AUBURN UNIVERSITY

AEROSPACE ENGINEERING

STUDY OF EFFECTS OF UNCERTAINTIES ON COMET AND ASTEROID ENCOUNTER AND CONTACT GUIDANCE REQUIREMENTS

FINAL REPORT

PART II. TUMBLING PROBLEM STUDIES

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

Under Contract NAS8-27664

May 3, 1974

ENGINEERING EXPERIMENT STATION
AUBURN UNIVERSITY
AUBURN, ALABAMA 36830

Aerospace Engineering Department

Auburn University

Auburn, Alabama 36830

STUDY OF EFFECTS OF UNCERTAINTIES ON COMET AND ASTEROID ENCOUNTER AND CONTACT GUIDANCE REQUIREMENTS

FINAL REPORT

PART II. TUMBLING PROBLEM STUDIES

Prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

GEORGE C. MARSHALL SPACE FLIGHT CENTER

Under Contract NAS8-27664

May 3, 1974

Arthur G. Bennett, Associate Professor
Project Director

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

John E. Cochran, Jr., Assistant Professor
Principal Investigator

STUDY OF EFFECTS OF UNCERTAINTIES ON COMET

AND ASTEROID ENCOUNTER AND CONTACT

GUIDANCE REQUIREMENTS

NAS8-27664

PROJECT SUMMARY

At the start of our work in June 1971, three principal tasks were assigned. Slightly restated, these tasks were:

(i) Develop a deterministic algorithm for guidance to rendezvous with comets and asteroids that can handle expected large target ephemeris errors.

(ii) Define the problem of determination of rotational state of a tumbling asteroid or cometary nucleus and develop possible schemes for this determination.

(iii) Investigate possible contact rendezvous schemes including the "harpoon" technique.

During the first year, a detailed investigation of a rendezvous guidance technique based on "encounter theory" was conducted. The definition and formulation of the tumbling problem was made and several possible algorithms phrased. A first investigation of the harpoon problem was conducted and frequencies and acceleration levels identified.

Early in the second year work, a successful deterministic rendezvous guidance algorithm based on optimal control theory was developed. The algorithm was considered sufficiently important that, with agreement of NASA, more emphasis was placed on the rendezvous investigation. To accommodate this work, the harpoon study was set aside. An effort was initiated on the rendezvous navigation problem wherein measurements are made and statistically processed onboard the spacecraft to provide the relative state information required for input to the guidance algorithm. Expenditures on the contract were low enough so that in June, 1972, a no-cost extension of the work to September, 1973, was possible. At this time the changes in objective were formalized and principal tasks were restated to include the navigation work (and eliminate the contact-rendezvous and harpoon investigation).

In September, 1973, delays caused by installation of a new computing machine at the University prevented generation of final data. The contract completion date was again extended, at no cost, to December 15, 1973, to allow time for this data generation and report preparation.

The final report of our work is presented in two volumes:

Part I. Guidance and Navigation Studies

Part II. Tumbling Problem Studies

Each of these volumes presents the technical details of the analyses conducted, the principal conclusions made, and listing of the computer programs employed, including descriptions of the operation of the programs. Technical abstracts of the work are included in each volume. In Part I, the body of the report reproduces a paper prepared for the AIAA 10th Electric Propulsion Conference entitled "Solar Electric Propulsion for Terminal Flight to Rendezvous with Comets and Asteroids." (AIAA Paper No. 73-1062). The title was changed for inclusion in this report and a few typographical errors were corrected.

STUDY OF EFFECTS OF UNCERTAINTIES ON COMET
AND ASTEROID ENCOUNTER AND CONTACT
GUIDANCE REQUIREMENTS

TABLE OF CONTENTS

STUDY OF EFFECTS OF UNCERTAINTIES ON COMET AND ASTEROID ENCOUNTER AND CONTACT GUIDANCE REQUIREMENTS

PART II. TUMBLING PROBLEM STUDIES

Abstract

The problem of determining the rotational motion of a tumbling celestial body of the asteroid type using spacecraft-acquired data is addressed. The rotational motion of the body is modeled by free-Eulerian motion of a triaxial, rigid body and its translational motion with respect to a nonrotating, observing spacecraft, which is not thrusting, is assumed to be uniform during the time observations are made. The mathematical details which form the basis for a digital simulation of the motion and observations are presented. Two algorithms for determining the motion from observations for the special case of uniform rotational motion are given. Results of studies of the performance of the algorithms using ideal data and "non-ideal" data are discussed. The "non-ideal" data is generated by inducing initial conditions which cause the body to "wobble" in general free-Eulerian motion. One algorithm performs satisfactorily when the rotational motion of the body is truly uniform, but, in general, fails to determine realistic values of the constants which are determined when the rotational motion is not uniform. The other algorithm does not perform well.

I. Introduction

Future missions to the asteroid belt or a comet using autonomous and/or semi-autonomous spacecraft of the CARD (Comet and Asteroid Rendezvous and Docking) type will foreseeably require the spacecraft to land on, or "dock" with, the tumbling target body. The successful completion of such a task will depend to a large extent on the available knowledge concerning the target's rotational motion and its translational motion relative to the rendezvous spacecraft. Such knowledge, if not available *a priori*, must be determined in some manner using observations of the motion of the target body made by sensors onboard the spacecraft. Thus, methods for determining the rotational and relative translational motions of the target need to be developed. The development of such methods, or algorithms, is the subject of this report.

A prerequisite of any motion determination algorithm, such as, for example, a satellite orbit determination algorithm, is the formulation of a suitable mathematical model for the motion which is to be determined. That is, the motion to be studied must be defined in physical terms and then described mathematically. The motion of interest here is that embodied in what we call "the tumbling problem." The definition of this problem is given the following section. Stated briefly, the motion consists of free-Eulerian motion of the target body about its center of mass and uniform translational motion of the body's center of mass with respect to the center of mass of a nonrotating, observing spacecraft. Such motion can be described exactly in mathematical terms and the motion determination problem is reduced to the determination of the "constants of the motion" which are the arbitrary constants obtained in integrating the equations of motion of the system.

Since the type of motion to be considered cannot be easily simulated physically, the mathematical model of the motion is useful not only in deriving the determination algorithms, but also in simulating the motion digitally to check the performance of the algorithms. Such a simulation capability was developed and utilized in this study and is discussed in the third section of this report.

The form of an algorithm for determining the constants of the motion depends, to a large degree, on the type of observational data available. In the fourth section of this document, two algorithms for determining these constants for the special case of uniform rotational motion (which will "probably" be the type of motion actually encountered) are described. The first algorithm, DUMRA (Determination of the Uniform Motion using Range and Angles), utilizes position data for three particular points on the surface of the target body at three instants of time. This data could be acquired by using the combination of (1) a scanning radar from which range and angles for many points on the body could be obtained, (2) a television camera, and (3) an earthbound observer. The observer would be used to identify the *same* three points on the body at the three instants of time and select the data for these points from among all that acquired (using, for example, a grided CRT display and a light pen).

The second algorithm utilizes simpler data. The data is range and range-rate data, also acquired via radar, for *one* specific point on the body at six instants of time, again with the aid of an observer to identify the point.

The performance of the algorithms was tested, as indicated previously, by simulating the motion of the body and from this motion determining the observations required for the algorithms. The section following the description of the algorithms is devoted to a discussion of these performance studies.

Conclusions drawn from the work conducted are presented at the close of the report and an Appendix containing information on the computer programs written and used during the study is also included.

II. Tumbling Problem Definition

A prerequisite to the development of algorithms for determining the rotational state of a tumbling body, such as an asteroid, is the selection of suitable mathematical models for the body and its motion. Since the type of body of interest is an asteroid or a comet nucleus, it is reasonable to assume that the body is a rigid one. Also, using available data[1] on the shape of asteroids, a triaxial, homogeneous, ellipsoidal, rigid body was selected as the physical model for the tumbling body. Of course, it is not suggested that any asteroid is actually perfectly homogeneous, nor that they all are ellipsoidal, but the shape appears reasonable and the torque-free rotational dynamics of <u>any</u> rigid body can be taken as those of a body with a triaxial inertia ellipsoid. Our physical ellipsoid is the reciprocal of the corresponding inertia ellipsoid and the lengths of its axes are denoted by a, b, and c where $a \geq b > c$.

On the basis of observations of fluctuations of of light reflected by asteroids, astronomers [2] have concluded that they rotate uniformly, or at least <u>almost</u> uniformly, about their centers of mass. Furthermore, the well-known fact that dissipation of rotational kinetic energy (internally, or through external means such as meteor impacts, with no significant change in rotational angular momentum will cause a triaxial body to ultimately rotate uniformly about its axis of maximum moment of inertia leads us to assume uniform, or almost uniform, rotational motion of the tumbling body about its axis of maximum moment of inertia. To allow for nonuniform motion, we have adopted a free-Eulerian dynamical model for the tumbling motion; that is, the rotational motion is assumed to be torque-free during the time of observation.

The relative motion of the spacecraft and the tumbling body centers of mass may be very nonuniform during the use of high thrust chemical systems. However, we assume that during the time that observations of the tumbling body are made the spacecraft's high thrust system will not be operating. Of course, relative accelerations of the tumbling body and the spacecraft may still occur due to (1) the body's gravitational attraction, (2) the use of the spacecraft's low thrust system, and (3) the gravity-gradient effect of the sun's gravitational field. However, by assuming a station keeping or a steady fly-by mode of operation for the spacecraft, the relative acceleration caused by the low thrust system should be small enough to be neglected over a period of time of several minutes and the same should be true of the acceleration due to the body's gravitational attraction and also that due to the sun. Hence, it is assumed that the relative motion is uniform.

The tumbling problem, as we have defined it, may thus be stated as follows:

> Using observations of the motion of a free, homogeneous, triaxial, rigid ellipsoid made from a reference point which is moving uniformly relative to the ellipsoid's center of mass, determine the motion of the ellipsoid; i.e., determine the constants of the motion and the physical characteristics of the body needed to predict its relative position and its orientation at any time in the future.

This is the problem we have addressed in this study. For the most part, however, it has been simplified by assuming uniform rotation as well as uniform relative translation. In the next section we consider the problem of simulating the observations which a spacecraft might make.

III. Simulation of Observations

Although the algorithms for determining the tumbling body motion which are described in the next section are predicated on the assumptions of uniform translational <u>and uniform</u> <u>rotational</u> motions, simulation capability to handle nonuniform, free-Eulerian motion of the triaxial body was developed. Such capability may be useful in future studies and may also be used, as it was here, to study the effects of small variations in the rotational motion on the uniform motion algorithms' performance. Details of how the observations of points on the surface of the ellipsoidal model are generated using a FORTRAN subroutine, STATE (see Appendix for listing) are given in what follows.

Physical Model

The tumbling body model discussed in the previous section is depicted in Figure 1. As stated, the model is a triaxial, rigid, homogeneous ellipsoid with axes of length a, b and c ($a \geq b > c$)

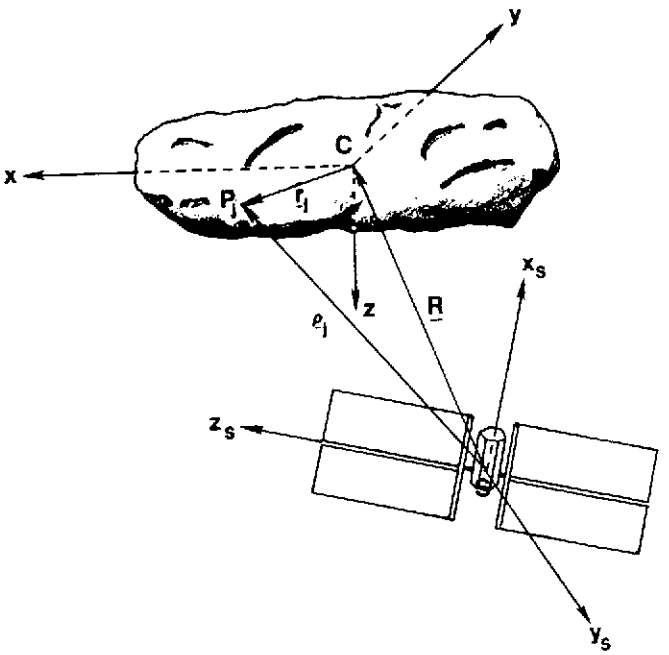

Fig. 1. Tumbling Problem Geometry

so that the moment of inertia about the z-axis (see Figure 1) is maximum. The moments of inertia about the axes x, y, and z are denoted by A, B, and C, respectively, and, as is obvious, $C > B > A$, in general.† For a homogeneous, triaxial ellipsoid we have,

$$A = (m/5)(b^2 + c^2)$$
$$B = (m/5)(a^2 + c^2) \qquad (1)$$
$$C = (m/5)(a^2 + b^2) \quad ,$$

where m is the mass of the ellipsoid. As we shall see, the mass of the ellipsoid does not affect its rotational motion as long as there are no external torques.

Rotational Motion

Let $Sx_s y_s z_s$ be an orthogonal, dextral, coordinate system which has its origin, S, at the center of mass of the spacecraft and which is nonrotating. Let CXYZ be a similar system which has its origin at C, the center of mass of the tumbling body and which is aligned with the $Sx_s y_s z_s$ system. Also, let $Cx_H y_H z_H$ denote a similar system which has its z_H-axis aligned with the constant rotational angular momentum vector, $\underline{H}$, of the tumbling body as shown in Figure 2. The orientation of the $Cx_H y_H z_H$ system is defined by the angles ψ_H and θ_H. Finally, let the Cxyz system, shown in Figure 2, be a principal, body-fixed coordinate system for the ellipsoid. The orientation of the Cxyz system with respect to the $Cx_H y_H z_H$ is defined by the set of Eulerian angles $\{\psi, \theta, \phi\}$.

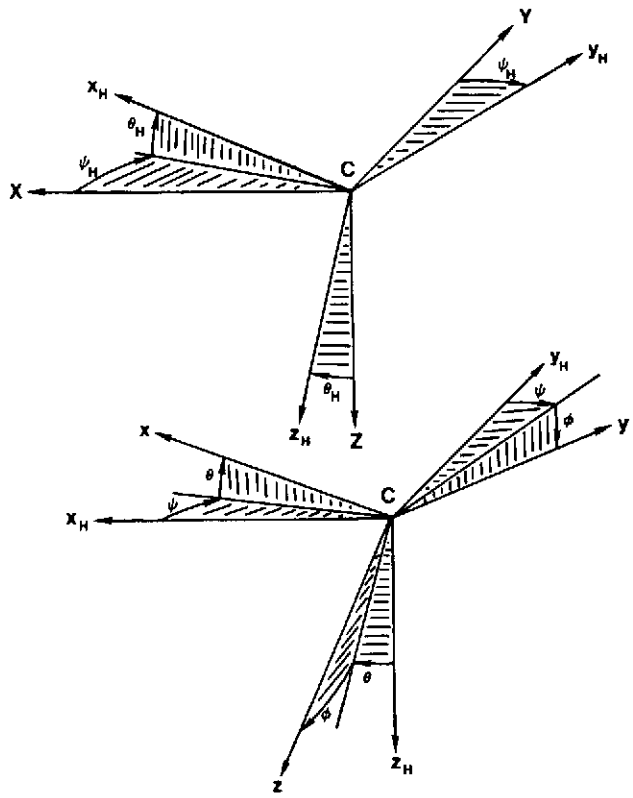

Fig. 2. Coordinate Systems.

The rotational motion of the ellipsoid is governed by Euler's equations for a free triaxial rigid body, which may be written in the forms,

$$A\dot{\omega}_x + (C-B)\,\omega_y\omega_z = 0$$
$$B\dot{\omega}_y + (A-C)\,\omega_z\omega_x = 0 \qquad (2)$$
$$C\dot{\omega}_z + (B-A)\,\omega_x\omega_y = 0,$$

where ω_x, ω_y and ω_z are the x-, y- and z-components, respectively, of $\underline{\omega}$, the angular velocity of the ellipsoid.

Since Eqs. (2) are homogeneous, we can and shall divide each of them by a constant factor. In addition, we shall introduce a new independent variable,

$$\tau = \omega_{z_o} t \quad , \qquad (3)$$

where ω_{z_o} is the value of ω_z at $t = 0$. Letting $()' = d()/d\tau$, we may then rewrite Eqs. (2) in the dimensionless forms,

$$\lambda'_x + (1-\sigma_2)\,\lambda_y\lambda_z = 0$$
$$\lambda'_y + (\sigma_1-1)\,\lambda_x\lambda_z = 0 \qquad (4)$$
$$\lambda'_z + (\sigma_2-\sigma_1)\,\lambda_x\lambda_y = 0$$

where $\lambda_x=\omega_x/\omega_{z_o}$, $\lambda_y=\omega_y/\omega_{z_o}$, $\lambda_z=\omega_z/\omega_{z_o}$, $\sigma_1 = A/C$ and $\sigma_2 = B/C$.

Equations (4) have the first integrals,

$$h^2 = \sigma_1^2\lambda_x^2 + \sigma_2^2\lambda_y^2 + \lambda_z^2 = \text{constant} \qquad (5)$$

and

$$\alpha = (\sigma_1\lambda_x^2 + \sigma_2\lambda_y^2 + \lambda_z^2) = \text{constant}, \qquad (6)$$

where $h = |\underline{H}|/C^2\,\omega_{z_o}^2$ and α is twice the rotational kinetic energy of the body divided by $C^2\,\omega_{z_o}^2$. The integrals (5) and (6) may be used to obtain the following solution([3]) to Eqs. (4):

$$\lambda_x = p\ \text{cn}\ u$$
$$\lambda_y = q\ \text{sn}\ u \qquad (7)$$
$$\lambda_z = r\ \text{dn}\ u\ ,$$

where

$$p = \{(\alpha - h^2)/[\sigma_1(1-\sigma_1)]\}^{\frac{1}{2}} \quad (8)$$

$$q = \{(\alpha - h^2)/[\sigma_2(1-\sigma_2)]\}^{\frac{1}{2}} \quad (9)$$

$$r = \{(h^2 - \sigma_1\alpha)/(1-\sigma_1)]\}^{\frac{1}{2}} \quad (10)$$

$$u = \lambda\tau - \nu, \ \nu = \text{constant}, \quad (11)$$

$$\lambda = \{(\sigma_2 - \sigma_1)(\alpha-h^2)/[(1-\sigma_2)(h^2-\sigma_1\alpha)]\}^{\frac{1}{2}} \quad (12)$$

and the functions cn u, sn u and dn u are Jacobian elliptic functions of modulus

$$k = \{[(\sigma_2-\sigma_1)(\alpha-h^2)]/[(1-\sigma_2)(h^2-\sigma_1\alpha)]\}^{\frac{1}{2}} . \quad (13)$$

Note that, since σ_1 and σ_2 do not contain the mass, m, of the body, the rotational motion of the body is not affected by the value of m, but is affected by the distribution of the mass.

To allow for arbitrary initial conditions on λ_x and λ_y (initially $\lambda_z = 1$, of course), the elliptic functions are rewritten in the forms,[4]

$$\begin{aligned} \text{cn } u &= \text{cn } (\lambda t-\nu) = \text{cn } \lambda t \text{ cn } \nu + \\ &\quad \text{sn } \lambda t \text{ sn } \nu \text{ dn } \lambda t \text{ dn } \nu)/\Delta \\ \text{sn } u &= \text{sn } (\lambda t-\nu) = (\text{sn } \lambda t \text{ cn } \nu \text{ dn } \nu - \\ &\quad \text{sn } \nu \text{ cn } \lambda t \text{ dn } \lambda t)/\Delta \\ \text{dn } u &= \text{dn } (\lambda t-\nu) = (\text{dn } \lambda t \text{ dn } \nu + \\ &\quad k^2 \text{ sn } \lambda t \text{ cn } \lambda t \text{ sn } \nu \text{ cn } \nu)/\Delta , \end{aligned} \quad (14)$$

where

$$\Delta = 1 - k^2 \text{ sn}^2 \lambda t \text{ sn}^2 \nu. \quad (15)$$

Since at t = 0, cn u = cn ν, sn u = − sn ν and dn u = dn ν, we have from Eqs. (7)

$$\begin{aligned} \text{cn } \nu &= \lambda_{x_o}/p \\ \text{sn } \nu &= \lambda_{y_o}/q \\ \text{dn } \nu &= 1/r . \end{aligned} \quad (16)$$

Thus, the solutions for λ_x, λ_y and λ_z may be expressed as

$$\begin{aligned} \lambda_x &= (\lambda_{x_o} \text{ cn } \lambda t - \lambda_{y_o} \, p/(qr) \text{ dn } \lambda t \text{ sn } \lambda t)/\Delta \\ \lambda_y &= (\lambda_{x_o} \, q/(pr) \text{ sn } \lambda t + \lambda_{y_o} \text{ cn } \lambda t \text{ dn } \lambda t)/\Delta \\ \lambda_z &= (\text{dn } \lambda t - [\lambda_{x_o} \lambda_{y_o} \, k^2/(pq)] \text{ sn } \lambda t \text{ cn } \lambda t)/\Delta , \end{aligned} \quad (17)$$

where $\Delta = 1 - (k^2 \lambda_{y_o}{}^2/q^2) \text{ sn}^2 \lambda t$. For use in Eqs. (16), p, q, r, λ and k are to be obtained by evaluating Eqs. (8)-(10), (12) and (13) at t=0 using Eqs. (5) and (6).

The angles θ and φ may be expressed as functions of time by using the fact that $\underline{H}$ may be written as

$$\underline{H} = A\omega_x \, \hat{i} + B\omega_y \hat{j} + C\omega_z \hat{k} , \quad (18)$$

or as

$$\underline{H} = - H \sin\theta \, \hat{i} + H \sin\phi \cos\phi \, \hat{j} + H \cos\phi \cos\theta \, \hat{k}, \quad (19)$$

where $\{\hat{i}, \hat{j}, \hat{k}\}$ is a set of unit vectors associated with the Cxyz system and $H = |\underline{H}|$. From Eqs. (18) and (19) and previous definitions, it follows that

$$\sin\theta = - \sigma_1 \lambda_x/h \quad (20)$$

and

$$\tan\phi = \sigma_2 \lambda_y/\lambda_z . \quad (21)$$

Expressing the angle ψ as a function of time is a more difficult task. Although it is well known[3] that ψ may be expressed as a linear function of time plus a periodic function of time composed of Jacobi's Theta functions, this was not done. Instead, ψ is calculated by integrating the following differential equation:

$$\psi' = h \, [1 + (\sigma_2 - 1) \sin^2 \phi]. \quad (22)$$

Note that if the rotational motion is uniform, $\phi = 0$ and $\psi = \psi_o + h\tau$. In the simulation program, a check is made to determine if the motion is uniform, and if so, the integration of Eq. (22) is by-passed and the orientation at the desired "time," τ, is computed using $\theta = \phi = 0$ and $\psi = \psi_o + h\tau$.

The transformation from a nonrotating set of unit vectors $\{\hat{I}, \hat{J}, \hat{K}\}$ directed along the positive x_s-, y_s- and z_s-axes, respectively, to the set of unit vectors $\{\hat{i},\hat{j},\hat{k}\}$ is

$$(\hat{i} \ \hat{j} \ \hat{k})^T = (\hat{I} \ \hat{J} \ \hat{K})^T \, \underline{\underline{A}}^T , \quad (23)$$

where a superscript T denotes the transpose and

$$\underline{\underline{A}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & s\phi \\ 0 & -s\phi & c\phi \end{bmatrix} \begin{bmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} c\psi & s\psi & 0 \\ -s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\times \begin{bmatrix} c\theta_H & 0 & -s\theta_H \\ 0 & 1 & 0 \\ s\theta_H & 0 & c\theta_H \end{bmatrix} \begin{bmatrix} c\psi_H & s\psi_H & 0 \\ -s\psi_H & c\psi_H & 0 \\ 0 & 0 & 1 \end{bmatrix} . \quad (24)$$

In Eq. (24), c denotes the cosine and s denotes the sine of the angle prefixed by such letter. Since ψ_H and θ_H are constant angles which specify the orientation of $\underline{H}$, they constitute two of the con-

stants of the motion. The other constants† are σ_1, σ_2, λ_{x_o}, λ_{y_o}, ω_{z_o} and ψ_o. Thus, we see that eight constants are needed to completely specify the free-Eulerian rotational motion of a particular body.

Translational Motion

The model adopted for the relative motion of the centers of mass of the body and the spacecraft is exceedingly simple. We assume that this motion is uniform so that the vector $\underline{R}$ from the point S to C is given by

$$\underline{R} = \underline{R}_o + \underline{V}t \ , \tag{25}$$

where

$$\underline{R}_o = X_o \hat{I} + Y_o \hat{J} + Z_o \hat{K} = \text{constant vector}$$

and

$$\underline{V} = \dot{X} \hat{I} + \dot{Y} \hat{J} + \dot{Z} \hat{K} = \text{constant vector},$$

are the relative position and velocity vectors at t = 0. The constant components of $\underline{R}_o$ and $\underline{V}$ are six constants which determine the relative translational motion.

Motion of a Point on the Body

The determination algorithms discussed in the next section are predicated on the use of observations of the movements of specific points on the surface of the tumbling body. If we let $\underline{r}_j$ be a vector from C to a point, P_j, on the surface of the body and $\underline{\rho}_j$ be a vector from S to P_j (see Figure 1) then we have, very simply,

$$\underline{\rho}_j = \underline{R} + \underline{r}_j \ . \tag{26}$$

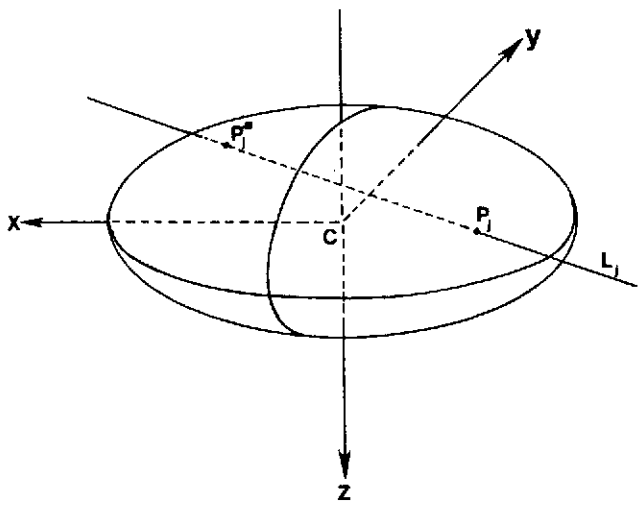

Fig. 3. Geometry for Visibility Determination.

† Actually, σ_1 and σ_2 are physical parameters.

The vectors $\underline{r}_j$ amd $\underline{\rho}_j$ may be expressed in the vector forms,

$$\underline{r}_j = x_j \hat{I} + y_j \hat{J} + z_j \hat{K} \tag{27}$$

and

$$\underline{\rho}_j = \xi_j \hat{I} + \eta_j \hat{J} + \zeta_j \hat{K} \ , \tag{28}$$

so that in matrix form,

$$\begin{Bmatrix} \xi_j \\ \eta_j \\ \zeta_j \end{Bmatrix} = \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} + \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} t + \underline{\underline{A}}^T \begin{Bmatrix} x_j \\ y_j \\ z_j \end{Bmatrix} \tag{29}$$

and

$$\begin{Bmatrix} \dot{\xi}_j \\ \dot{\eta}_j \\ \dot{\zeta}_j \end{Bmatrix} = \begin{Bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{Bmatrix} + \underline{\underline{A}}^T \tilde{\underline{\underline{\omega}}} \begin{Bmatrix} x_j \\ y_j \\ z_j \end{Bmatrix} \ , \tag{30}$$

where

$$\tilde{\underline{\underline{\omega}}} = \omega_{z_o} \begin{bmatrix} 0 & -\lambda_z & \lambda_y \\ \lambda_z & 0 & -\lambda_x \\ -\lambda_y & \lambda_x & 0 \end{bmatrix} \ . \tag{31}$$

Equations (29) and (31) are used in subroutine, STATE, to generate the relative position, $\underline{\rho}_j$, and relative velocity, $\dot{\underline{\rho}}_j$, of three points, P_j, j=1,2,3, on the surface of the ellipsoid. The range, ρ_j, and range-rate, $\dot{\rho}_j$, for each point are also determined from

$$\rho_j = (\xi_j^2 + \eta_j^2 + \zeta_j^2)^{\frac{1}{2}} \tag{32}$$

and

$$\dot{\rho}_j = \dot{\underline{\rho}}_j \cdot \underline{\rho}_j / \rho_j , \tag{33}$$

respectively.

As stated above, the points P_j are "on the surface" of the ellipsoid. This constraint is satisfied by allowing one to input only the x- and y-components of $\underline{r}_j$ and determining the z-component from

$$z_j = c \ (1 - x_j^2/a^2 - y_j^2/b^2)^{\frac{1}{2}} \ , \tag{34}$$

where a, b and c are to be provided and x_j and y_j are constrained by the relations $x_j \leq a$, $y_j \leq b$, $1 - x_j^2/a^2 - y_j^2/b^2 \geq 0$. Note that we have assumed that the points are all "on the bottom" (i.e., $z \geq 0$) of the ellipsoid.

Observability of a Point on the Ellipsoid

Clearly, a point P_j, on the ellipsoid will not, in general, be visible from a point S at all times. To determine whether or not P_j is visible, it is sufficient to find the two (in general) intersections of the line of sight, L_j, which is colinear with $\underline{\rho}_j$, with the ellipsoid (see Figure 3)

and determine if the point, P_j, is the near point of intersection. Such a determination is made in subroutines STATE and VISIB in the following manner.

The components of the vector, $\underline{\rho}_j$, in the $\{\hat{i},\hat{j},\hat{k}\}$ basis are determined using

$$\begin{Bmatrix} \xi_j^A \\ \eta_j^A \\ \zeta_j^A \end{Bmatrix} = \underline{\underline{A}} \begin{Bmatrix} \xi_j \\ \eta_j \\ \zeta_j \end{Bmatrix} \tag{35}$$

and the direction cosines, ℓ_{A_j}, m_{A_j} and n_{A_j}, of L_j are determined using the equations,

$$\begin{aligned} \ell_{A_j} &= \xi_j^A/\rho_j \\ m_{A_j} &= \eta_j^A/\rho_j \\ n_{A_j} &= \zeta_j^A/\rho_j \end{aligned} \tag{36}$$

The coordinates x_j^*, y_j^* and z_j^* of the "other" point of intersection of L_j with the ellipsoid must satisty the equations,

$$(x_j^* - x_j)/\ell_{A_j} = (y_j^* - y_j)/m_j = (z_j^* - z_j)/n_j \tag{37}$$

and

$$1 = (x_j^*/a)^2 + (y_j^*/b)^2 + (z_j^*/c)^2. \tag{38}$$

The solution of Eqs. (37) and (38) is[†]

$$\begin{aligned} x_j^* &= 2\, c_2/c_1 - x_j \\ y_j^* &= (m_{A_j}/\ell_{A_j})(x_j^* - x_j) + y_j \\ z_j^* &= (n_{A_j}/\ell_{A_j})(x_j^* - x_j) + z_j, \end{aligned} \tag{39}$$

where

$$\begin{aligned} c_1 &= \ell_{A_j}^2/a^2 + m_{A_j}^2/b^2 + n_{A_j}^2/c^2 \\ c_2 &= (c_1 - \ell_{A_j}^2/a^2)\, x_j - m_{A_j}\, \ell_{A_j} y_j/b^2 - n_{A_j} \ell_{A_j} z_j/c^2 \end{aligned} \tag{40}$$

We note that if $\ell_{A_j} = 0$ the second and third of Eqs. (39) appear to be singular. This singularity is, however, only "apparent," since

$$\lim_{\ell \to 0} \frac{(x_j^* - x_j)}{\ell_{A_j}} = \frac{-2[y_j(m_{A_j}/b^2) + z_j(n_{A_j}/c^2)]}{[m_{A_j}^2/b + n_{A_j}^2/c^2]} \tag{41}$$

Using the coordinates (x_j^*, y_j^*, z_j^*) the distance, ρ_j^*, may be computed and if $\rho_j^* > \rho_j$, the point, P_j, is visible, but if $\rho_j > P_j^*$, the point, P_j, is on the "backside" of the ellipsoid and cannot be observed.

In summary, the motions of three points on the ellipsoid are simulated digitally in subroutine, STATE, which utilizes Eqs. (29) and (30) (and all equations needed to evaluate those equations). Values of a, b and c, as well as $\underline{R}_o$, $\underline{V}$, θ_H, ψ_H, $\underline{\omega}_o = \omega_{z_o}(\lambda_{x_o}\hat{i} + \lambda_{y_o}\hat{j} + \hat{k})$, x_j and y_j, j=1,2,3, must be supplied to STATE by the user. As the ellipsoid rotates and translates, the visibility of the chosen points is checked by using Eqs. (39), (32) and (29). If the point, P_j, is not visible, a message to that effect is printed, but, since subroutine, STATE, was used only to check out the motion determination algorithms, the data requested by the algorithms is still supplied. In future simulation work, however, the visibility provision should be used to make the simulation more realistic by "cutting off" data from an unobservable point.

V. Algorithms for the Case of Uniform Motion

Because asteroids and cometary nuclei may logically be expected to be rotating uniformly and because the exact solution to the free-Eulerian motion problem is much more complex if the motion is not uniform, algorithms for determining the eleven constants, ψ_H, θ_H, $\omega = |\underline{\omega}|$, $\dot{X}$, $\dot{Y}$, $\dot{Z}$, X_o, Y_o, Z_o, x_1 and y_1, were developed. For the uniform motion case, ψ_H and θ_H determine the orientation of the target's angular velocity vector $\underline{\omega}$, which is, of course, colinear with its rotational angular momentum vector, $\underline{H}$, and ω is the magnitude of the angular velocity. The constants $\dot{X}$, $\dot{Y}$, $\dot{Z}$, X_o, Y_o and Z_o determine the translational motion of a point, C, on the spin axis of the target and x_1 and y_1 locate a point, P_1, with respect to C. The z-coordinate of P_1 is not determined, since the location of the center of mass of the target on the z-axis cannot be found if it does not "wobble." That is, if the motion is uniform, all points on the axis of rotation move in the same manner as the center of mass and hence are not distinguishable from it. This fact is taken into account by setting $z_1 = 0$. Furthermore, since the orientation of the Cxyz system in the asteroid is immaterial if the motion is uniform, ψ_o can be set equal to zero also.

Two distinct algorithms were developed for the uniform motion case. The first is called DUMRA (Determination of the Uniform Motion using Range and Angles). This algorithm is presently set up to accept position vectors, which it is assumed may be obtained from range-and-angle data for three specific points on the target's surface. The second algorithm is called DUMRRR (Determination of the Uniform Motion using Range and Range-Rate). This algorithm utilizes range and range-rate data

[†] A quadratic equation in x_j^* may be obtained. One root of this equation is obviously x_j, so that, the first of Eqs. (39) may be obtained by factoring the quadratic.

for one point on the target's surface along with estimates of the eleven constants to be determined and iteratively produces more exact values for the constants. More detailed descriptions of these algorithms are given in the remainder of this section.

Algorithm DUMRA

The solution procedure utilizing position data is as follows:

1. Obtain position vectors, $\underline{\rho}_j$, j=1,2,3, for three distinct, non-colinear points, P_j, j=1,2,3, on the surface of the target at three times, t_j, j=1,2,3, such that $t_2 - t_1 = t_3 - t_2 = \Delta t$.

2. Compute approximate velocity vectors for each point at time, t_2, using

$$\dot{\underline{\rho}}_j(t_2) = [\underline{\rho}_j(t_3) - \underline{\rho}_j(t_1)]/2\Delta t, \quad j=1,2,3. \tag{42}$$

3. Compute the angular velocity of the body, $\underline{\omega}$, using[5]

$$\underline{\omega} = \frac{[\dot{\underline{\rho}}_3(t_2) - \dot{\underline{\rho}}_2(t_2)] \times [\dot{\underline{\rho}}_2(t_2) - \dot{\underline{\rho}}_1(t_2)]}{[\dot{\underline{\rho}}_3(t_2) - (\dot{\underline{\rho}}_2(t_2)] - [\underline{\rho}_2(t_2) - \underline{\rho}_1(t_2)]}, \tag{43}$$

so that in matrix form,

$$\underline{\omega} = [\omega_1 \ \omega_2 \ \omega_3]^T. \tag{44}$$

4. Compute ψ_H and θ_H using

$$\psi_H = \tan^{-1}(\omega_2/\omega_1) \tag{45}$$

and

$$\theta_H = \tan^{-1}(\sqrt{1-\cos^2\theta_H}/\cos\theta_H), \tag{46}$$

where

$$\cos\theta_H = \omega_3/|\underline{\omega}|. \tag{47}$$

5. Compute x_1 and y_1. First, compute

$$\Delta\underline{\rho}_1 = \underline{\rho}_1(t_3) - 2\,\underline{\rho}_1(t_2) + \underline{\rho}_1(t_1) \tag{48}$$

in the matrix form,

$$\Delta\underline{\rho}_1 = \underline{\underline{A}}_3^T \underline{\underline{A}}_2^T [\underline{\underline{A}}^T(t_3) - 2\,\underline{\underline{I}} + \underline{\underline{A}}^T(t_1)] \begin{matrix} x_1 \\ y_1 \\ 0 \end{matrix}, \tag{49}$$

where

$$\underline{\underline{A}}_1(t) = \begin{bmatrix} c\omega t & s\omega t & 0 \\ -s\omega t & c\omega t & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{50}$$

$$\underline{\underline{I}} = 3 \times 3 \text{ identity matrix}, \tag{51}$$

$$\underline{\underline{A}}_2 = \begin{bmatrix} c\theta_H & 0 & -s\theta_H \\ 0 & 1 & 0 \\ s\theta_H & 0 & c\theta_H \end{bmatrix}, \tag{52}$$

$$\underline{\underline{A}}_3 = \begin{bmatrix} c\psi_H & s\psi_H & 0 \\ -s\psi_H & c\psi_H & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{53}$$

and t_2 has been arbitrarily set equal to zero. Equation (48) may be solved for x_1 and y_1, viz.,

$$\begin{Bmatrix} x_1 \\ y_1 \end{Bmatrix} = \begin{bmatrix} [2(c\omega\Delta t - 1)]^{-1} & 0 & 0 \\ 0 & [2(c\omega\Delta t - 1)]^{-1} & 0 \end{bmatrix} \underline{\underline{A}}_2 \underline{\underline{A}}_3 \, \Delta\underline{\rho}_1. \tag{54}$$

6. Compute $\underline{R}(t_2) \equiv \underline{R}_o$. This may be done using the equation

$$\underline{R}_o = \underline{\rho}_1(t_2) - \underline{\underline{A}}_3^T \underline{\underline{A}}_2^T \begin{Bmatrix} x_1 \\ y_1 \\ 0 \end{Bmatrix}. \tag{55}$$

7. Compute $\underline{V}_o$ from the equation,

$$\underline{V}_o = [\underline{\rho}(t_1) - \underline{R}_o - \underline{\underline{A}}_3^T \underline{\underline{A}}_2^T \underline{\underline{A}}_1^T(t_1) \begin{Bmatrix} x_1 \\ y_1 \\ 0 \end{Bmatrix}]/\Delta t. \tag{56}$$

The requisite constants are given by Eqs. (43), (45), (46), (54), (55) and (56). Note that to extract the coordinates x_1 and y_1, it was necessary to allow for the curvature of the motion of P_1 due to rotation.

Algorithm DUMRRR

The solution procedure based on the use of range and range-rate data is an iterative one. The main steps are as follows:

1. Guess values for the eleven unknowns, ψ_H, θ_H, ω, X_o, Y_o, Z_o, $\dot{X}$, $\dot{Y}$, $\dot{Z}$, x_1 and y_1.

2. Set $\psi = 0$ at $t = 0$ and compute, for six times $t_j > 0$,

$$\rho_{1_e} = (\underline{R}^T \underline{R} + 2\,\underline{R}^T \underline{\underline{A}}\,\underline{r}_1 + \underline{r}_1^T \underline{r}_1)_e \tag{57}$$

and

$$(\dot{\rho}_1\rho_1)_e = (\underline{V}_o^T \underline{R} + \underline{R}^T \underline{\underline{A}}^T \tilde{\underline{\underline{\omega}}}\,\underline{r}_1 + \underline{V}_o^T \underline{\underline{A}}^T \underline{r}_1)_e, \tag{58}$$

where the subscript e denotes an estimated value,

$$\tilde{\underline{\underline{\omega}}} = \begin{bmatrix} 0 & -\omega & 0 \\ \omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}_e \tag{59}$$

and $\underline{\underline{A}}^T = \underline{\underline{A}}_3^T \underline{\underline{A}}_2^T \underline{\underline{A}}_1^T$.

3. Use the observations $\rho_1 = |\underline{\rho}_1|$ and $\dot{\rho}_1 = \dot{\underline{\rho}}_1/\rho_1$ at the same times, t_j, and the estimated values to obtain

$$\Delta\underline{z} = \begin{Bmatrix} \rho_1 - \rho_{1e} \\ \dot{\rho}_1\rho_1 - (\dot{\rho}_1\rho_1)_e \end{Bmatrix} \tag{60}$$

at each time, t_j, j=1,2,...,6.

4. Evaluate the partial derivative matrix,

$$\underline{\underline{H}}(t_j) = \begin{bmatrix} \frac{\partial \rho_1}{\partial \underline{x}} \\ \vdots \\ \frac{\partial \dot{\rho}_1 \rho_1}{\partial \underline{x}} \end{bmatrix} \tag{61}$$

where $\underline{x} = (\psi_H\ \theta_H\ \omega\ X_o\ Y_o\ Z_o\ X\ Y\ Z\ x_1\ y_1)^T$, at each time, t_j, j=1,2,...,6.

5. Form the augmented matrices,

$$\underline{\underline{B}} = \begin{bmatrix} \underline{\underline{H}}(t_1) \\ \hline \underline{\underline{H}}(t_2) \\ \hline \vdots \\ \underline{\underline{H}}(t_5) \\ \hline (\frac{\partial \rho_1}{\partial \underline{x}})_{t=t_6} \end{bmatrix} \tag{62}$$

and

$$\underline{C} = [\underline{\Delta z}^T(t_1)\ ;\ \underline{\Delta z}^T(t_2)\ ;\ \ldots\ ;\ \underline{\Delta z}^T(t_5)\ ;\ (\rho_1 - \rho_{1_e})_{t=t_6}]^T \tag{63}$$

6. Invert $\underline{\underline{B}}$ and compute

$$\underline{\Delta x} = \underline{\underline{B}}^{-1}\ \underline{C}\ C_f\ , \tag{64}$$

where C_f is a convergence factor ($C_f \leq 1$) which may be varied to improve convergence.

7. Find a new estimate of $\underline{x}$ using

$$\underline{x}_e^{i+1} = \underline{x}_e^{i} + \underline{\Delta x}\ . \tag{65}$$

8. Check the norm, $||\ \underline{C}\ ||$, to see if it is sufficiently small. If it is, go to 10.

9. To decrease $||\ \underline{C}\ ||$, compute a new partial derivative matrix $\underline{\underline{B}}$ and a new matrix $\underline{C}$ using the current estimates and repeat steps 6 through 8.

10. The elements of $\underline{x}_e^{i+1}$ are the desired values of the constants.

The matrix $\underline{\underline{B}}$ in Eq. (62) is such that some of it its terms are much larger than others. Therefore, to increase numerical accuracy in obtaining the inverse of $\underline{\underline{B}}$, dimensionless variables are used in the computer subroutine DERIV in which the matrices $\underline{\underline{H}}(t_j)$ are computed.

It should be noted that the method of solution used in this algorithm is of the "batch processor" type. Another method for solving such non-linear observation equations as Eqs. (57) and (58) is to use the pseudo-inverse,

$$\underline{\underline{D}} = (\underline{\underline{H}}^T\ \underline{\underline{H}})^{-1}, \tag{66}$$

and compute $\underline{\Delta x}$ from the equation,

$$\underline{\Delta x} = \underline{\underline{D}}\ \underline{\underline{H}}^T\ \underline{\Delta z}\ , \tag{67}$$

at successive instants of time, correcting the estimate, $\underline{x}_e$, each time. This method of solution was attempted initially when there were still errors in the program. Since it appeared that the matrix $\underline{\underline{H}}^T\underline{\underline{H}}$ was ill-conditioned, the pseudo-inverse method was abandoned in favor of the more cumbersome, but more trustworthy, "batch-processor" method outlined here.

VI. Performance of the Algorithms

The algorithms, DUMRA and DUMRRR, were tested using the simulation capability provided by subroutine, STATE, (and subsidiary subroutines). They were not exercised as much as was desired because of time constraints, but whether they operate satisfactorily in the uniform motion cases considered is evident from the results discussed in this section. Nonuniform motion due to target body "wobble" is another matter and is also discussed infra.

To test the performance of the algorithms, values of the physical parameters a, b and c thought to be typical of an asteroid were chosen. These values are given in Table 1 along with values of the nine constants of the motion and the

TABLE 1
PHYSICAL PARAMETERS AND CONSTANTS OF THE MOTION

$a = 1.75 \times 10^4$ m	$\omega = 3.31 \times 10^{-4}$ rad/sec
$b = 8.0 \times 10^3$ m	$\theta_H = 20°$
$c = 3.5 \times 10^3$ m	$\psi_H = 10°$
$X_o = -1.0 \times 10^6$ m	$u = 1.0$ m/sec
$Y_o = 0.0$	$v = .5$ m/sec
$Z_o = 0.0$	$w = .5$ m/sec
$x_1 = -1.75 \times 10^4$ m	$y_1 = 0.0$
$x_2 = -1.0 \times 10^4$ m	$y_2 = 6.5652 \times 10^4$ m
$x_3 = -1.0 \times 10^4$ m	$y_3 = 0.0$

x- and y-coordinates of three points on the body. The z-coordinates of these points are determined in STATE as indicated previously. The values of the constants which are listed constitute only one of several sets of values used during the check-out process and hence are "typical."

The first step in the check-out of each of the two algorithms was to test its accuracy under ideal conditions represented by perfectly uniform motion. The performance of each algorithm when the simulated motion was not uniform was then tested.

For perfectly uniform motion, algorithm, DUMRA, reproduced the input in most cases to four significant figures and always attained percent errors less than 1 percent. Theoretically, the only error which is present (apart from basic computational errors, that is, is that due to the approximate calculation of the velocity of each of the points from the position data. This error does not appear sufficiently large to be a problem if the increment in time, Δt, is small compared to the rotational period of the target. In the cases examined here, $\omega = 3.31 \times 10^{-4}$ rad/sec, so that the period of rotation is 1.9×10^{4} sec $\approx$ 5.28 hrs. For these numerical values, Δt = 90 sec., gave satisfactory results.

When nonuniformity of the rotational motion was introduced by giving ω_{x_o} and/or ω_{y_o} non-zero values, the effect on algorithm DUMRA was anomalous when the maximum value of the angle, $\Theta = \cos^{-1}(\cos\theta \cos\phi)$, between the rotational angular momentum vector and the angular velocity vector was greater than a few degrees. The algorithm extracted fairly accurate (within 10 percent) values for the components of $\underline{R}_o$ and $\underline{V}_o$ at times, but failed as often as it was successful. The reason for this behavior has not been determined at this time, but appears to be connected with the use of the position vectors to calculate velocity vectors.

The performance of algorithm, DUMRRR, was not satisfactory. In the runs made, the use of the algorithm usually resulted in the reduction of square root of the sum of the squares of the elements of the matrix, $\underline{C}$, but this reduction did not lead to consistent improvements in all the constants which must be determined. The errors in $\underline{R}_o$ were, for example, reduced by applying the algorithm, but those in $\underline{V}_o$ and $\underline{r}$, in general were not. In a few cases, application of the algorithm resulted in progressively smaller values of $||\underline{C}||$ for several iterations, but continued applications resulted in divergence.

Attempts were made to improve the convergence properties of the algorithm by varying the convergence factor, C_f. An initial value of 0.02 was given to C_f and this value increased to 0.1 and 1.0 when convergence criterion was reduced to 10^{-4} and 10^{-6}, respectively. These efforts were not successful.

Since the algorithm, DUMRRR, did not perform satisfactorily when confronted with uniform motion, tests with nonuniform rotational motion were not conducted.

Conclusions and Recommendations

In summary, the capability for simulating the motion of an asteroidal-type body and observations of points on such a body has been developed. The rotational motion simulated may be general free-Eulerian or uniform, while the translational motion is uniform. This simulation capability has been used to test two algorithms for determining the constants of the motion from simulated observations when the rotational motion is uniform.

The algorithm, DUMRA, performed the job for which it was designed satisfactorily. However, when required to work in the presence of "noise" caused by nonuniform rotational motion, the algorithm produced erroneous results. Failure of the algorithm for large nutation angles (angle Θ) was expected, but even small angles unexpectedly caused a great deal of trouble. Because, small nutation angles may actually be encountered, the algorithm should be modified to allow for such an eventuality. This could be done by using the angular velocity determination portion of the algorithm with a small Δt to obtain five values of $\underline{\omega}$. An approximate solution which allows for small nutation angles (see Ref. 5, Appendix D) could then be used to obtain ω_{x_o}, ω_{y_o}, ω_{z_o}, σ_1 and σ_2. Minor changes in DUMRA would then allow the center of mass location and motion to be determined.

The work on algorithm, DUMRRR, based on the use of range and range-rate data, although not successful, should not be considered to have been fruitless. The failure of this algorithm reinforced an intuitive belief that more information than just range and range-rate data are needed to determine the motion. In particular, it appears that angular data is necessary to adequately define the plane of rotation of the vector $\underline{r}_1$ and hence the direction of $\underline{\omega}$. It was hoped that the requirement for such data could be eliminated by having very good estimates of the angles ψ_H and θ_H, but this does not seem to be the case.

An important aspect of this study is that an observer was assumed to be available for the purpose of identifying points and selecting data. Totally autonomous operation of course precludes the use of such an observer. Thus, some type of automatic identification technique should be developed if a totally autonomous determination of the target body's motion is a requirement of actual missions.

References

1. Meissinger, H. F., and Greenstadt, E. W., "Design and Science Instrumentation of an Unmanned Vehicle for Sample Return from Asteroid Eros," Paper presented at Asteroid Conference, Tucson, Arizona, March 8-10, 1971.

2. Kopal, Zdeněk, "The Axial Rotation of Asteroids," Astrophysics and Space Science, Vol. 6, 1970, pp. 33-35.

3. Whittaker, E. T., A Treatise on Analytical Dynamics of Particles and Rigid Bodies, Cambridge University Press, New York, 1970, pp. 144-152.

4. Byrd, P. F., and Friedman, M. D., Handbook of Elliptic Integrals for Engineers and Physicists. Springer-Verlag, Berlin, 1954, p. 23.

5. Cochran, John E., "First Year Report, Study of Effects of Uncertainties on Comet and Asteroid Encounter and Contact Guidance Requirements," Contract NAS8-27664, June 1972, pp. 93-94.

APPENDIX - COMPUTER PROGRAMS

General Program Descriptions

The DUMRA and DUMRRR programs are written in FORTRAN IV and executed on the IBM 370/155. The programs are research tools, not developed production routines. The steps in the simulation, and the names of the subroutines used, are briefly as follows.

DUMRA

Values of time, T, and the time increment, DT, are input to subroutine, MAIN, which calls subroutine, DUMRA, which calls subroutine, STATE, at times, T1=T, T2=T + DT and T3=T + 2*DT. STATE receives input from subroutine, DATA, the first time it is called and produces the position vector sets, R1 (I,J), R2 (I,J), R3 (I,J), I = 1,3, J = 1, 3, for times, T1, T2 and T3, respectively, Here, I is the coordinate index; i.e., I = 1, 2 or 3 implies the x-, y- or z-coordinate, respectively, of a point, P_j. The integer J is the number of the point. The position vector sets are used to determine the constants of the motion. In both STATE and DUMRA matrix algebra subroutines are used to perform the necessary calculations. Subroutine STATE, also calls DJELF and DCEL1 during the simulation process.

DUMRRR

The MAIN subroutine of this program executes the DUMRRR algorithm described in the body of this report. Subroutine, MAIN, accepts a time, T, a time increment, DT, and estimates of the constants of the motion. It calls subroutine, RHOES, which in turn calls STATE and computes a difference matrix DZ (I,1), I = 1,2, formed from the differences in the actual range, RHO, and range times range-rate, and the estimated values of range, ROE, and range times range-rate, RORODE, respectively, for the times at which it is called. Subroutine, DERIV, is also called from MAIN for each of the six times necessary and computes the matrix H(I,J), I = 1,2,...,11, J = 1,2 of partial derivatives of the observations with respect to the constants being determined. Matrix algebra subroutines are called from MAIN, RHOES, STATE and DERIV as needed. The observations are stacked in an 11 x 1 vector, C(I,1), and the partial derivatives are stacked in the 11 x 11 matrix, B(I,J). Increments in the constants of the motion are calculated as an 11 x 1 vector, DX(I,1), and new estimates obtained. The algorithm is repeatedly executed until the norm of the differences in the actual and estimated observations is "sufficiently" small.

Subroutine Names and Descriptions†

DUMRA

MAIN Controls execution and reads in T and DT.

STATE Simulates the free-Eulerian Rotational motion and the uniform translational variation, produces observational data and checks the visibility of the observed points.

DUMRA Determines the eleven constants, PSIH, THETAH, WW, $\underline{R}_o$, $\underline{V}_o$, x_1 and y_1.

MATMUL Matrix multiplication subroutine.

TILDE Matrix algebra subroutine which forms a skew-symmetric, 3 x 3 matrix from a 3 x 1 matrix (vector) for computation of a cross-product.

DOTPRO Matrix algebra subroutine which forms the dot product of two vectors.

DATA Reads in data needed in STATE.

VISIB Checks data from STATE and prints out a message if any of the three points being observed is(are) not visible.

DJELF Computes the Jacobian elliptic functions snu, cnu and dnu.

CELI Computes the complete elliptic integral of the first kind of modulus k.

Entries are indicated in the flowchart in Figure A-1.

DUMRRR

MAIN Controls execution and executes the algorithm, DUMRRR.

STATE Same as for DUMRA except for minor modifications.

RHOES Computes the difference $\underline{\Delta z}$ in the actual and estimated observations.

DOTPR Same as for DUMRA.

TRPOSE Transposes an input matrix.

TILDE See DUMRA above.

MATMUL See DUMRA above.

DATA Reads in actual data.

A37IMS Inverts matrix B.

DJELF See DUMRA above .

DECL1 See DUMRA above.

Entries are indicated in the flowchart in Figure A-2.

†Subroutines are used in both DUMRA and DUMRRR.

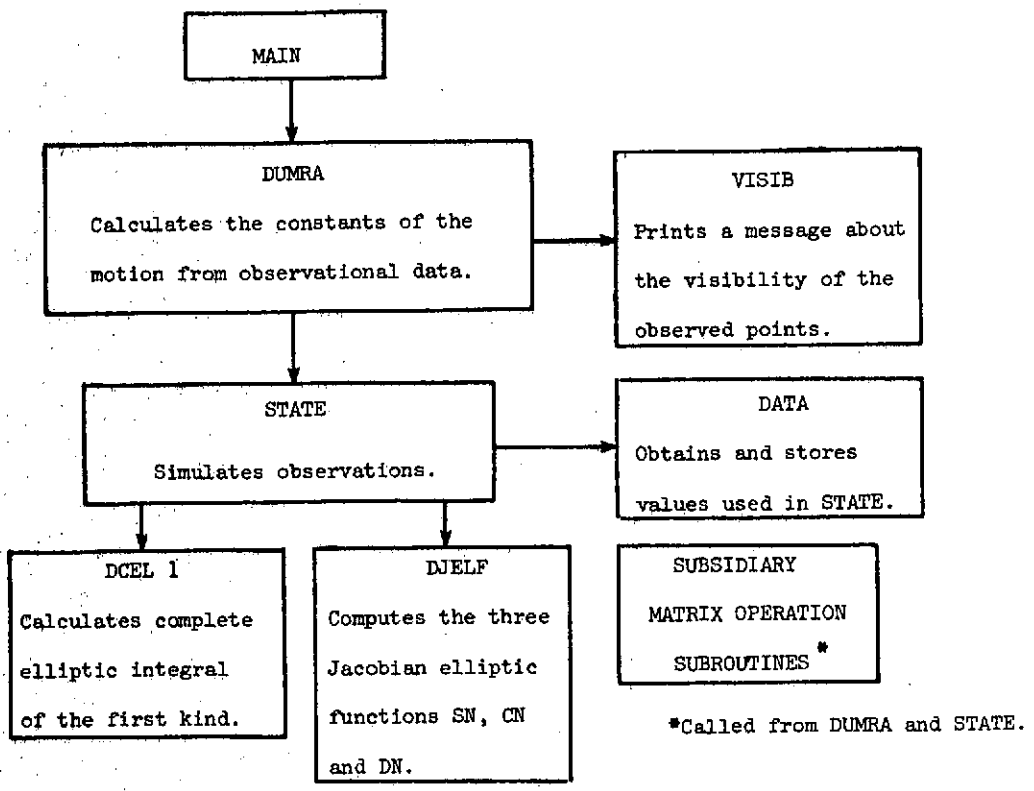

Fig. A-1. Simplified Flowchart for DUMRA

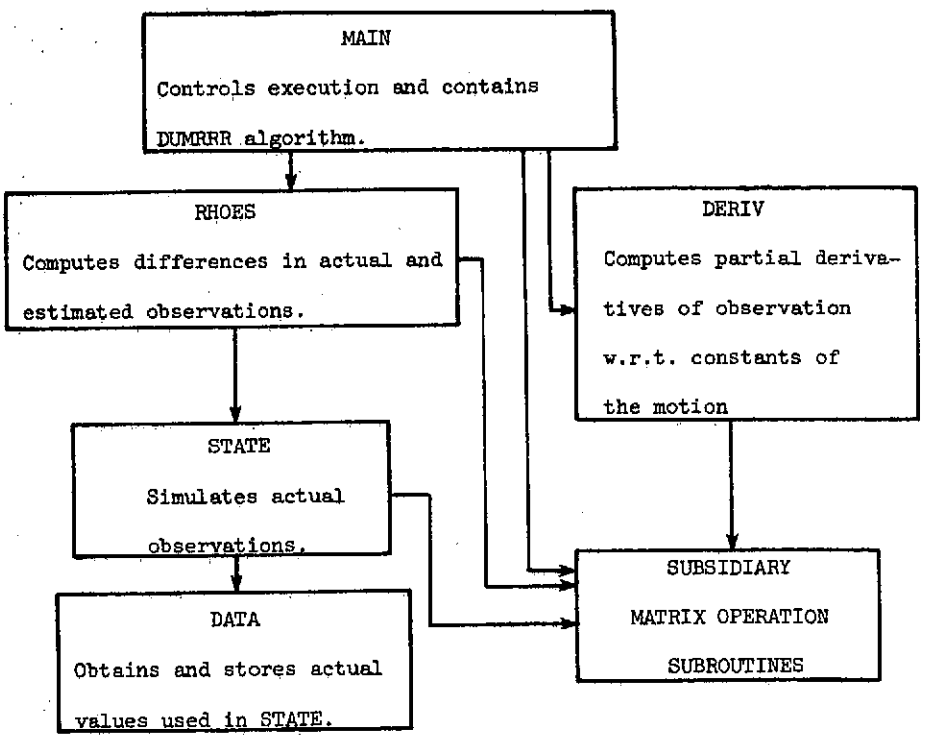

Fig. A-2. Simplified Flowchart for DUMRRR

Variable Names and Definitions

The most important variables in regard to INPUT and OUTPUT are defined in the sections which follow. Other variable names are defined throughout this appendix.

Input Data

The data which must be provided as input to programs DUMRA and DUMRRR is described below. All field specifications are F16.0 and the card column in which a particular piece of data must be located are indicated by numbers in parentheses following the mathematical symbol for the piece of data.

DUMRA

Card No.	Information on Card
1	Initial time and time increment: t_c(cc1-16), Δt(cc 17-32).
2	Constants of the Motion: ω_{x_o} (cc 1-16), ω_{y_o} (cc 17-32) ω_{z_o} (cc 33-48), ψ_o(cc 49-64), θ_H(cc 65-80).
3	Constants of the Motion: ψ_H(cc 1-16) $\dot{X}$(cc 49-64), X_o(cc 65-80).
4	Constants of the Motion: Y_o (cc 1-16), Z_o(cc 17-32). Coordinates of Points: x_1(cc 49-64), y_1(cc 65-80), x_2(cc 65-80)
5	Coordinates of Points: y_2(cc 1-16), x_3(cc 17-32), y_3(cc 33-48).
6	Physical Parameters: a(cc-1-16), b(cc 17-32), c(cc 33-48). (Semi-axes of ellipsoid)

DUMRRR

Card No.	Information on Card
1	Initial Time and Time Increment: TC(BC 1-16), DT(cc 17-32) .
2	Estimated Values of: ψ_H(cc 1-16), θ_H(cc 17-33), ω(cc 33-48), X_o(cc 49-64), Y_o(cc 65-80) .
3	Estimated Values of: Z_o(cc 1-16), $\dot{X}$(cc 17-32), $\dot{Y}$(cc 33-48), $\dot{Z}$(cc 49-64), x_1(cc 65-80) .
4	Estimated Value of: y_1(cc 1-16)
5	Actual Values of: ω_{x_o} (cc 1-16), ω_{y_o} (cc 17-32), ω_{z_o} (cc 33-48), ψ_o(cc 49-64), θ_H(cc 65-80)
6	Actual Values of: ψ_H(cc 1-16), $\dot{X}$(cc 17-32), $\dot{Y}$(cc 33-48), $\dot{Z}$(cc 49-64), X_o(cc 65-80)
7	Actual Values of: Y_o(cc 1-16), Z_o(cc 17-32), x_1(cc 33-48). y_1(cc 49-64), x_2(cc 65-80)
8	Actual Values of: y_2(cc 1-16), x_3(cc 17-32), y_3(cc 33-48)
9	Physical Parameters: a(cc 1-16), b(cc 17-32), c(cc 33-48))

Output Descriptions

The outputs of the programs, DUMRA and DUMRRR, are self-explanatory if one refers to following definitions of quantities which appear on the printouts.

DUMRA

WXO, WYO, WZO ≡ initial values of the components of angular velocity in the Cxyz system.

PSIO ≡ initial value of ψ .

THTAH ≡ θ_H.

PSIH ≡ ψ_H.

XD, YD, ZD ≡ X, Y, Z, respectively.

X, Y, Z ≡ X_o, Y_o, Z_o, respectively.

SX, SY, SZ ≡ x-, y- and z-coordinates, respectively, of the point indicated.

PHI ≡ ϕ .

THETA ≡ θ .

WW ≡ $|\underline{\omega}|$.

W ≡ angular velocity velocity, $\underline{\omega}$. The x_s-, y_s- and z_s-components of $\underline{\omega}$ are listed under W.

V ≡ $\underline{V}_o$ - The components, $\dot{X}$, $\dot{Y}$ and $\dot{Z}$, are listed under V.

RO ≡ $\underline{R}(t_2)$, the vector $\underline{R}$ at time, TC + DT. The components of $\underline{R}$ are listed under RO.

SR1, SR2, SR3 ≡ the components of $\underline{r}_1$, $\underline{r}_2$, and $\underline{r}_3$, respectively, in a coordinate system rotated with respect to the Cxyz system so that ψ = 0 at time, TC + DT, and translated so that z_3 = 0 are printed underneath these symbols.

The errors, $\frac{\text{actual-estimated}}{\text{actual}}$, in the values for the components of $\underline{R}(t_2)$, $\underline{r}_1$, $\underline{r}_2$, $\underline{r}_3$ and $\underline{V}_o$ computed

using simulated values as true values, are printed also under the headings DRO, DSR 1, DSR 2, DSR 3, and DVO, respectively. Similar errors in the components of $\dot{\underline{r}}_1$, $\dot{\underline{r}}_2$ and $\dot{\underline{r}}_3$ are also given under the headings DRDOT 11, DRDOT 2 and DRDOT 3, respectively. It should be noted that when 0.999 D03 is printed this does not indicate a 99,900 percent error, but is an indication that the actual value and the calculated values are both zero.

DUMRRR

The output of DUMRRR includes the values of the constants of the motion and the other data needed for the simulation of the motion in the same output format as that used in DUMRA. Additional output includes the estimated values of the constants of the motion and the coordinates of P_1, the times, T, at which data is taken, the vector $\underline{C}$ formed from the actual and estimated observations, the estimated values of the constants PSIH, THETAH, WW (WE on printout), $\underline{R}_o$, $\underline{V}_o$ and x_1 and y_1 and the matrix, DX, of changes in the constants.

DUMRA Program Listing

```
      IMPLICIT REAL*8(A-H,O-Z)
      NN=1
      READ(5,1)TC,DT
    1 FORMAT(2F16.0)
      DO 100 I=1,100
      NC=1
      CALL DUMRA(TC,DT,NC,NN)
      TC=TC+DT
  100 CONTINUE
      STOP
      END
```

```
      SUBROUTINE DUMRA(T1,DT,NC,NN)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION VS(3,1),ROS(3,1),SRRS(3,3),RDOTS(3,3),DSR(3,3
     .),DV(3,1),DRO(3,1),DRDOT(3,3)
      DIMENSION R1(3,3),SR1(3,1),SR2(3,1),SR3(3,1),RDOT(3,3)
      DIMENSION WI(3),SRI(3,3),RHO(3),RHOD(3),R2(3,3)
      DIMENSION R3(3,3),DR1(3,3),DR2(3,3),RDDOT(3,3),W(3,1)
      DIMENSION DELR1(3,1),H(3,3),DELRD1(3,1),DELRD2(3,1)
      DIMENSION ATAP(3,3),ATH(3,3),DELT(3,3),VCV(3,1),V(3,1)
      DIMENSION WT(3,3),RD(3,1),V2(3,1),RO(3,1),SR(3,3)
      DIMENSION SRI1(3,1),SRI2(3,1),SRI3(3,1)
      DIMENSION ATAPT(3,3),A3(3,3),A3T(3,3),SR1T3(3,1)
      DIMENSION A3T1(3,3),A3T3(3,3),DATI(3,3),DDR1(3,1)
      DIMENSION E(3,3),AT3(3,3),AT1(3,3),XI(3)
      DIMENSION DRI1(3,1),D2R1(3,1)
      DIMENSION SR11(3,1)
      DIMENSION SSR(3,3),VS1(3,1),ROS1(3,1)
      DIMENSION AB(3,1),BC(3,3)
      DO 210 I=1,3
      DO 209 J=1,3
      R1(I,J)=0.0D0
      R2(I,J)=0.0D0
  209 R3(I,J)=0.0D0
      V2(I,1)=0.0D0
  210 V(I,1)=0.000
C
C     OBTAIN DATA
C
      CALL STATE(R1,SR,SRI,RDOT,WI,DPSI,T1,T,NC,IVIS,RHO,RHOD
     .,TTT,PPP,NN,VS1,ROS1,SSR)
      CALL VISIB(IVIS)
      PRINT 947,T
      T2=T1+DT
      NC=3
      CALL STATE(R2,SR,SRI,RDOT,WI,DPSI,T2,T,NC,IVIS,RHO,RHOD
     .,TTT,PPP,NN,VS1,ROS1,SSR)
      DO 116 I=1,3
      VS(I,1)=VS1(I,1)
      ROS(I,1)=ROS1(I,1)
      DO 116 J=1,3
      SRRS(I,J)=SSR(I,J)
  116 RDOTS(I,J)=RDOT(I,J)
      PRINT 200,DT,T,T2
  200 FORMAT(/,5X,'DT = ',F12.6,10X,'T = ',F12.6,10X,'T2 = ',
     .F12.6,/)
      CALL VISIB(IVIS)
      PRINT 310,DPSI
      T3=T1+2.0*DT
```

```
      CALL STATE(R3,SR,SRI,RDOT,WI,DPSI,T3,T,NC,IVIS,RHO,RHOD
     .,TTT,PPP,NN,VSI,ROSI,SSR)
      CALL VISIB(IVIS)
      PRINT 947,T
  947 FORMAT(5X,'T=',D16.8,/)
  310 FORMAT(10X,'PSI = ',D20.8,//)
C
C     FIND RDOT AT TIME T2
C
      DO 101 I=1,3
      DO 101 J=1,3
      DR1(I,J)=0.0
      DR2(I,J)=0.0
  101 RDOT(I,J)=0.0
      DO 10 J=1,3
      DO 10 I=1,3
      DR1(I,J)=(R2(I,J)-R1(I,J))/DT
      DR2(I,J)=(R3(I,J)-R2(I,J))/DT
   10 RDOT(I,J)=(DR1(I,J)+DR2(I,J))/2.0D0
C
C      FIND W
C
      IF(DT.GT.100.0)GO TO 543
      DO 11 I=1,3
      DELR1(I,1)=0.0D0
      DELRD1(I,1)=0.0D0
      DELRD2(I,1)=0.0D0
      DELR1(I,1)=R2(I,2)-R2(I,1)
      DELRD1(I,1)=RDOT(I,2)-RDOT(I,1)
      DELRD2(I,1)=RDOT(I,3)-RDOT(I,2)
      DOTP=0.0D0
      VCV(I,1)=0.0D0
   11 XI(I)=DELRD2(I,1)
      CALL TILDE(XI,DELT)
      CALL MATMUL(DELT,DELRD1,VCV,3,3,1)
      CALL DOTPRO(DELRD2,DELR1,DOTP)
      DO 15 I=1,3
   15 W(I,1)=VCV(I,1)/DOTP
      WS=W(1,1)*W(1,1)+W(2,1)*W(2,1)+W(3,1)*W(3,1)
C
C     FORM APSI AND ATH
C
  543 WW=DSQRT(WS)
      PRINT 544,WW
  544 FORMAT(/,5X,'WW = ',D20.8,/)
      DSI=DSIN(WW*DT)
      DCO=DCOS(WW*DT)
      CTH=W(3,1)/WW
      STH=DSQRT(1.000-CTH*CTH)
      TH=DATAN(STH/CTH)
      DTH=TH*180.0/3.14159
      IF(W(2,1).LT.1.0E-6)GO TO 27
      PSIH=DATAN2(W(2,1),W(1,1))
      GO TO 28
   27 PSIH=0.0
   28 PSIHD=PSIH*180.0/3.141592
      CP=DCOS(PSIH)
      SP=DSIN(PSIH)
      DO 22 I=1,3
      DO 22 J=1,3
      ATH(I,J)=0.0D0
   22 H(I,J)=0.0D0
      ATH(2,2)=1.0D0
      ATH(1,1)=CTH
      ATH(1,3)=-STH
      ATH(3,1)=STH
      ATH(3,3)=CTH
      H(3,3)=1.0D0
      H(1,1)=CP
      H(1,2)=SP
      H(2,1)=-SP
      H(2,2)=CP
      CALL MATMUL(ATH,H,ATAP,3,3,3)
      DO 120 I=1,3
```

```
      DO 120 J=1,3
      A3(I,J)=0.0
  120 ATAPT(I,J)=ATAP(J,I)
C     FORM A3T1 AND A3T3
C
      DO 30 I=1,3
      DO 30 J=1,3
      E(I,J)=0.0D0
      A3T1(I,J)=0.0D0
   30 A3T3(I,J)=0.0D0
      A3T1(1,1)=DCO
      A3T3(1,1)=DCO
      A3T1(1,2)=DSI
      A3T3(1,2)=-DSI
      A3T1(2,1)=-DSI
      A3T3(2,1)=DSI
      A3T1(2,2)=DCO
      A3T3(2,2)=DCO
      A3T1(3,3)=1.0D0
      A3T3(3,3)=1.0D0
      DO 31 I=1,3
      DDR1(I,1)=0.0D0
      D2R1(I,1)=0.0D0
   31 E(I,1)=1.0D0
C
C     FORM AT1 AND AT3
C
      CALL MATMUL(ATAPT,A3T1,AT1,3,3,3)
      CALL MATMUL(ATAPT,A3T3,AT3,3,3,3)
C
C     FORM D
C
      D=2.0D0*DCO-2.0D0
C
C     FORM DDR1
C
      DO 33 I=1,3
   33 DDR1(I,1)=R3(I,1)-2.0D0*R2(I,1)+R1(I,1)
C
C     FIND SR1
C
      CALL MATMUL(ATAP,DDR1,D2R1,3,3,1)
      SR1(3,1)=0.0D0
      SR1(1,1)=D2R1(1,1)/D
      SR1(2,1)=D2R1(2,1)/D
C
C     FIND RO
C
      CALL MATMUL(ATAPT,SR1,SR11,3,3,1)
      DO 35 I=1,3
   35 RO(I,1)=R2(I,1)-SR11(I,1)
C
C     FIND V
C
      CALL MATMUL(AT1,SR1,SR11,3,3,1)
      DO 34 I=1,3
   34 V(I,1)=(-R1(I,1)+RO(I,1)+SR11(I,1))/DT
C
C     FIND SR2 AND SR3
C
      DO 36 I=1,3
      SR12(I,1)=R2(I,2)-RO(I,1)
   36 SR13(I,1)=R2(I,3)-RO(I,1)
      CALL MATMUL(ATAP,SR12,SR2,3,3,1)
      CALL MATMUL(ATAP,SR13,SR3,3,3,1)
  150 FORMAT(3(2X,6D20.8,/))
      PRINT 114
  114 FORMAT(12X,'W',17X,'V',17X,'RO',19X,'SR1',17X,'SR2',17X
     .,'SR3',/)
      PRINT 150,(W(I,1),V(I,1),RO(I,1),SR1(I,1),SR2(I,1),SR3(
     .I,1),I=1,3)
      DO 121 I=1,3
      DV(I,1)=999.0
      DRO(I,1)=999.0
```

```
      DO 121 J=1,3
      DSR(I,J)=999.0
  121 ORDOT(I,J)=999.0
      DO 115 I=1,3
      IF(DABS(VS(I,1)).LT.1.0D-12)GO TO 117
      DV(I,1)=(V(I,1)-VS(I,1))/VS(I,1)
  117 IF(DABS(ROS(I,1)).LT.1.0D-12)GO TO 118
      DRO(I,1)=(RO(I,1)-ROS(I,1))/ROS(I,1)
  118 IF(DABS(SRRS(I,1)).LT.1.0D-12)GO TO 119
      DSR(I,1)=(SR1(I,1)-SRRS(I,1))/SRRS(I,1)
  119 IF(DABS(SRRS(I,2)).LT.1.0D-12)GO TO 122
      DSR(I,2)=(SR2(I,1)-SRRS(I,2))/ SRRS(I,2)
  122 IF(DABS(SRRS(I,3)).LT.1.0D-12)GO TO 123
      DSR(I,3)=(SR3(I,1)-SRRS(I,3))/SRRS(I,3)
  123 CONTINUE
      DO 115 J=1,3
      IF(DABS(RDOTS(I,J)).LT.1.0D-12)GO TO 115
      ORDOT(I,J)=(RDOT(I,J)-RDOTS(I,J))/RDOTS(I,J)
  115 CONTINUE
      PRINT 50
      PRINT 110,(DRO(I,1),(DSR(I,J),J=1,3),I=1,3)
      PRINT 51
      PRINT 110,(DV(I,1),(DRDOT(I,J),J=1,3),I=1,3)
  110 FORMAT(5X,4D20.8,/)
   50 FORMAT(//,14X,'DRO',16X,'DSR 1',15X,'DSR 2',15X,'DSR 3'
     .,/)
   51 FORMAT(//,14X,'DV',16X,'DRDOT 1',13X,'DRDOT 2',13X,
     .'DRDOT 3',/)
      RETURN
      END
```

```
      SUBROUTINE STATE(RR,SRR,SRRI,RDOT,WI,DPSI,TC,T2,NC,IVIS
     .,RHO,RHOD,DTHETA,DPHI,NN,V,RO,SSR)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION RDOT(3,3),RR(3,3),SRR(3,3),SRRI(3,3),WI(3)
      DIMENSION V(3),CRO(3),APHI(3,3),ATHETA(3,3),APSI(3,3),
     1APSIH(3,3),AO(3,3),A5(3,3),A6(3,3),CR(3),SR(3,1)
      DIMENSION A3T(3,3),R(3,1),RA(3,1),SRP(3,1),SRPI(3,1)
      DIMENSION IV(3),A7(3,3),ATHTAH(3,3),SRI(3,1),WB(3)
      DIMENSION RHO(3),RHOD(3)
      DIMENSION D(3,3),E(3)
      DIMENSION WBT(3,3),WXSRA(3,3),RDD(3,3),RD2(3,3)
      DIMENSION RO(3,1),SSR(3,3)
      DIMENSION SSRA(3,3),A6T(3,3)
      DIMENSION AB(3,1),BC(3,3)
      DIMENSION XI(3),WD(3),WDT(3,3),WDXSRR(3,3),SRRA(3,1)
      EQUIVALENCE(THETAH,THTAH)
      IV(1)=5
      IV(2)=3
      IV(3)=1
      K3=0
      I1=0
      IF(NC.NE.1) GO TO 6
      CALL DATA(WXO,WYO,WZO,PSIO,THTAH,PSIH,V,CRO,SRR,SA,SB,S
     .C,NN)
      WYO=.00005
      THETAH=30.0
      PSIH=30.0
      WRITE(6,999)SA,SB,SC
  999 FORMAT(7X,'SA=',F16.8,7X,'SB=',F16.8,7X,'SC=',F16.8,//)
      SAS=SA*SA
      SBS=SB*SB
      SCS=SC*SC
      A=(SBS+SCS)
      B=(SAS+SCS)
      C=(SAS+SBS)
      SIGX = A/C
      SIGY = B/C
      DO 920 I=1,3
```

```
920 SRR(3,I)=SC*DSQRT(1.000-SRR(1,I)**2/SAS-SRR(2,I)**2/SBS
   .)
    PRINT 47,WXO,WYO,WZO,PSIO,THTAH,PSIH
 47 FORMAT(7X,'WXO=',E16.8,6X,'WYO=',E16.8,6X,'WZO=',E16.8,
   .//,7X,'PSIO=',E16.8,5X,'THTAH=',E16.8,4X,'PSIH=',E16.8,
   .//)
    PRINT 48,V(1),V(2),V(3),CRO(1),CRO(2),CRO(3)
 48 FORMAT(7X,'XD=',E16.8,7X,'YD=',E16.8,7X,'ZD=',E16.8,//,
   .7X,'X=',E16.8,8X,'Y=',E16.8,8X,'Z=',E16.8,//)
    DO 910 I=1,3
910 PRINT 49,I,SRR(1,I),SRR(2,I),SRR(3,I)
 49 FORMAT(7X,'POINT ',I1,4X,'SX=',E16.8,7X,'SY=',E16.8,7X,
   .'SZ=',E16.8,//)
    PI=3.14159265
    PSIO=PSIO*PI/180.0
    THTAH=THTAH*PI/180.0
    PSIH=PSIH*PI/180.0
    W1P = WXO/WZO
    W2P = WYO/WZO
    W3P = 1.0000000000
    PSIP = PSIO
    SIGZ = 1.00000000
    TP=.5*(SIGX*W1P**2+SIGY*W2P**2+W3P**2)
    HP=(SIGX**2*W1P**2+SIGY**2*W2P**2+W3P**2)**.5
    SK=(((2.*TP-HP**2)*(SIGY-SIGX))/((HP**2-2.*SIGX*TP)*(1.
   .-SIGY)))**.5
    CALL DCEL1(RES,SK,IER)
    XLAMP=((HP**2-2.*SIGX*TP)*(1.-SIGY)/(SIGX*SIGY))**.5
    A1=(SIGY*(1.-SIGY)/(SIGX*(1.-SIGX)))**.5
    A2=(SIGX*(1.-SIGX)/(SIGY*(1.-SIGY)))**.5
    A3=((SIGX*SIGY)**.5*(SIGY-SIGX))/((1.-SIGX)*(1.-SIGY))*
   .*.5
    ROW=((HP**2-2.*SIGX*TP)/(1.-SIGX))**.5
    DELTA=(SIGY*(SIGY-SIGX)/(1.-SIGX))*(W2P**2/ROW**2)
    WRITE(6,3)A1,A2,A3,ROW,DELTA
  3 FORMAT(1X,5(2X,D12.5))
    TAUP=0.0
    T=0.0
    DTAU=0.0001*RES/XLAMP
    WRITE(6,545)TP,HP,RES,SIGX,SIGY,SIGZ
    WRITE(6,546)SK,XLAMP
545 FORMAT(6(3X,D12.5))
546 FORMAT(2(3X,D12.5),/)
    DO  5 I = 1,3
    DO  5  J = 1,3
    ATHETA(I,J) = 0.0000000
    APSI(I,J) = 0.00000
    ATHTAH(I,J) = 0.0
    APSIH(I,J) = 0.00000
  5 APH(I,J) =  0.00000
    ATHTAH(1,1) = DCOS(THTAH)
    ATHTAH(1,3) = -DSIN(THTAH)
    ATHTAH(2,2) = 1.0000000
    ATHTAH(3,3) = ATHTAH(1,1)
    ATHTAH(3,1) =-ATHTAH(1,3)
    APSIH(1,1) = DCOS(PSIH)
    APSIH(1,2) = DSIN(PSIH)
    APSIH (2,2) = APSIH(1,1)
    APSIH(2,1) =-APSIH(1,2)
    APSIH(3,3) = 1.00000000
    CALL MATMUL(ATHTAH,APSIH,A0,3,3,3)
  6 IF(T.GT.TC)GO TO 901
    IVIS=1
    I2=0
    T =  TAUP/WZO
    SCK=1.-SK**2
    IF(SK.LT.1.0D-05)K3=2
    IF(K3.EQ.2)TAUP=TC*WZO
    IF(K3.EQ.2)T=TC
    ARG=XLAMP*TAUP
    CALL DJELF(SN,CN,DN,ARG,SCK)
    WX=(W1P*CN-A1*(W3P/ROW)*W2P*SN*DN)/(1.-DELTA*SN**2)
    WY=(W2P*CN*DN+A2*(W3P/ROW)*W1P*SN)/(1.-DELTA*SN**2)
    WZ=(W3P*DN-A3*W1P*(W2P/ROW)*SN*CN)/(1.-DELTA*SN**2)
    CT=DSQRT(HP*HP-SIGX*SIGX*WX*WX)/HP
```

```
      ST=-SIGX*WX/HP
      SP=SIGY*WY/(HP*CT)
      CP=SIGZ*WZ/(HP*CT)
      SPST=SP*ST
      CPST=CP*ST
      PSIP=PSIP+HP*(SIGY-1.)*SP*SP*DTAU
      PSI=PSIP+HP*TAUP
      THETA = DATAN2(ST,CT)
      PHI = DATAN2(SP,CP)
      CCT=CT*CP
      SCT=DSQRT(1.0D0-CCT*CCT)
      CTHETA=DATAN2(SCT,CCT)
      DCTHET=CTHETA*180.0D0/3.141592
      DPHI=PHI*180.00000/3.14159
      DTHETA = THETA*180.0000000/3.14159
      DPSI = PSI*180.000000/3.14159
      DO 222 I = 1,3
      DO 222 J = 1,3
      APHI(I,J) = 0.0D0
      ATHETA(I,J) = 0.0D0
      IF(K3.LT.2)GO TO 222
      CT=1.0D0
      ST=0.0D0
      CP=1.0D0
      SP=0.0D0
  222 APSI(I,J) = 0.0D0
      APHI(1,1) = 1.0000000
      APHI(2,2) = CP
      APHI(2,3) = SP
      APHI(3,2) = -SP
      APHI(3,3) = CP
      ATHETA(1,1) = CT
      ATHETA(1,3) = -ST
      ATHETA(2,2) = 1.0000000
      ATHETA(3,3) =  CT
      ATHETA(3,1) =  ST
      APSI(1,1) = DCOS(PSI)
      APSI(1,2) = DSIN(PSI)
      APSI(2,1) = - APSI(1,2)
      APSI(2,2) =   APSI(1,1)
      APSI(3,3) = 1.0D0
      CALL MATMUL(ATHETA,APSI,A5,3,3,3)
      CALL MATMUL(APHI,A5,A6,3,3,3)
      CALL MATMUL (A6,A0,A7,3,3,3)
      DO 33 I = 1,3
      DO  33  J = 1,3
   33 A3T(I,J) = A7(J,I)
      DO 949 KK=1,3
      DO 950 I=1,3
  950 SR(I,1)=SRR(I,KK)
      CALL MATMUL(A3T,SR,SRI,3,3,1)
      DO  10  I = 1,3
      CR(I) = V(I)*T + CRO(I)
   10 R(I,1)=CR(I)+SRI(I,1)
      CALL MATMUL (A7,R,RA,3,3,1)
      RHO1=DSQRT(R(1,1)*R(1,1)+R(2,1)*R(2,1)+R(3,1)*R(3,1))
      RHO(KK)=RHO1
      XLA=RA(1,1)/RHO1
      XMA=RA(2,1)/RHO1
      XNA=RA(3,1)/RHO1
      A4=XLA*XLA/SAS+XMA*XMA/SBS+XNA*XNA/SCS
      B4=(A4-XLA*XLA/SAS)*SR(1,1)-XMA*XLA/SBS*SR(2,1)-XNA*XLA
     ./SCS*SR(3,1)
      SRP(1,1)=2.00000*B4/A4-SR(1,1)
      SRP(2,1)=(XMA/XLA)*(SRP(1,1)-SR(1,1))+SR(2,1)
      SRP(3,1) = (XNA/XLA)*(SRP(1,1)- SR(1,1))+SR(3,1)
      CALL MATMUL (A3T,SRP,SRPI,3,3,1)
      RHO2=DSQRT((CR(1)+SRPI(1,1))**2+(CR(2)+SRPI(2,1))**2+(C
     .R(3)+SRPI(3,1))**2)
      IF(RHO2.LT.RHO1) IVIS=IVIS+IV(KK)
   50 CONTINUE
  948 CONTINUE
      DO 949 IJ=1,3
      RR(IJ,KK)=R(IJ,1)
```

```
949 SRRI(IJ,KK)=SRI(IJ,1)
    WX = WX*WZO
    WY = WY*WZO
    WZ = WZ*WZO
    WB(1)=WX
    WB(2)=WY
    WB(3)=WZ
    XI(1)=(SCS-SBS)/A
    XI(2)=(SAS-SCS)/B
    XI(3)=(SBS-SAS)/C
    WD(1)=WB(2)*WB(3)*XI(1)
    WD(2)=WB(3)*WB(1)*XI(2)
    WD(3)=WB(1)*WB(2)*XI(3)
    DO 908 J=1,3
    DO 908 I=1,3
908 RDOT(I,J)=V(I)+A3T(I,1)*(-WB(3)*SRR(2,J)+WB(2)*SRR(3,J)
   .)+A3T(I,2)*(WB(3)*SRR(1,J)-WB(1)*SRR(3,J))+A3T(I,3)*(-W
   .B(2)*SRR(1,J)+WB(1)*SRR(2,J))
    DO 899 J=1,3
899 RHOD(J)=(RR(1,J)*RDOT(1,J)+RR(2,J)*RDOT(2,J)+RR(3,J)*RD
   .OT(3,J))/RHO(J)
    DO 909 I=1,3
909 WI(I)=WX*A3T(I,1)+WY*A3T(I,2)+WZ*A3T(I,3)
930 CONTINUE
109 TAUP = TAUP + DTAU
    T=TAUP/WZO
    I1=1
    T2=T
    IF(K3.EQ.2)GO TO 901
110 CONTINUE
    GO TO 6
901 CALL TILDE(WB,WBT)
    IF(T.GT.TC)T=(TAUP-DTAU)/WZO
    CALL MATMUL(WBT,SRR,WXSRA,3,3,3)
    CALL MATMUL(WBT,WXSRA,RDD,3,3,3)
    CALL DOTPRO(WB,WB,WW)
    IF(K3.EQ.2)GO TO 699
    DO 601 I=1,3
    SSRA(I,1)=SRR(I,1)
601 RO(I,1)=CR(I)+A3T(I,3)*SRR(3,1)
    CALL TILDE(WD,WDT)
    CALL MATMUL(WDT,SRR,WDXSRR,3,3,3)
    DO 606 I=1,3
    DO 606 J=1,3
606 RDD(I,J)=RDD(I,J)+WDXSRR(I,J)
    GO TO 711
699 CONTINUE
    DO 700 I=1,2
700 SSRA(I,1)=-RDD(I,1)/WW
    SSRA(3,1)=0.0D0
    RDD(3,2)=RDD(3,2)-RDD(3,1)
    RDD(3,3)=RDD(3,3)-RDD(3,1)
    RDD(3,1)=0.0D0
    CALL MATMUL(A3T,RDD,RD2,3,3,3)
    DO 703 I=1,3
703 RO(I,1)=RR(I,1)+RD2(I,1)/WW
711 CONTINUE
    DO 702 J=1,3
    DO 702 I=1,3
702 A6T(I,J)=A6(J,I)
    DO 705 I=1,3
    DO 705 J=2,3
705 SSRA(I,J)=SRR(I,J)
    SSRA(3,2)=SRR(3,2)-SRR(3,1)
    SSRA(3,3)=SRR(3,3)-SRR(3,1)
    CALL MATMUL(A6T,SSRA,SSR,3,3,3)
    PRINT 112,DPHI,DTHETA,DCTHET
112 FORMAT(5X,'PHI=',D16.8,5X,'THETA=',D16.8,5X,'DCAPTHETA=
   .',D16.8,/)
911 CONTINUE
    RETURN
    END
```

```
      SUBROUTINE DATA(W1,W2,W3,PSIO1,THTAH1,PSIH1,V1,CR1,SRR1
     .,S1,S2,S3,NN)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION CR1(3),V1(3),SRR1(3,3)
      DIMENSION VP(3),CROP(3),SRRP(3,3)
      IF(NN.GT.1)GO TO 10
      READ(5,100)WXOP,WYOP,WZOP,PSIOP,THTAHP,PSIHP,VP(1),VP(2
     .),VP(3),CROP(1),CROP(2),CROP(3),SRRP(1,1),SRRP(2,1),SRRP
     .(1,2),SRRP(2,2),SRRP(1,3),SRRP(2,3)
  100 FORMAT(5F16.0)
      READ 101,SAP,SBP,SCP
  101 FORMAT(3F16.0)
      SRRP(3,1)=0.0D0
      SRRP(3,2)=0.0D0
      SRRP(3,3)=0.0D0
      NN=NN+1
   10 CONTINUE
      S1=SAP
      S2=SBP
      S3=SCP
      PSIH1=PSIHP
      PSIO1=PSIOP
      THTAH1=THTAHP
       W1=WXOP
      W2=WYOP
      W3=WZOP
      DO 20 J=1,3
      CR1(J)=CROP(J)
      V1(J)=VP(J)
      DO 20 I=1,3
      SRR1(I,J)=SRRP(I,J)
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE VISIB(IVIS)
      IF(IVIS.NE.1) GO TO 19
      PRINT 3
    3 FORMAT(/,6X,'ALL POINTS VISIBLE',/)
   19 IF(IVIS.LT.6) GO TO 17
      PRINT 6
    6 FORMAT(/,6X,'POINT NUMBER 1 NOT VISIBLE',/)
      IVIS=IVIS-5
   17 IF(IVIS.LT.4) GO TO 7
      PRINT 8
    8 FORMAT(/,6X,'POINT NUMBER 2 NOT VISIBLE',/)
      IVIS=IVIS-3
    7 IF(IVIS.LT.2) GO TO 9
      PRINT 18
   18 FORMAT(/,6X,'POINT NUMBER 3 NOT VISIBLE',/)
    9 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DOTPRO(D,E,DOT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION D(3,1),E(3,1)
      DOT=D(1,1)*E(1,1)+D(2,1)*E(2,1)+D(3,1)*E(3,1)
      RETURN
      END
```

```
      SUBROUTINE MATMUL (A,B,C,L,M,N)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(L,M),B(M,N),C(L,N)
      DO 15 I=1,L
      DO 15 K=1,N
      TEMP=0.0
      DO 14 J=1,M
   14 TEMP=TEMP+A(I,J)*B(J,K)
   15 C(I,K)=TEMP
      RETURN
      END
```

```
      SUBROUTINE TILDE(E,D)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION D(3,3),E(3)
      DO 5 I=1,3
    5 D(I,I)=0.0
      D(1,2)=-E(3)
      D(2,1)= E(3)
      D(1,3)= E(2)
      D(3,1)=-E(2)
      D(2,3)=-E(1)
      D(3,2)= E(1)
      RETURN
      END
```

Listings of subroutines DJELF and DCEL 1 appear in the listing of program, DUMRRR.

```
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION BA(11,11),B(11,11),H(2,11),R1(3,1),DZ(2,1),RO
     .(3,1),VO(3,1),CR(3,1),DX(11,1),WEIGHT(11,11),PRO(11,1),
     .A(3,3),C(11,1),AT(3,3)
      NN=1
      NM=0
      READ(5,1)TC,DT
    1 FORMAT(2F16.0)
      CHECK=1.0D0
      T=TC
      NC=1
      ROEST=1.0D05
      WST=1.0D-04
      READ(5,13)PSIH,THETAH,WE,RO(1,1),RO(2,1)
      READ(5,13)RO(3,1),VO(1,1),VO(2,1),VO(3,1),R1(1,1)
      READ(5,13)R1(2,1)
   13 FORMAT(5F16.0)
      R1(3,1)=0.0D0
      PRINT 937,PSIH,THETAH,WE,RO(1,1),RO(2,1),RO(3,1),VO(1,1
     .),VO(2,1),VO(3,1),R1(1,1),R1(2,1),R1(3,1)
  937 FORMAT(5X,'PSIH=',D16.8,5X,'THETAH=',D16.8,5X,'WE=',D16
     ..8,/,5X,'XO=',D16.8,5X,'YO=',D16.8,5X,'ZO=',D16.8,/,5X,
     .'XOD=',D16.8,5X,'YOD=',D16.8,5X,'ZOD=',D16.8,/,5X,'X1='
     .,D16.8,5X,'Y1=',D16.8,5X,'Z1=',D16.8,//)
    2 CONTINUE
      NM=NM+1
      INM=NM/10
      PSIH=PSIH*3.141592/180.0
      THETAH=THETAH*3.141592/180.0
      DO 30 K=1,6
      PSI=WE*T
      PRINT 947,T
  947 FORMAT(5X,'T=',D16.8,/)
  948 CONTINUE
      CALL RHOES(DZ,T,NC,PSIH,THETAH,WE,RO,VO,R1,CR,A,PSI,ROE
     .,RORODE,NN,NM,INM)
      DZ(1,1)=DZ(1,1)/ROEST
      DZ(2,1)=DZ(2,1)/ROEST/ROEST/WST
      CALL TRPOSE(A,AT,3,3)
      TND=T*WST
      DO 38 I=1,3
      RO(I,1)=RO(I,1)/ROEST
      VO(I,1)=VO(I,1)/ROEST/WST
   38 R1(I,1)=R1(I,1)/ROEST
      ROE=ROE/ROEST
      WE=WE/WST
      CALL DERIV(H,RO,VO,R1,WE,PSIH,THETAH,ROE,PSI,TND,AT,A)
      IF(K.EQ.6)GO TO 33
      DO 31 I=1,2
      DO 31 J=1,11
      B(2*K-2+I,J)=H(I,J)
   31 C(2*K-2+I,1)=DZ(I,1)
      T=T+DT
      DO 39 I=1,3
      RO(I,1)=RO(I,1)*ROEST
      VO(I,1)=VO(I,1)*ROEST*WST
   39 R1(I,1)=R1(I,1)*ROEST
      WE=WE*WST
   30 CONTINUE
   33 CONTINUE
      DO 32 J=1,11
   32 B(11,J)=H(1,J)
      C(11,1)=DZ(1,1)
      PRINT 919
  919 FORMAT(10X,'C MATRIX',/)
      PRINT 51,(C(I,1),I=1,11)
```

```
 51 FORMAT(11(5X,D20.8,/)/)
949 CONTINUE
    DO 4 I=1,11
    DO 4 J=1,11
  4 BA(I,J)=B(I,J)
    CALL A37IMS(11,BA)
    CALL MATMUL(BA,C,DX,11,11,1)
    CF=0.02
    IF(CHECK.LT.1.0D-4)CF=0.1
    IF(CHECK.LT.1.0D-6)CF=1.0
    DO 5 I=1,3
    RO(I,1)=RO(I,1)+DX(I+3,1)*CF
  5 VO(I,1)=VO(I,1)+DX(I+6,1)*CF
    R1(1,1)=R1(1,1)+DX(10,1)*CF
    R1(2,1)=R1(2,1)+DX(11,1)*CF
    PSIH=PSIH+DX(1,1)*CF
    THETAH=THETAH+DX(2,1)*CF
    WE=WE+DX(3,1)*CF
    WE=WE*WST
    DO 40 I=1,3
    RO(I,1)=RO(I,1)*ROEST
 40 VO(I,1)=VO(I,1)*ROEST*WST
    R1(1,1)=R1(1,1)*ROEST
    R1(2,1)=R1(2,1)*ROEST
    CHECK=0.0D0
    DO 35 I=1,11
 35 CHECK=C(I,1)*C(I,1)+CHECK
    CHECK=DSQRT(CHECK)
    PSIH=PSIH*180.0/3.141592
    THETAH=THETAH*180.0/3.141592
    PRINT 8,PSIH,THETAH,WE,CHECK
  8 FORMAT(5X,'PSIH=',D20.8,4X,'THETAH=',D20.8,4X,'WE=',D2
   .0.8,4X,'CHECK=',D20.8,//)
    PRINT 9,(RO(I,1),VO(I,1),I=1,3)
  9 FORMAT(15X,'RO',23X,'VO',/,3(5X,D20.8,5X,D20.8,/),//)
    PRINT 10,(R1(I,1),I=1,2)
 10 FORMAT(15X,'R1',/,2(5X,D20.8,/),//)
    PRINT 11,(DX(I,1),I=1,11)
 11 FORMAT(20X,'DX MATRIX',/,4(5X,3(D20.8,3X),//),//)
950 CONTINUE
    IF(CHECK.LT.1.0D-07)GO TO 111
    T=TC
    NC=2
    GO TO 2
111 STOP
    END
```

```
    SUBROUTINE DERIV(H,RO,VO,R1,WE,PSIH,THETAH,ROE,PSI,T,AT
   .,A)
    IMPLICIT REAL*8(A-H,O-Z)
    DIMENSION A(3,3),AT(3,3),H(2,11),RO(3,1),VO(3,1)
    DIMENSION R1(3,1),D1(3,1),D2(3,1),D3(3,1),D4(1,3)
    DIMENSION D5(1,3),D6(1,3),C(3,1),R1T(1,3),ROT(1,3)
    DIMENSION R(3,1),G31(3,1),VOT(1,3)
    DIMENSION D10(3,3),WET(3,3),G11(1,1)
    AA=R1(1,1)*DCOS(PSI)-R1(2,1)*DSIN(PSI)
    BB=R1(1,1)*DSIN(PSI)+R1(2,1)*DCOS(PSI)
    DO 6 I=1,3
    R(I,1)=RO(I,1)+VO(I,1)*T
    D1(I,1)=0.0
    D2(I,1)=0.0
  6 D3(I,1)=0.0
    D1(1,1)=1.0
    D2(2,1)=1.0
    D3(3,1)=1.0
    CALL TRPOSE(R1,R1T,3,1)
    CALL TRPOSE(R,ROT,3,1)
    CALL TRPOSE(VO,VOT,3,1)
    C(1,1)=-AA*DCOS(THETAH)*DSIN(PSIH)-BB*DCOS(PSIH)
```

```
C(2,1)=AA*DCOS(THETAH)*DCOS(PSIH)-BB*DSIN(PSIH)
C(3,1)=0.0
CALL MATMUL(ROT,C,G11,1,3,1)
H(1,1)=G11(1,1)/ROE
C(1,1)=-AA*DCOS(PSIH)*DSIN(THETAH)
C(2,1)=-AA*DSIN(PSIH)*DSIN(THETAH)
C(3,1)=-AA*DCOS(THETAH)
CALL MATMUL(ROT,C,G11,1,3,1)
H(1,2)=G11(1,1)/ROE
C(1,1)=-BB*DCOS(PSIH)*DCOS(THETAH)-AA*DSIN(PSIH)
C(2,1)=-BB*DSIN(PSIH)*DCOS(THETAH)+AA*DCOS(PSIH)
C(3,1)=-BB*DSIN(THETAH)
CALL MATMUL(ROT,C,G11,1,3,1)
H(1,3)=G11(1,1)*T/ROE
CALL MATMUL(A,D1,G31,3,3,1)
CALL MATMUL(R1T,G31,G11,1,3,1)
H(1,4)=(R(1,1)+G11(1,1))/ROE
CALL MATMUL(A,D2,G31,3,3,1)
CALL MATMUL(R1T,G31,G11,1,3,1)
H(1,5)=(R(2,1)+G11(1,1))/ROE
CALL MATMUL(A,D3,G31,3,3,1)
CALL MATMUL(R1T,G31,G11,1,3,1)
H(1,6)=(R(3,1)+G11(1,1))/ROE
H(1,7)=T*H(1,4)
H(1,8)=T*H(1,5)
H(1,9)=T*H(1,6)
CALL MATMUL(A,D1,G31,3,3,1)
CALL MATMUL(ROT,G31,G11,1,3,1)
H(1,10)=(G11(1,1)+R1(1,1))/ROE
CALL MATMUL(A,D2,G31,3,3,1)
CALL MATMUL(ROT,G31,G11,1,3,1)
H(1,11)=(G11(1,1)+R1(2,1))/ROE
C(1,1)=-AA*DCOS(THETAH)*DSIN(PSIH)-BB*DCOS(PSIH)
C(2,1)=AA*DCOS(THETAH)*DCOS(PSIH)-BB*DSIN(PSIH)
C(3,1)=0.0
CALL MATMUL(VOT,C,G11,1,3,1)
H(2,1)=G11(1,1)
C(1,1)=BB*DSIN(PSIH)*DCOS(THETAH)-AA*DCOS(PSIH)
C(2,1)=-BB*DCOS(PSIH)*DCOS(THETAH)-AA*DSIN(PSIH)
C(3,1)=0.0
CALL MATMUL(ROT,C,G11,1,3,1)
H(2,1)=H(2,1)+WE*G11(1,1)
C(1,1)=-AA*DCOS(PSIH)*DSIN(THETAH)
C(2,1)=-AA*DSIN(PSIH)*DSIN(THETAH)
C(3,1)=-AA*DCOS(THETAH)
CALL MATMUL(VOT,C,G11,1,3,1)
H(2,2)=G11(1,1)
CD=DCOS(PSI)*WE*R1(2,1)+DSIN(PSI)*WE*R1(1,1)
C(1,1)=DCOS(PSIH)*DSIN(THETAH)*CD
C(2,1)=DSIN(PSIH)*DSIN(THETAH)*CD
C(3,1)=DCOS(THETAH)*CD
CALL MATMUL(ROT,C,G11,1,3,1)
H(2,2)=H(2,2)+G11(1,1)
C(1,1)=-R1(2,1)
C(2,1)=R1(1,1)
C(3,1)=0.0
CALL MATMUL(AT,C,G31,3,3,1)
CALL MATMUL(ROT,G31,G11,1,3,1)
H(2,3)=G11(1,1)
C(1,1)=-BB*DCOS(PSIH)*DCOS(THETAH)-AA*DSIN(PSIH)
C(2,1)=-BB*DSIN(PSIH)*DCOS(THETAH)+AA*DCOS(PSIH)
C(3,1)=-BB*DSIN(THETAH)
CALL MATMUL(VOT,C,G11,1,3,1)
H(2,3)=H(2,3)+T*G11(1,1)
C(1,1)=-AA*DCOS(PSIH)*DCOS(THETAH)+BB*DSIN(PSIH)
C(2,1)=-AA*DSIN(PSIH)*DCOS(THETAH)-BB*DCOS(PSIH)
C(3,1)=-AA*DSIN(THETAH)
CALL MATMUL(ROT,C,G11,1,3,1)
H(2,3)=H(2,3)+G11(1,1)*T*WE
CALL TRPOSE(D1,D4,3,1)
CALL TRPOSE(D2,D5,3,1)
CALL TRPOSE(D3,D6,3,1)
DO 12 I=1,3
DO 12 J=1,3
```

```
12 WET(I,J)=0.0
   WET(1,2)=-WE
   WET(2,1)=WE
   CALL MATMUL(WET,R1,G31,3,3,1)
   CALL MATMUL(AT,G31,G31,3,3,1)
   CALL MATMUL(D4,G31,G11,1,3,1)
   H(2,4)=VO(1,1)+G11(1,1)
   CALL MATMUL(D5,G31,G11,1,3,1)
   H(2,5)=VO(2,1)+G11(1,1)
   CALL MATMUL(D6,G31,G11,1,3,1)
   H(2,6)=VO(3,1)+G11(1,1)
   CALL MATMUL(AT,R1,G31,3,3,1)
   CALL MATMUL(D4,G31,G11,1,3,1)
   H(2,7)=ROT(1,1)+G11(1,1)
   CALL MATMUL(D5,G31,G11,1,3,1)
   H(2,8)=ROT(1,2)+G11(1,1)
   CALL MATMUL(D6,G31,G11,1,3,1)
   H(2,9)=ROT(1,3)+G11(1,1)
   CALL MATMUL(WET,D1,G31,3,3,1)
   CALL MATMUL(AT,G31,G31,3,3,1)
   CALL MATMUL(ROT,G31,G11,1,3,1)
   H(2,10)=G11(1,1)
   CALL MATMUL(AT,D1,G31,3,3,1)
   CALL MATMUL(VOT,G31,G11,1,3,1)
   H(2,10)=H(2,10)+G11(1,1)
   CALL MATMUL(WET,D2,G31,3,3,1)
   CALL MATMUL(AT,G31,G31,3,3,1)
   CALL MATMUL(ROT,G31,G11,1,3,1)
   H(2,11)=G11(1,1)
   CALL MATMUL(AT,D2,G31,3,3,1)
   CALL MATMUL(VOT,G31,G11,1,3,1)
   H(2,11)=H(2,11)+G11(1,1)
   RETURN
   END
```

```
   SUBROUTINE RHOES(DZ,T,NC,PSIH,THETAH,WE,RO,VO,R1,CR,A,P
  .SI,ROE,RORODE,NN,NM,INM)
   IMPLICIT REAL*8(A-H,O-Z)
   DIMENSION RR(3,3),SRR(3,3),SRRI(3,3),RDOT(3,3),WI(3),
  .RHO(3),RHOD(3),RO(3,1),R1(3,1),CR(3,1),VO(3,1),
  .WET(3,3),A(3,3),AT(3,3),G31(3,1)
   DIMENSION DZ(2,1),G32(3,1),V11(3,1)
   DIMENSION R(3,1),V(3,1)
   CALL STATE(RR,SRR,SRRI,RDOT,WI,DPSI,T,T2,NC,IVIS,RHO,RH
  .OD,DTHETA,DPHI,DDD,FFF,NN,NM,INM)
   NC=2
   DO 12 I=1,3
   G31(I,1)=0.0
12 CR(I,1)=RO(I,1)+VO(I,1)*T
   A(1,1)=DCOS(PSI)*DCOS(THETAH)*DCOS(PSIH)-DSIN(PSI)*DSIN
  .(PSIH)
   A(1,2)=DCOS(PSI)*DCOS(THETAH)*DSIN(PSIH)+DSIN(PSI)*DCOS
  .(PSIH)
   A(1,3)=-DCOS(PSI)*DSIN(THETAH)
   A(2,1)=-DSIN(PSI)*DCOS(THETAH)*DCOS(PSIH)-DCOS(PSI)*DSIN
  .(PSIH)
   A(2,2)=-DSIN(PSI)*DCOS(THETAH)*DSIN(PSIH)+DCOS(PSI)*DCOS
  .(PSIH)
   A(2,3)=DSIN(PSI)*DSIN(THETAH)
   A(3,1)=DSIN(THETAH)*DCOS(PSIH)
   A(3,2)=DSIN(THETAH)*DSIN(PSIH)
   A(3,3)=DCOS(THETAH)
   CALL TRPOSE(A,AT,3,3)
   CALL MATMUL(AT,R1,G31,3,3,1)
   DO 17 I=1,3
17 R(I,1)=CR(I,1)+G31(I,1)
   CALL DOTPRO(R,R,ROES)
   ROE=DSQRT(ROES)
   DO 15 I=1,3
```

```
      DO 15 J=1,3
      G32(I,1)=0.0D0
      V11(I,1)=0.0D0
   15 WET(I,J)=0.0D0
      WET(1,2)=-WE
      WET(2,1)=WE
      CALL MATMUL(WET,R1,G32,3,3,1)
      CALL MATMUL(AT,G32,V11,3,3,1)
      DO 18 I=1,3
   18 V(I,1)=V0(I,1)+V11(I,1)
      CALL DOTPRO(R,V,RORODE)
      DZ(1,1)=RHO(1)-ROE
      DZ(2,1)=RHOD(1)*RHO(1)-RORODE
      RHODE=RORODE/ROE
      RETURN
      END
```

```
      SUBROUTINE DCEL1(RES,AK,IER)
C     SUBROUTINE DCEL1
C
C     PURPOSE
C        CALCULATE COMPLETE ELLIPTIC INTEGRALS
C     OF THE FIRST KIND
C
C     DESCRIPTION OF PARAMETERS
C     RES   -RESULT VALUE IN DOUBLE PRECISION
C       AK    -MODULLUS (INPUT) IN DOUBLE PRECISION
C       IER   -RESULTANT ERROR CODE WHERE
C              IER=0 NO ERROR
C              IER=1 AK NOT IN RANGE -1 TO +1
      DOUBLE PRECISION RES,AK,GEO,ARI,AARI
      IER=0
      ARI=2.D0
      GEO=(0.5D0-AK)+0.5D0
      GEO=GEO+GEO*AK
      RES=0.5D0
      IF(GEO)1,2,4
    1 IER=1
    2 RES=1.075
      RETURN
    3 GEO=GEO*AARI
    4 GEO=DSQRT(GEO)
      GEO=GEO+GEO
      AARI=ARI
      ARI=ARI+GEO
      RES=RES+RES
      IF(GEO/AARI-0.99999995D0)3,5,5
    5 RES=RES/ARI*6.28318530717958650
      RETURN
      END
```

```
      SUBROUTINE DJELF(SN,CN,DN,X,SCK)
      DIMENSION ARI(12),GEO(12)
C     SUBROUTINE DJELF
C
C     PURPOSE
C        COMPUTES THE THREE JACOBIAN ELLIPTIC FUNCTIONS
C     SN,CN,DN.
C
C     DESCRIPTION OF PARAMETERS
C        SN    - RESULT VALUE SN(X) IN DOUBLE PRECISION
C        CN    - RESULT VALUE CN(X) IN DOUBLE PRECISION
C        DN    - RESULT VALUE DN(X) IN DOUBLE PRECISION
C        X     - DOUBLE PRECISION ARGUMENT OF JACOBIAN
```

```
C      ELLIPTIC FUNCTIONS
C         SCK  - SQUARE OF COMPLEMENTARY MODULUS IN DOUBLE
C      PRECISION
C
C         EVALUATION
C         CALCULATION IS DONE USING THE PROCESS OF
C      THE ARITHMETIC GEOMETRIC MEAN TOGETHER WITH GAUSS
C      DESCENDING TRANSFORMATION BEFORE INVERSION OF THE
C      INTEGRAL TAKES PLACE.
C
C
      DOUBLE PRECISION SN,CN,DN,X,SCK,ARI,GEO,CM,Y,A,B,C,D
C
C         TEST MODULUS
C
      CM=SCK
      Y=X
      IF(SCK)3,1,4
    1 D=DEXP(X)
      A=1.D0/D
      B=A+D
      CN=2.D0/B
      DN=CN
      A=(D-A)/2.D0
      SN=A*CN
C         DEGENERATE CASE SCK=0 GIVES RESULTS
C              CN X = DN X = 1/COSH X
C              SN X = TANH X
    2 RETURN
C
C         JACOBIS MODULUS TRANSFORMATION
C
    3 D=1.D0-SCK
      CM=-SCK/D
      D=DSQRT(D)
      Y=D*X
    4 A=1.D0
      DN=1.D0
      DO 6 I=1,12
      L=I
      ARI(I)=A
      CM=DSQRT(CM)
      GEO(I)=CM
      C=(A+CM)*5.D0
      IF(DABS(A-CM)-1.D-9*A)7,7,5
    5 CM=A*CM
    6 A=C
C
C         START BACKWARD RECURSION
C
    7 Y=C*Y
      SN=DSIN(Y)
      CN=DCOS(Y)
      IF(SN)8,13,8
    8 A=CN/SN
      C=A*C
      DO 9 I=1,L
      K=L-I+1
      B=ARI(K)
      A=C*A
      C=DN*C
      DN=(GEO(K)+A)/(B+A)
    9 A=C/B
      A=1.D0/DSQRT(C*C+1.D0)
      IF(SN)10,11,11
   10 SN=-A
      GO TO 12
   11 SN=A
   12 CN=C*SN
   13 IF(SCK)14,2,2
   14 A=DN
      DN=CN
      CN=A
```

```
      SN=SN/D
      RETURN
      END



      SUBROUTINE A37IMS(N,A)
      IMPLICIT REAL*8(A-H,O-Z)
  250 FORMAT(1HO,20HMATRIX A IS SINGULAR)
      DIMENSION A(11,11),B(11),C(11),IXIMAX(11),JXJMAX(11)
      DIMENSION ICHK(11),JCHK(11),D(11,11)
      ISING=0
      MC=N
      DO 132 J=1,N
      ICHK(J)=0
  132 JCHK(J)=0
  145 DO 420 IXA=1,N
      AMAX=0.0D00
      DO 160 I=1,N
      IF(ICHK(I))146,146,160
  146 DO 158 J=1,N
      IF(JCHK(J))147,147,158
  147 IF(AMAX-DABS(A(I,J)))150,158,158
  150 AMAX=DABS(A(I,J))
      IMAX=I
      JMAX=J
  158 CONTINUE
  160 CONTINUE
      ICHK(IMAX)=1
      JCHK(JMAX)=1
      IXIMAX(IXA)=IMAX
      JXJMAX(IXA)=JMAX
      IF(AMAX-1.0E-30)240,290,290
  240 WRITE(6,250)
      ISING=1
      STOP
  290 DUM=A(IMAX,JMAX)
      DO 310 J=1,N
      IF(JCHK(J))304,304,310
  304 A(IMAX,J)=A(IMAX,J)/DUM
  310 CONTINUE
      DO 380 I=1,N
      IF(I-IMAX)320,380,320
  320 DO 340 J=1,N
      IF(JCHK(J))330,330,340
  330 A(I,J)=A(I,J)-A(I,JMAX)*A(IMAX,J)
  340 CONTINUE
  380 CONTINUE
  420 CONTINUE
      DO 430 J=1,N
      JT=JXJMAX(J)
      DO 430 I=1,N
  430 D(I,J)=A(I,JT)
      DO 440 I=1,N
      DO 440 J=1,N
  440 A(I,J)=0.0
      DO 450 I=1,N
  450 A(I,I)=1.0
      DO 560 IXA=1,N
      DO 452 I=1,N
  452 B(I)=D(I,IXA)
      IMAX=IXIMAX(IXA)
      JMAX=JXJMAX(IXA)
      DO 460 J=1,N
  460 A(IMAX,J)=A(IMAX,J)/B(IMAX)
      DO 558 I=1,N
      IF(I-IMAX)470,558,470
  470 DO 480 J=1,N
  480 A(I,J)=A(I,J)-B(I)*A(IMAX,J)
  558 CONTINUE
  560 CONTINUE
```

```
      DO 570 IXA=1,N
      IMAX=IXIMAX(IXA)
      JMAX=JXJMAX(IXA)
      IF(IMAX-JMAX)565,570,565
  565 DO 566 J=1,N
      DUM=A(IMAX,J)
      A(IMAX,J)=A(JMAX,J)
  566 A(JMAX,J)=DUM
      DO 569 I=IXA,N
      IF(IXIMAX(I)-JMAX)569,567,569
  567 IXIMAX(I)=IMAX
      GO TO 570
  569 CONTINUE
  570 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TRPOSE(A,R,N,M)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),R(1)
      IR=0
      DO 10 I=1,N
      IJ=I-N
      DO 10 J=1,M
      IJ=IJ+N
      IR=IR+1
   10 R(IR)=A(IJ)
      RETURN
      END
```

Listings of subroutines STATE, DOTPRO, TILDE, MATMUL and DATA appear in the listing of program DUMRA.