(5,3)=-Y(4)*SL*SB-Y(1)*Y(5)*CL*SB-Y(1)*Y(6)*SL*CB
      T2(6,3)=Y(4)*CB-Y(1)*Y(6)*SB
      T2(4,4)= CB*CL
      T2(5,4)=SL*CB
      T2(6,4)=SB
      T2(4,5)=-Y(1)*SL*CB
      T2(5,5)=Y(1)*CL*CB
      T2(4,6)=-Y(1)*CL*SB
      T2(5,6)=-Y(1)*SL*SB
      T2(6,6)=Y(1)*CB
C     COMPUTE RIGHT PARTIAL DERIVATIVE
      T3(1,1)=P1/R
      T3(1,2)=P2/R
      T3(1,3)=P3/R
      T3(2,1)=-P2/(Q**2)
      T3(2,2)=P1/(Q**2)
      T3(3,1)=-P1*P3/(Q*R**2)
      T3(3,2)=-P2*P3/(Q*R**2)
      T3(3,3)=Q/(R**2)
```

```
      T3(4,1)=(T1(4)*R**2-V*P1)/(R**3)
      T3(4,2)=(R**2*T1(5)-V*P2)/(R**3)
      T3(4,3)=(T1(6)*R**2-V*P3)/(R**3)
      T3(4,4)=P1/R
      T3(4,5)=P2/R
      T3(4,6)=P3/R
      T3(5,1)=(T1(5)/(Q**2))-(2.*W*P1)/(Q**4)
      T3(5,2)=(-T1(4)/(Q**2))-(2.*W*P2)/(Q**4)
      T3(5,4)=-P2/(Q**2)
      T3(5,5)=P1/(Q**2)
      T3(6,1)=(P1*T1(6)-P3*T1(4))/(Q*R**2)-2.*Q*P1*T1(6)/(R*
     **4)+
     $2.*AU*P1*P3/(Q*R**4)+P1*P3*AU/(Q**3*R**2)
      T3(6,2)=(P2*T1(6)-P3*T1(5))/(Q*R**2)-2.*Q*P2*T1(6)/(R*
     **4)+
     $2.*AU*P2*P3/(Q*R**4)+AU*P2*P3/(Q**3*R**2)
      T3(6,3)=-AU/(Q*R**2)-2.*P3*T1(6)*Q/(R**4)+2.*AU*P3**2/
     $(Q*R**4)
      T3(6,4)=-P1*P3/(Q*R**2)
      T3(6,5)=-P2*P3/(Q*R**2)
      T3(6,6)=Q/(R**2)
C    COMPUTE PSI
      DO 12 I=1,ISLEN
      DO 12 J=1,ISLEN
      T4(I,J)=0.0D0
      DO 12 K=1,ISLEN
      DO 12 L=1,ISLEN
   12 T4(I,J)=T4(I,J)+T2(I,K)*PHI(K,L)*T3(L,J)
      DO 9990 I=1,6
      DO 9990 J=1,6
      G(I,1)=0.0
 9990 PSI(I,J)=T4(I,J)
      DO 9991 I=1,4
      DO 9991 J=1,6
 9991 BQ(I,J)=B(I,J)
C    PREDICT COVARIANCE
      DO 13 I=1,ISLEN
      DO 13 J=1,ISLEN
      T5(I,J)=QP(I,J)
      DO 13 K=1,ISLEN
      DO 13 L=1,ISLEN
   13 T5(I,J)=T5(I,J)+T4(I,K)*RM(K,L)*T4(J,L)
      DY(1)=-2.0D-05
      DY(2)=-4.1D-05
      DY(3)=-3.3D-05
      DY(4)=1.5D-05
      DY(5)=9.21D-05
      DY(6)=4.1D-05
      DO 600 I=1,6
      DO 600 J=1,6
  600 QEXT(I,J)=DY(I)*DY(J)
```

```
      DO 980 I=1,6
  980 QEXT(I,I)=QEXT(I,I)*1.0D6
      DO 601 I=1,6
      DO 601 J=1,6
  601 T5(I,J)=T5(I,J)+QEXT(I,J)
C     COMPUTE THE FILTER
      DO 14 I=1,IOLEN
      DO 14 J=1,IOLEN
      T6(I,J)=RN(I,J)
      DO 14 K=1,ISLEN
      DO 14 L=1,ISLEN
   14 T6(I,J)=T6(I,J)+B(I,K)*T5(K,L)*B(J,L)
      CALL MINV(T6,IOLEN,16,LL,MM,D)
      DO 15 I=1,ISLEN
      DO 15 J=1,IOLEN
      FILT(I,J)=0.0D0
      DO 15 K=1,ISLEN
      DO 15 L=1,IOLEN
   15 FILT(I,J)=FILT(I,J)+T5(I,K)*B(L,K)*T6(L,J)
C     COMPUTE PREDICTED OBSERVATIONS
      DO 16 I=1,IOLEN
      ZP(I)=0.0D0
      DO 16 J=1,ISLEN
   16 ZP(I)=ZP(I)+B(I,J)*Y(J)
C     COMPUTE THE ERROR IN OBSERVATIONS
      DO 17 I=1,IOLEN
   17 ZE(I)=Z(I)-ZP(I)
C     UPDATE Y
      DO 18 I=1,ISLEN
      T1(I)=Y(I)
      DO 18 J=1,IOLEN
   18 T1(I)=T1(I)+FILT(I,J)*ZE(J)
C     UPDATE COVARIANCE
      DO 19 I=1,ISLEN
      DO 19 J=1,ISLEN
      T4(I,J)=0.0D0
      DO 19 K=1,IOLEN
   19 T4(I,J)=T4(I,J)+FILT(I,K)*B(K,J)
      DO 20 I=1,ISLEN
      DO 20 J=1,ISLEN
      T4(I,J)=-T4(I,J)
      IF(I.EQ.J)T4(I,J)=T4(I,J)+1.0D0
   20 CONTINUE
      DO 21 I=1,ISLEN
      DO 21 J=1,ISLEN
      RM(I,J)=0.0D0
      DO 21 K=1,ISLEN
   21 RM(I,J)=RM(I,J)+T4(I,K)*T5(K,J)
   99 CONTINUE
C     SAVE Y(K+1,K+1)
      DO 22 I=1,ISLEN
```

```
   22 YL(I)=T1(I)
      CALL CONOBS(6,1,4,PSI,G,BQ)
C   CONVERT TO X
      CL=DCOS(YL(2))
      CB=DCOS(YL(3))
      SL=DSIN(YL(2))
      SB=DSIN(YL(3))
      XP(1)=YL(1)*CB*CL-C(1)
      XP(2)=YL(1)*CB*SL-C(2)
      XP(3)= YL(1)*SB-C(3)
      XP(4)=YL(4)*CB*CL-YL(1)*YL(6)*SB*CL-YL(1)*YL(5)*CB*SL
      XP(5)=YL(4)*CB*SL-YL(1)*YL(6)*SB*SL+YL(1)*YL(5)*CB*CL
      XP(6)=YL(4)*SB+YL(1)*YL(6)*CB
      WRITE(6,912)(XP(J),J=1,6)
  912 FORMAT(1H0,3X,'XHAT'/4X,1P6D12.5/)
      DO 913 I=1,6
  913 XPNEW(I)=XP(I)
      RETURN
      ENTRY FLINIT
     *              ENTRY
C    ZERO ARRAYS
      DO 101 I=1,ISLEN
      DO 102 J=1,ISLEN
      T2(I,J)=0.0D0
      T3(I,J)=0.0D0
      RM(I,J)=0.0D0
      PHI(I,J)=0.0D0
  102 QP(I,J)=0.0D0
      DO 101 J=1,IOLEN
  101 FILT(I,J)=0.0D0
C    INITIALIZE VARIABLES
      YL(1)=DSQRT((XP(1)+C(1))**2+(XP(2)+C(2))**2+(XP(3)+C(3
     *))**2)
      YL(2)=DATAN2((XP(2)+C(2)),(XP(1)+C(1)))
      ARG2=DSQRT((XP(1)+C(1))**2+(XP(2)+C(2))**2)
      YL(3)=DATAN2((XP(3)+C(3)),ARG2)
      YL(4)=((XP(1)+C(1))*XP(4)+(XP(2)+C(2))*XP(5)+(XP(3)+C(
     *3))*XP(6))
     $/YL(1)
      YL(5)=((XP(1)+C(1))*XP(5)-(XP(2)+C(2))*XP(4))/
     $((XP(1)+C(1))**2+(XP(2)+C(2))**2)
      YL(6)=(((XP(1)+C(1))**2+(XP(2)+C(2))**2)*XP(6)-(XP(3)+
     *C(3))*
     $((XP(1)+C(1))*XP(4)+(XP(2)+C(2))*XP(5)))/(YL(1)**2*ARG
     *2)
      RM(1,1)=1.0D0
      RM(2,2)=1.0D0
      RM(3,3)=1.0D0
      RM(4,4)=1.0D0
      RM(5,5)=1.0D0
      RM(6,6)=1.0D0
```

```
      WRITE(6,652)
  652 FORMAT(1H0,3X,'INITIAL COVARIANCE MATRIX')
      DO 650 I=1,6
  650 WRITE(6,651)(RM(I,J),J=1,6)
  651 FORMAT(1H0,4X,1P6D12.5)
      RETURN
      END
      SUBROUTINE OBSERV
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL*1 L$L,L$E,LMON,LF,LT
      COMMON/V$RBLE/XT(6),XP(6),XE(6),Z(6),ZP(6),ZE(6)
      COMMON/T$MER/DELT,DT,TIME,TI,TF,N,ISLEN,IOLEN
      COMMON/SY$TEM/L$L(40),L$E(10)
      COMMON/M$NITR/LMON(20)
      COMMON/NOI$E/IRAN(10),DG(10),RFRAF(6)
      COMMON/KALMAN/CRUD(66),RN(4,4),B(4,6)
      COMMON/OFFSET/C(3)
      COMMON/YTRUE/YTRUE(6)
      DIMENSION ZT(6)
      DATA LF,LT/F,T/
C  OBSERVATIONS
C     RFRAF(1 TO 4) = STDDEV FOR ZT(1 TO 4) RESPECTIVELY
      YTRUE(1)=DSQRT((XT(1)+C(1))**2+(XT(2)+C(2))**2+(XT(3)+
     *C(3))**2)
      RANGE=YTRUE(1)
      YTRUE(2)=DATAN2((XT(2)+C(2)),(XT(1)+C(1)))
      ARG1=DSQRT((XT(1)+C(1))**2+(XT(2)+C(2))**2)
      YTRUE(3)=DATAN2((XT(3)+C(3)),ARG1)
      YTRUE(4)=((XT(1)+C(1))*XT(4)+(XT(2)+C(2))*XT(5)+(XT(3)
     *+C(3))*XT(6
     $))/YTRUE(1)
      YTRUE(5)=((XT(1)+C(1))*XT(5)-(XT(2)+C(2))*XT(4))/(ARG1
     ***2)
      YTRUE(6)=((ARG1**2)*XT(6)-(XT(3)+C(3))*((XT(1)+C(1))*X
     *T(4)
     $+(XT(2)+C(2))*XT(5)))/(YTRUE(1)**2*ARG1)
      RFRAF(1)=DABS(YTRUE(1)*3.0D-03)
      RFRAF(2)=EANG
      RFRAF(3)=EANG
      RFRAF(4)=1.0D-02
      DG(1)=GRAND(0.0D0,RFRAF(1),5)
      DG(2)=GRAND(0.0D0,RFRAF(2),2)
      DG(3)=GRAND(0.0D0,RFRAF(3),3)
      DG(4)=GRAND(0.0D0,RFRAF(4),8)
      IF(RANGE.GT.RMIN)GO TO 2
      IF(RANGE.LT.RMIN.AND.RANGE.GT.RDMIN)GO TO 3
      IF(RANGE.LT.RDMIN.AND.RANGE.GT.RMIN) GO TO 6
      IF(RANGE.LT.RDMIN)GO TO 8
    2 WRITE(6,1)(DG(LSCNT),LSCNT=2,3)
      IOLEN=2
      GO TO 4
```

```
    3 IOLEN=3
      WRITE(6,1)(DG(LSCNT),LSCNT=1,IOLEN)
    1 FORMAT(///,10X,'NOISE',5X,1P4D12.5)
      GO TO 4
    6 WRITE(6,1)(DG(LSCNT),LSCNT=2,4)
      IOLEN=3
      GO TO 4
    8 IOLEN=4
      WRITE(6,1)(DG(LSCNT),LSCNT=1,IOLEN)
    4 CONTINUE
      DO 38 I=1,IOLEN
      DO 32 J=1,IOLEN
   32 RN(I,J)=0.0D0
      DO 38 J=1,ISLEN
   38 B(I,J)=0.0D0
      IF(IOLEN.EQ.2)GO TO 34
      IF(L$L(20).AND.IOLEN.EQ.3)GO TO 37
      DO 33 I=1,IOLEN
   33 B(I,I)=1.0D0
      GO TO 36
   37 B(1,2)=1.0D0
      B(2,3)=1.0D0
      B(3,4)=1.0D0
      DO 39 I=1,3
   39 RN(I,I)=RFRAF(I+1)**2
      GO TO 35
   34 B(1,2)=1.0D0
      B(2,3)=1.0D0
      RN(1,1)=RFRAF(2)**2
      RN(2,2)=RFRAF(3)**2
      GO TO 35
   36 DO 30 I=1,IOLEN
   30 RN(I,I)=RFRAF(I)**2
   35 CONTINUE
      DO 41 I=1,IOLEN
      ZT(I)=0.0D0
      DO 41 J=1,ISLEN
   41 ZT(I)=ZT(I)+B(I,J)*YTRUE(J)
      DO 43 I=1,IOLEN
   43 Z(I)=ZT(I)+DG(I)
      RETURN
      ENTRY OVINIT
     *                ENTRY
      A=0.0D0
      EANG=2.906D-04
      RMIN=1.0D20
      RDMIN=1.0D20
      RETURN
      END
      FUNCTION GRAND(RMEAN,STDDEV,ISLCT)
      IMPLICIT REAL*8(A-H,O-Z)
```

```
C    PURPOSE
C       COMPUTES A NORMALLY DISTRIBUTED RANDOM NUMBER WITH A G
       *IVEN
C       MEAN AND STANDARD DEVIATION
        A=0.0D0
        DO 50 I=1,12
     50 A=A+URAND(ISLCT)
        GRAND=(A-6.0D0)*STDDEV+RMEAN
        RETURN
        END
        FUNCTION URAND(ISLCT)
        IMPLICIT REAL*8(A-H,O-Z)
        COMMON/NOI$E/IRAN(10),DG(10),RFRAF(6)
        IY=IRAN(ISLCT)*65539
        IF(IY)5,6,6
      5 IY=IY+2147483647+1
      6 URAND=DFLOAT(IY)*4.656613D-10
        IRAN(ISLCT)=IY
        RETURN
        ENTRY URINIT(ISLCT)
       *             ENTRY
        URAND=0.0D0
        URINIT=0.0D0
        RETURN
        END
        BLOCK DATA
        IMPLICIT REAL*8 (A-H,O-Z)
        COMMON/NOI$E/IRAN(10),DG(10),RFRAF(6)
        DATA IRAN/69800661,54218059,51070625,15239339,75892237
       *,
       *10418327,81767867,59847821,52031357,26256073/
        END
        SUBROUTINE CYCLOT
        IMPLICIT REAL*8(A-H,O-Z)
        REAL GUTL
        LOGICAL*1 L$L,L$E,LMON,LF,LT
        COMMON/V$RBLE/XT(6),XP(6),XE(6),Z(6),ZP(6),ZE(6)
        COMMON/T$MER/DELT,DT,TIME,TI,TF,N,ISLEN,IOLEN
        COMMON/SY$TEM/L$L(40),L$E(10)
        COMMON/M$NITR/LMON(20)
        COMMON/NOI$E/IRAN(10),DG(10),RFRAF(6)
        COMMON/KALMAN/P(6,6),FILT(6,4),U(6),RN(4,4),B(4,6)
        COMMON/FORCE/F(3)
        COMMON/OFFSET/C(3)
        COMMON/MOOCH/T6(4,4)
        DATA LF,LT/F,T/
    601 FORMAT(1H ,3X,'TRUE STATE VECTOR'/4X,1P6D12.5)
    603 FORMAT(1H ,'TIME=',1PD12.5)
    604 FORMAT(1H ,'NORMED DISTANCE=',1PD12.5,3X,'NORMED VELOC
       *ITY=',
       11PD12.5,3X,'NORMED FORCE=',1PD12.5)
```

```
605 FORMAT(1HO,'$$$  RENDEZVOUS $$$')
606 FORMAT(1HO,'MINIMUM NORMED DISTANCE=',1PD12.5,3X,'AT T
   *IME=',
   1 1PD12.5)
607 FORMAT(1HO,'MINIMUM NORMED VELOCITY =',1PD12.5,3X,'AT
   *TIME=',
   1 1PD12.5)
608 FORMAT(////)
609 FORMAT(1HO,'**** OUT OF TIME ***')
610 FORMAT(1HO,'FORCE VECTOR',/,1H ,1P3D12.5)
611 FORMAT(1HO,3X,'PREDICTED STATE VECTOR'/4X,1P6D12.5)
612 FORMAT(1HO,3X,'ERROR IN STATE VECTOR'/4X,1P6D12.5)
613 FORMAT(1HO,3X,'TRUE OBSERVATIONS'/4X,1P6D12.5)
614 FORMAT(1HO,3X,'PREDICTED OBSERVATIONS'/4X,1P6D12.5)
615 FORMAT(1HO,3X,'RESIDUAL ERROR'/4X,1P6D12.5)
617 FORMAT(1HO,3X,'COVARIANCE MATRIX')
618 FORMAT(1H ,6X,1P6D12.5)
619 FORMAT(1H1,10X,' SIMULATION RESULTS',////)
    IF(L$L(1))WRITE(6,619)
620 FORMAT(1H ,3X,'NORMED POSITION ERROR = ',1P1D12.5,5X,'
   *NORMED VELOC
   $ITY ERROR = ',1P1D12.5)
    FT=0.0D0
    T1=0.0D0
    T2=0.0D0
    DO 1 I=1,3
    F(I)=F(I)*CF1
    FT=FT+F(I)**2
    T1=T1+XT(I)*XT(I)
  1 T2=T2+XT(I+3)*XT(I+3)
    FT=DSQRT(FT)
    XS=DSQRT(T1)
    XV=DSQRT(T2)
    IF(L$L(9))GO TO 2
    IF(XS.GT.XO)GO TO 3
    XO=XS
    TXS=TIME
  3 IF(XV.GT.XVO)GO TO 2
    XVO=XV
    TXV=TIME
  2 WRITE(6,603)TIME
    WRITE(6,604)XS,XV,FT
    CHK=DABS(TF-TIME)
    IF(CHK.LE.(1.10D0*DELT)) GO TO 50
    GO TO 60
 50 FTHRST=DSQRT(F(1)**2+F(2)**2+F(3)**2)
    GUTL=FTHRST/1.0D-04
    GUTTR=DFLOAT(IFIX(GUTL))
    QCHK=GUTTR+1.0D0
    TF=TF+QCHK*DELT
 60 CONTINUE
```

```
      IF(L$L(2))RETURN
      DO 99901 I$=1,ISLEN
99901 XE(I$)=XT(I$)-XP(I$)
      PNORM=DSQRT(XE(1)**2+XE(2)**2+XE(3)**2)
      VNORM=DSQRT(XE(4)**2+XE(5)**2+XE(6)**2)
      WRITE(6,610)F
      WRITE(6,601)XT
      WRITE(6,611)XP
      WRITE(6,612)XE
      WRITE(6,620)PNORM,VNORM
      WRITE(6,613)Z
      WRITE(6,614)ZP
      WRITE(6,615)ZE
      WRITE(6,617)
      DO 6 I=1,ISLEN
    6 WRITE(6,618)(P(I,J),J=1,ISLEN)
      WRITE(6,608)
99902 L$L(1)=LF
      RETURN
      ENTRY TERMIN
     *                ENTRY
      IF(XS.LT.1.0D-2.AND.XV.LT.1.0D-4)GO TO 4
      IF(TIME.GE.TF)GO TO 5
      RETURN
    4 WRITE(6,605)
      L$L(6)=LF
      RETURN
    5 WRITE(6,609)
      L$L(6)=LF
      RETURN
      ENTRY RECAP
     *                ENTRY
      WRITE(6,606)XO,TXS
      WRITE(6,607)XVO,TXV
      RETURN
      ENTRY CYINIT
      XO=1.0D40
      XVO=1.0D40
C     GRAVITATIONAL ACCELLERATION INVERSE
      CF1=1.0D0/(9.80665D-08*8.64D4**2)
      RETURN
      END
      SUBROUTINE TARGET(S,T)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION S(3)
      COMMON/VAR1/A,RN,EPS,ET,TSTAR,DET
  601 FORMAT(1H0,'CONVERGENCE= ',1PD12.5,' KEPEQ')
      EO=ET
      ET=ET+DET
      DO 1 I=1,100
      SINET=DSIN(ET)
```

```
      COSET=DCOS(ET)
      F=RN*(T+TSTAR)-ET+EPS*SINET
      DF=EPS*COSET-1.0D0
      ET=ET-F/DF
      DIF=DABS(F/DF)
      IF(DIF.LT.1.0D-10)GO TO 2
    1 CONTINUE
      WRITE(6,601)DIF
    2 DET=ET-EO
      S(1)=A*(COSET   -EPS)
      S(2)=A*DSQRT(1.0D0-EPS*EPS)*SINET
      S(3)=0.0D0
      RETURN
      END
      SUBROUTINE MINV(A,N,NSQ,L,M,BIGA)
      IMPLICIT REAL*8 (A-H,O-Z)
     *              00000135
      DIMENSION A(NSQ),L(6),M(6)
C
     *              00013200
C        DESCRIPTION OF PARAMETERS
     *              00013500
C           A - INPUT MATRIX, DESTROYED IN COMPUTATION AND R
     *EPLACED BY  00013600
C                RESULTANT INVERSE.
     *              00013700
C           N - ORDER OF MATRIX A
     *              00013800
C        BIGA - RESULTANT DETERMINANT
     *              00013900
C           L - WORK VECTOR OF LENGTH N
     *              00014000
C           M - WORK VECTOR OF LENGTH N
     *              00014100
C
     *              00014200
      NK=-N
     *              00017500
      DO 190 K=1,N
     *              00017600
      NK=NK+N
     *              00017700
      L(K)=K
     *              00017800
      M(K)=K
     *              00017900
      KK=NK+K
     *              00018000
      BIGA=A(KK)
     *              00018100
      DO 30 J=K,N
     *              00018200
      IZ=N*(J-1)
```

```
  *                 00018300
   DO 30 I=K,N
  *                 00018400
   IJ=IZ+I
  *                 00018500
10 IF(DABS(BIGA)-DABS(A(IJ))) 20,30,30
20 BIGA=A(IJ)
  *                 00018800
   L(K)=I
  *                 00018900
   M(K)=J
  *                 00019000
30 CONTINUE
  *                 00019100
C
  *                 00019200
C        INTERCHANGE ROWS
  *                 00019300
C
  *                 00019400
   J=L(K)
  *                 00019500
   IF(J-K) 60,60,40
  *                 00019600
40 KI=K-N
  *                 00019700
   DO 50 I=1,N
  *                 00019800
   KI=KI+N
  *                 00019900
   HOLD=-A(KI)
  *                 00020000
   JI=KI-K+J
  *                 00020100
   A(KI)=A(JI)
  *                 00020200
50 A(JI) =HOLD
  *                 00020300
C
  *                 00020400
C        INTERCHANGE COLUMNS
  *                 00020500
C
  *                 00020600
60 I=M(K)
  *                 00020700
   IF(I-K) 90,90,70
  *                 00020800
70 JP=N*(I-1)
  *                 00020900
   DO 80 J=1,N
  *                 00021000
   JK=NK+J
  *                 00021100
   JI=JP+J
  *                 00021200
   HOLD=-A(JK)
  *                 00021300
   A(JK)=A(JI)
  *                 00021400
```

```
   80 A(JI) =HOLD
      *               00021500
C
      *               00021600
C        DIVIDE COLUMN BY MINUS PIVOT (VALUE OF PIVOT ELEMEN
      *T IS           00021700
C        CONTAINED IN BIGA)
      *               00021800
C
      *               00021900
   90 IF(BIGA) 110,100,110
      *               00022000
  100 RETURN
      *               00022100
  110 DO 130 I=1,N
      *               00022200
      IF(I-K) 120,130,120
      *               00022300
  120 IK=NK+I
      *               00022400
      A(IK)=A(IK)/(-BIGA)
      *               00022500
  130 CONTINUE
      *               00022600
C
      *               00022700
C        REDUCE MATRIX
      *               00022800
C
      *               00022900
      DO 160 I=1,N
      *               00023000
      IK=NK+I
      *               00023100
      IJ=I-N
      *               00023200
      DO 160 J=1,N
      *               00023300
      IJ=IJ+N
      *               00023400
      IF(I-K) 140,160,140
      *               00023500
  140 IF(J-K) 150,160,150
      *               00023600
  150 KJ=IJ-I+K
      *               00023700
      A(IJ)=A(IK)*A(KJ)+A(IJ)
      *               00023800
  160 CONTINUE
      *               00023900
C
      *               00024000
C        DIVIDE ROW BY PIVOT
      *               00024100
C
      *               00024200
      KJ=K-N
      *               00024300
      DO 180 J=1,N
      *               00024400
```

```
      KJ=KJ+N
    *               00024500
      IF(J-K) 170,180,170
    *               00024600
  170 A(KJ)=A(KJ)/BIGA
    *               00024700
  180 CONTINUE
    *               00024800
C
    *               00024900
C         REPLACE PIVOT BY RECIPROCAL
    *               00025000
C
    *               00025100
      A(KK)=1.0/BIGA
    *               00025200
  190 CONTINUE
    *               00025300
C
    *               00025400
C         FINAL ROW AND COLUMN INTERCHANGE
    *               00025500
C
    *               00025600
      K=N
    *               00025700
  200 K=(K-1)
    *               00025800
      IF(K) 270,270,210
    *               00025900
  210 I=L(K)
    *               00026000
      IF(I-K) 240,240,220
    *               00026100
  220 JQ=N*(K-1)
    *               00026200
      JR=N*(I-1)
    *               00026300
      DO 230 J=1,N
    *               00026400
      JK=JQ+J
    *               00026500
      HOLD=A(JK)
    *               00026600
      JI=JR+J
    *               00026700
      A(JK)=-A(JI)
    *               00026800
  230 A(JI) =HOLD
    *               00026900
  240 J=M(K)
    *               00027000
      IF(J-K) 200,200,250
    *               00027100
  250 KI=K-N
    *               00027200
      DO 260 I=1,N
    *               00027300
      KI=KI+N
    *               00027400
```

```
      HOLD=A(KI)
     *                 00027500
      JI=KI-K+J
     *                 00027600
      A(KI)=-A(JI)
     *                 00027700
  260 A(JI) =HOLD
     *                 00027800
      GO TO 200
     *                 00027900
  270 RETURN
     *                 00028000
      END
     *                 00028100
      SUBROUTINE CONOBS (NP,NU,NY,F,G,H)
     *                 CNBS0000
      DIMENSION F(6,6),G(6,1),H(4,6),B(6,24),S1(6,6),W(6,6),
     *S(6,6)
      DOUBLE PRECISION DET1
      LO=NU
     *                 CNBS0050
      DO 32 I=1,NP
     *                 CNBS0060
      DO 32 J=1,NU
     *                 CNBS0070
32    S(I,J) = G(I,J)
     *                 CNBS0080
      DO 33 I=1,NP
     *                 CNBS0090
      DO 33 J=1,NP
     *                 CNBS0100
   33 S1(I,J)=F(I,J)
     *                 CNBS0110
      DO 85 ITEST=1,2
     *                 CNBS0120
      IF (LO .EQ. 0) GO TO 90
     *                 CNBS0130
      CALL NORML (NP,LO,S)
     *                 CNBS0140
      DO 35 I = 1,NP
     *                 CNBS0150
      DO 35 J = 1,LO
     *                 CNBS0160
35          B(I,J) = S(I,J)
     *                 CNBS0170
      L=1
     *                 CNBS0180
      MOU=NP-1
     *                 CNBS0190
      DO 40 IT=1,MOU
     *                 CNBS0200
      CALL MMUL (NP,NP,LO,S1,S,W)
     *                 CNBS0210
      CALL NORML (NP,LO,W)
     *                 CNBS0220
      DO 20 I=1,NP
     *                 CNBS0230
      DO 20 J=1,LO
     *                 CNBS0240
      J1=(J+L*LO)
```

```
    *              CNBS0250
     B(I,J1)=W(I,J)
    *              CNBS0260
  20 S(I,J)=W(I,J)
    *              CNBS0270
  40 L=L+1
    *              CNBS0280
     NDIM=L*LO
    *              CNBS0290
     DO 41 I = 1,NP
    *              CNBS0292
     SUM = 0.0
    *              CNBS0294
     DO 42 J = 1,NDIM
    *              CNBS0296
  42 SUM = SUM + B(I,J)**2
    *              CNBS0298
     SUM = SQRT(SUM)
    *              CNBS0300
     IF (SUM .EQ. 0.) GO TO 41
    *              CNBS0302
     DO 45 J = 1,NDIM
    *              CNBS0304
  45 B(I,J) = B(I,J)/SUM
    *              CNBS0306
  41 CONTINUE
    *              CNBS0308
           DO 1000 II = 1,NP
    *              CNBS0310
                DO 1000 JJ = 1,NP
    *              CNBS0320
                   SUM = 0.0
    *              CNBS0330
                   DO 1001 KK = 1,NDIM
    *              CNBS0340
1001                    SUM = SUM + B(II,KK)*B(JJ,KK)
    *              CNBS0350
1000       W(II,JJ) = SUM
    *              CNBS0360
     CALL MXINV (NP,W,DET1,IER)
     IF (ITEST .GT. 1) GO TO 87
    *              CNBS0380
     IF (IER .EQ. 0) GO TO 70
    *              CNBS0390
     GO TO 90
    *              CNBS0420
  70 WRITE(6,300) DET1
 300 FORMAT (26H CONTROLLABLE      DET =   ,1PE14.4)
    *              CNBS0440
  90 LO=NY
    *              CNBS0450
     DO 93 J=1,NY
    *              CNBS0460
     DO 93 I=1,NP
    *              CNBS0470
  93 S(I,J)=H(J,I)
    *              CNBS0480
  85 CALL MTRANS (NP,NP,F,S1)
    *              CNBS0490
  87 IF (IER .EQ. 0) GO TO 170
```

```
      *              CNBS0500
      WRITE(6,902) DET1
  902 FORMAT (26H NOT OBSERVABLE    DET =   ,1PE14.4)
      *              CNBS0520
      GO TO 91
      *              CNBS0530
  170 WRITE(6,901) DET1
  901 FORMAT (26H OBSERVABLE        DET =   ,1PE14.4)
      *              CNBS0550
   91 RETURN
      *              CNBS0560
      END
      *              CNBS0570
      SUBROUTINE MATRIX
      *              MTRX0000
      DIMENSION A(6,6),B(6,6),C(6,6)
      ENTRY      MMUL (MP,NP,NU,A,B,C)
      *              MTRX0520
      DO 11 L=1,MP
      *              MTRX0530
      DO 11 I=1,NU
      *              MTRX0540
      SUM = 0.0
      *              MTRX0550
      DO 31 J=1,NP
      *              MTRX0560
   31 SUM = SUM + A(L,J)*B(J,I)
      *              MTRX0570
   11 C(L,I) = SUM
      *              MTRX0580
      RETURN
      *              MTRX0590
      ENTRY      MTRANS  (M,N,A,B)
      *              MTRX0790
      DO 10 I=1,M
      *              MTRX0800
      DO 10 J=1,N
      *              MTRX0810
   10 B(J,I)  = A(I,J)
      *              MTRX0820
      RETURN
      *              MTRX0830
      ENTRY    MXINV (NP,A,DET1,IER)
      DOUBLE PRECISION DA(6,6),DET1,DB(1,1)
      DO 1 I = 1,NP
      *              MTRX0860
      DO 1 J = 1,NP
      *              MTRX0870
    1 DA(I,J) = A(I,J)
      *              MTRX0880
      CALL DINVER (NP,6,DA,0,1,DB,DET1,IER)
      IF (IER .NE. 0) RETURN
      *              MTRX0900
      DO 2 I = 1,NP
      *              MTRX0910
      DO 2 J = 1,NP
      *              MTRX0920
    2 A(I,J) = DA(I,J)
      *              MTRX0930
      RETURN
```

```
      *                MTRX0940
       ENTRY NORML (NR,NC,A)
      *                MTRX1000
             DO 4 J = 1,NC
      *                MTRX1010
                  SUM = 0.0
      *                MTRX1020
                  DO 3 I = 1,NR
      *                MTRX1030
3                    SUM = SUM + A(I,J)**2
      *                MTRX1040
             SUM = SQRT(SUM)
      *                MTRX1050
       IF (SUM .EQ. 0.) GO TO 4
      *                MTRX1060
                  DO 5 I = 1,NR
      *                MTRX1070
     5                 A(I,J) = A(I,J)/SUM
      *                MTRX1080
     4 CONTINUE
      *                MTRX1090
       RETURN
      *                MTRX1100
       END
      *                MTRX1110
       SUBROUTINE DINVER (NA,NAD,A,NB,NBD,B,DET1,IERROR)
C      THIS SUBROUTINE IS A MODIFICATION OF THE UNIVERSITY OF
      * FLORIDA        DNVR0010
C      COMPUTER CENTER'S INVERT . IT USES DOUBLE PRECISION AN
      *D HAS BEEN      DNVR0020
C      RENAMED DINVER. C FOSHA 2-69
      *                DNVR0030
       DIMENSION A(NAD,NAD),B(NBD,NBD),BD(6),INDEX(6)
       DOUBLE PRECISION A,B,BD,SAVE,PIVOT,DET1
       DET1=1.0D0
       IERROR = 0
      *                DNVR0070
       DO   130   I = 1, NA
      *                DNVR0080
       PIVOT = 0.0D0
      *                DNVR0090
C                 SEARCH FOR PIVOTAL ELEMENT
      *                DNVR0100
       DO   60    J = I, NA
      *                DNVR0110
       IF (DABS(A(J,I)) .LE. DABS(PIVOT)) GO TO 60
      *                DNVR0120
       PIVOT = A(J,I)
      *                DNVR0130
       INDEX(I) = J
      *                DNVR0140
    60 CONTINUE
      *                DNVR0150
       IF (DABS(PIVOT)  .LT. 1. D-6) GO TO 250
      *                DNVR0160
       IF (INDEX(I) .EQ. I)      GO TO 90
      *                DNVR0170
       DET1=-DET1
C                 INTERCHANGE ROWS TO PUT PIVOTAL ELEMENT ON D
      *IAGONAL         DNVR0190
```

```
      DO   80    L = 1, NA
     *              DNVR0200
          SAVE = A(I,L)
     *              DNVR0210
      A(I,L) = A(INDEX(I), L)
     *              DNVR0220
80    A(INDEX(I), L) = SAVE
     *              DNVR0230
   90 DET1=DET1*PIVOT
      A(I,I) = 1.0D0
     *              DNVR0250
      DO   91    KK=1,NA
     *              DNVR0260
91    A(I,KK)=A(I,KK)/PIVOT
     *              DNVR0270
C                   REDUCE NON-PIVOTAL ROWS
     *              DNVR0280
      DO   130   LJ = 1, NA
     *              DNVR0290
          IF (LJ .EQ. I)     GO TO 130
     *              DNVR0300
      SAVE = A(LJ, I)
     *              DNVR0310
      A(LJ,I) = 0.0D0
     *              DNVR0320
      DO   120   K = 1, NA
     *              DNVR0330
120   A(LJ,K) = A(LJ,K) - SAVE * A(I,K)
     *              DNVR0340
130   CONTINUE
     *              DNVR0350
C                   INTERCHANGE COLUMNS
     *              DNVR0360
      NA1=NA+1
     *              DNVR0370
      DO   160   KKK = 1, NA
     *              DNVR0380
      K = NA1 - KKK
     *              DNVR0390
      IF (INDEX(K) .EQ. K)     GO TO 160
     *              DNVR0400
      DO   98    L = 1, NA
     *              DNVR0410
      SAVE = A(L,K)
     *              DNVR0420
      A(L,K) = A(L, INDEX(K))
     *              DNVR0430
98    A(L, INDEX(K)) = SAVE
     *              DNVR0440
160   CONTINUE
     *              DNVR0450
C                   A INVERSE IS NOW STORED IN A
     *              DNVR0460
C                   FIND SOLUTION VECTORS FOR ALL CONSTANT VECTO
     *RS INPUT      DNVR0470
      IF (NB .LE. 0)   RETURN
     *              DNVR0480
      DO   190   K = 1, NB
     *              DNVR0490
      DO   180   I = 1, NA
```

```
     *                DNVR0500
      BD(I) = 0.0D0
     *                DNVR0510
      DO   180   J = 1, NA
     *                DNVR0520
180   BD(I) = BD(I) + A(I,J)*B(J,K)
     *                DNVR0530
      DO   190   I = 1, NA
     *                DNVR0540
190   B(I,K) = BD(I)
     *                DNVR0550
C                    SOLUTION VECTORS NOW IN B
     *                DNVR0560
      RETURN
     *                DNVR0570
C                    IF CONTROL REACHES 250, MATRIX IS SINGULAR
     *                DNVR0580
250   IERROR = +1
     *                DNVR0590
      DET1=0.0D0
      RETURN
     *                DNVR0610
      END
     *                DNVR0620
      SUBROUTINE INTGR
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL*1 L$L,L$E,LMON,LF,LT
      LOGICAL*1 LS1,LS2
      COMMON/V$RBLE/CRAP(6),XT(6),XE(6),Z(6),ZP(6),ZE(6)
      COMMON/T$MER/DELT,DT,TIME,TI,TF,N,ISLEN,IOLEN
      COMMON/SY$TEM/L$L(40),L$E(10)
      COMMON/M$NITR/LMON(20)
      COMMON/NOI$E/IRAN(10),DG(10),RFRAF(6)
      COMMON/OFFSET/C(3)
      COMMON/XPROP/XPNEW(6)
      DIMENSION XINT(6),SUM(6)
      DATA LF,LT/F,T/
      L$L(12)=LT
  700 CONTINUE
      TIME=TIME-DELT
      DO 1 ICYCLE=1,N
      DO 33 I=1,ISLEN
   33 SUM(I)=0.0D0
      L$L(10)=LT
      DO 10 I1=1,4
      LS1=I1.EQ.2. OR.I1.EQ.3
      LS2=I1.EQ.4
      L$L(11)=I1.EQ.3
      F=F1
      FS=F5
      IF(LS1)F=F2
      IF(LS1)FS=F3
      IF(LS2)FS=F4
      TS=TIME+DT*FS
      DO 20 I=1,ISLEN
 20   XINT(I)=XT(I)+FS*XINT(I)
      DO 31 I=1,NP1
      J=I-1
      IF(J.GT.0) GO TO 2
      DUM=DOFX$N(J,TS)
```

```
      DUM=GOFX$N(J,TS)
      L$L(12)=LF
      GO TO 31
    2 XINT(J)=DT*(DOFXN(J)+GOFXN(J))
      SUM(J)=SUM(J)+F*XINT(J)
31    CONTINUE
      L$L(10)=LF
10    CONTINUE
      TIME=TIME+DT
      DO 11 I=1,ISLEN
11    XT(I)=XT(I)+SUM(I)
    1 CONTINUE
  701 CONTINUE
      RETURN
      ENTRY INTINI
      F1=1.0D0/6.0D0
      F2=2.0D0*F1
      F3=1.0D0/2.0D0
      F4=1.0D0
      F5=0.0D0
      NP1=ISLEN+1
      DO 32 I=1,ISLEN
   32 XINT(I)=XT(I)
      RETURN
      END
      FUNCTION DOFX$N(J,TS)
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL*1 L$L,L$E,LMON,LF,LT
      COMMON/V$RBLE/CRAP(6),XT(6),XE(6),Z(6),ZP(6),ZE(6)
      COMMON/T$MER/DELT,DT,TIME,TI,TF,N,ISLEN,IOLEN
      COMMON/SY$TEM/L$L(40),L$E(10)
      COMMON/M$NITR/LMON(20)
      COMMON/NOI$E/IRAN(10),DG(10),RFRAF(6)
      COMMON/VAR1/A,RN,EPS,EO,TSTAR,TRASH
      COMMON/NEWDET/DETN
      COMMON/OFFSET/C(3)
      DIMENSION D(3),S(3)
      DATA LF,LT/F,T/
99601 FORMAT(1H0,I4,'IMPROPER INDEX *DOFX')
      IF(.NOT.L$L(11))CALL TARGEN(S,TS)
      S2=0.0D0
      D2=0.0D0
      DO 1 I=1,3
      D(I)=S(I)+C(I)+XT(I)
      D2=D2+D(I)**2
1     S2=S2+S(I)**2
      DN=DSQRT(D2)
      SN=DSQRT(S2)
      RATI=GM/(SN*SN*SN)
      RAT2=(SN/DN)**3
      DOFX$N=0.0D0
      RETURN
      ENTRY DOFXN(J)
      GO TO (99999,99999,99999,99998,99998,99998),J
      WRITE(6,99601)J
      DOFXN=0.0D0
      L$E(2)=LT
      RETURN
99999 DOFXN=XT(J+3)
      RETURN
```

```
99998 DOFXN=RATI*(S(J-3)-D(J-3)*RAT2)
      RETURN
      ENTRY DXININ(J)
      GM=9.90549D05
      DETN=1.0D-3
      DXININ=0.0D0
      RETURN
      END
      FUNCTION GOFX$N(J,TS)
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL*1 L$L,L$E,LMON,LF,LT
      COMMON/V$RBLE/XT(6),XP(6),XE(6),Z(6),ZP(6),ZE(6)
      COMMON/T$MER/DELT,DT,TIME,TI,TF,N,ISLEN,IOLEN
      COMMON/SY$TEM/L$L(40),L$E(10)
      COMMON/M$NITR/LMON(20)
      COMMON/NOI$E/IRAN(10),DG(10),RFRAF(6)
      COMMON/FORCE/F(3)
      COMMON/OFFSET/C(3)
      COMMON/XPROP/XPNEW(6)
      DATA LF,LT/F,T/
99601 FORMAT(1H0,I4,'IMPROPER INDEX *GOFX')
      IF(L$L(12))TIM1=TS
      TAUO=TF-TIM1
      TAU=TF-TS
      TRAT1=(6.0D0/TAUO**2)*(1.0D0-2.0D0*(TAU/TAUO))
      TRAT2=(2.0D0/TAUO)*(1.0D0-3.0D0*(TAU/TAUO))
      GOFX$N=0.0D0
      RETURN
      ENTRY GOFXN(J)
      GO TO (99999,99999,99999,99998,99998,99998),J
      WRITE(6,99601)J
      L$E(3)=LT
      GOFXN=0.0D0
      RETURN
99999 GOFXN=0.0D0
      RETURN
99998 F(J-3)=TRAT1*XPNEW(J-3)+TRAT2*XPNEW(J)
      GOFXN=F(J-3)
      RETURN
      ENTRY GXININ(J)
      GXININ=0.0D0
      RETURN
      END
      SUBROUTINE TARGEN(S,T)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION S(3)
      COMMON/VAR1/A,RN,EPS,ET,TSTAR,TRASH
      COMMON/NEWDET/DETN
  601 FORMAT(1H0,'CONVERGENCE=',1PD12.5,' KEPEQ')
      EO=ET
      ET=ET+DETN
      DO 1 I=1,100
      SINET=DSIN(ET)
      COSET=DCOS(ET)
      F=RN*(T+TSTAR)-ET+EPS*SINET
      DF=EPS*COSET-1.0D0
      ET=ET-F/DF
      DIF=DABS(F/DF)
      IF(DIF.LT.1.0D-10)GO TO 2
    1 CONTINUE
```

```
      WRITE(6,601)DIF
    2 DETN=ET-EO
      S(1)=A*(COSET-EPS)
      S(2)=A*DSQRT(1.0D0-EPS*EPS)*SINET
      S(3)=0.0D0
      RETURN
      END
//GO.SYSIN DD *
TEST RUN
TFFTTTTTFTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
 10   6   4
    34.0       39.0       0.01       0.1
 .366447   .275514   .192315   -.141411   -.106223   -.072940
3318.76000.0052056   .846765   6.0        1142.8
0.01      0.01      0.01
```