(6)

National Defence / Défense nationale

STRAPDOWN NAVIGATION ALGORITHMS IMPLEMENTED IN THE INERTIAL AND GEOGRAPHIC FRAMES

by

D.F. Liang

and

R.H. Johnson

DEFENCE RESEARCH ESTABLISHMENT OTTAWA

TECHNICAL NOTE 82-25

Canada

September 1983
Ottawa

STRAPDOWN NAVIGATION ALGORITHMS IMPLEMENTED IN THE INERTIAL AND GEOGRAPHIC FRAMES

National Defence Headquarters
Ottawa, Ontario, Canada

Sep 83

National Defence / Défense nationale

STRAPDOWN NAVIGATION ALGORITHMS IMPLEMENTED IN THE INERTIAL AND GEOGRAPHIC FRAMES

by

D.F. Liang

and

R.H. Johnson

DEFENCE RESEARCH ESTABLISHMENT OTTAWA

TECHNICAL NOTE 82-25

PCN
32G

September 1983
Ottawa

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA R & D

Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified

1 ORIGINATING ACTIVITY
Defence Research Establishment Ottawa
National Defence Headquarters
Ottawa, Ontario K1A 0Z4

2a DOCUMENT SECURITY CLASSIFICATION
Unclassified

2b GROUP

3 DOCUMENT TITLE
STRAPDOWN NAVIGATION ALGORITHMS IMPLEMENTED IN THE INERTIAL AND GEOGRAPHIC FRAMES

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)
Technical Note

5 AUTHOR(S) (Last name, first name, middle initial)
LIANG, DAVID FESENG
JOHNSON, ROBERT HARVEY

6 DOCUMENT DATE
September 1983

7a TOTAL NO OF PAGES
64

7b NO OF REFS
11

8a PROJECT OR GRANT NO
32G

9a ORIGINATOR'S DOCUMENT NUMBER(S)
DREO-TN - 82-25

8b CONTRACT NO

9b OTHER DOCUMENT NO (S) (Any other numbers that may be assigned this document)

10 DISTRIBUTION STATEMENT
Unlimited Distribution

11 SUPPLEMENTARY NOTES

12. SPONSORING ACTIVITY
DREO
ELECTRONICS DIVISION

13 ABSTRACT

This report presents two strapdown navigation software processing algorithms, one implemented in the inertial frame and the other in the local-level geographic frame.

The presentation includes derivation of various mechanization equations to obtain positional and navigational information from input data provided by three gyros and accelerometers. Directional cosine matrix and quaternion update procedures together with initialization schemes are also presented. In the inertial frame implementation a new exact algorithm was derived to transform inertially referenced data into geographic coordinates.

Simulation results of a C141 race-track flight profile are presented to indicate the level of performance accuracy for the above two processing algorithms. Complete listings of software programs are also presented in the Appendices.

DSIS
77-065

UNCLASSIFIED
Security Classification

KEY WORDS

STRAPDOWN NAVIGATION
NAVIGATION ALGORITHMS
GEOGRAPHICAL COMPUTATIONAL FRAME
INERTIAL NAVIGATION FRAME

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the organization issuing the document.

2a. DOCUMENT SECURITY CLASSIFICATION: Enter the overall security classification of the document including special warning terms whenever applicable.

2b. GROUP: Enter security reclassification group number. The three groups are defined in Appendix 'M' of the DRB Security Regulations.

3. DOCUMENT TITLE: Enter the complete document title in all capital letters. Titles in all cases should be unclassified. If a sufficiently descriptive title cannot be selected without classification, show title classification with the usual one-capital-letter abbreviation in parentheses immediately following the title.

4. DESCRIPTIVE NOTES: Enter the category of document, e.g. technical report, technical note or technical letter. If appropriate, enter the type of document, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the document. Enter last name, first name, middle initial. If military, show rank. The name of the principal author is an absolute minimum requirement.

6. DOCUMENT DATE: Enter the date (month, year) of Establishment approval for publication of the document.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the document.

8a. PROJECT OR GRANT NUMBER: If appropriate, enter the applicable research and development project or grant number under which the document was written.

8b. CONTRACT NUMBER: If appropriate, enter the applicable number under which the document was written.

9a. ORIGINATOR'S DOCUMENT NUMBER(S): Enter the official document number by which the document will be identified and controlled by the originating activity. This number must be unique to this document.

9b. OTHER DOCUMENT NUMBER(S): If the document has been assigned any other document numbers (either by the originator or by the sponsor), also enter this number(s).

10. DISTRIBUTION STATEMENT: Enter any limitations on further dissemination of the document, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this document from their defence documentation center."

(2) "Announcement and dissemination of this document is not authorized without prior approval from originating activity."

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document, even though it may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall end with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (TS), (S), (C), (R), or (U).

The length of the abstract should be limited to 20 single-spaced standard typewritten lines, 7½ inches long.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a document and could be helpful in cataloging the document. Key words should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context.

ABSTRACT

This report presents two strapdown navigation software processing algorithms, one implemented in the inertial navigational frame and the other in the local-level geographic frame.

The presentation includes derivation of various mechanization equations to obtain positional and navigational information from input data provided by three gyros and accelerometers. Directional cosine matrix and quaternion update procedures together with initialization schemes are also presented. In the inertial frame implementation a new exact algorithm was derived to transform inertially referenced data into geographic coordinates.

Simulation results of a C141 race-track flight profile are presented to indicate the level of performance accuracy for the above two processing algorithms. Complete listings of software programs are also presented in the Appendices.

RÉSUMÉ

Ce rapport présente deux algorithms de traitement par logiciel de "strapdown navigation", un implanté dans le contexte de navigation inertiale, et l'autre dans le contexte de navigation géographique locale.

Le présentation inclus des dérivations de diverses équatios de mécanisation afin d'obtenir de l'information de position et de navigation des donneés d'entrée venant de trois gyroscopes et accéléromètres. Une matrice des cosinus de direction et une procédure de renouvel lement de "quaternion" ainsi qu'une routine d'initialisation sont aussi présenteés. Dans l'implantation par inertie, un nouvel algorithme exact a été derivé afin de transformer les "donneés d'inertie" en coordonneés géographiques.

Les résultats de simulation d'un profil de vol d'avions C141 sont présentés pour indequer le degré de précision obtenu avec les deux algorithmes mentionnés ci-haut. Un relevé des programmes logiciels est aussi présenté en appendice.

TABLE OF CONTENTS

PAGE

ABSTRACT/RÉSUMÉ iii

TABLE OF CONTENTS iv

LIST OF ILLUSTRATIONS vi

1.0 GENERAL INTRODUCTION 1

1.1 Introduction 1

1.2 General Description of Coordinate Frames 2

2.0 INERTIAL - FRAME NAVIGATOR 3

2.1 Introduction 3

2.2 Attitude Computation 4

2.3 Gravity Model 5

2.4 Position and Velocity Computations 6

2.5 Coriolis Correction 6

2.6 Computation of Navigation Quantities 7

2.7 Navigator Initialization 9

2.8 Navigator Block Diagram 11

2.9 Test Results 11

3.0 LOCAL-LEVEL NAVIGATOR 11

3.1 Introduction 11

3.2 Gravity Model 12

3.3 Quaternion Attitude Update Algorithms 13

3.4 Mechanization Equations 13

TABLE OF CONTENTS (CONT'D)

PAGE

3.5 Navigator Block Diagram 15

3.6 Test Results 15

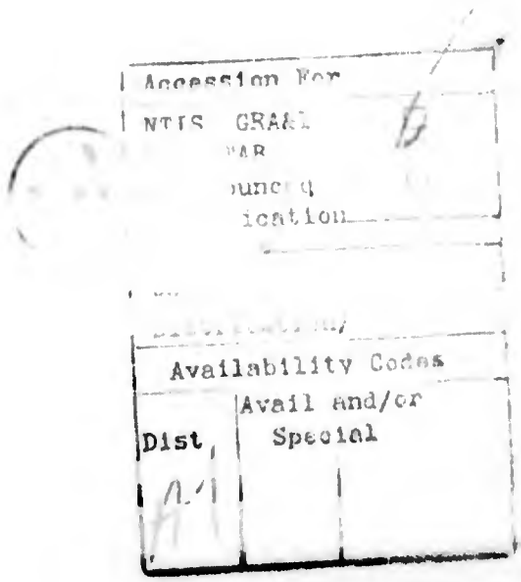
4.0 CONCLUSIONS.....16

5.0 REFERENCES 16

APPENDIX I 27

APPENDIX II 41

APPENDIX III 49



LIST OF ILLUSTRATIONS

PAGE

FIGURE 1 - NAVIGATOR SIMULATION 2

FIGURE 2 - LOCAL-LEVEL GEOGRAPHIC FRAME 3

FIGURE 3 - LATITUDE-ALTITUDE REPRESENTATION 17

FIGURE 4 - INERTIAL-FRAME NAVIGATOR 18

FIGURE 5 - 'RACETRACK' FLIGHT PROFILE 19

FIGURE 6 - NAVIGATOR POSITION ERROR (METERS) VS. TIME (SEC) 20

FIGURE 7 - POSITION ERROR-TRANSFORMATION BEFORE UPDATE 21

FIGURE 8 - POSITION ERROR-TRANSFORMATION AFTER UPDATE 22

FIGURE 9 - LOCAL-LEVEL NAVIGATOR BLOCK DIAGRAM 23

FIGURE 10 - NAVIGATOR'S POSITION ERROR (METERS) VS. TIME (SEC) 24

FIGURE 11 - NAVIGATOR'S ALTITUDE ERROR (METERS) VS. TIME (SEC) 25

1.0 GENERAL INTRODUCTION

1.1 Introduction

Conventional gimballed inertial navigation systems (INS) are now in comparatively widespread application, in vehicles as diverse as civil and military aircraft, missiles, submarines and space-craft.

In a gimballed inertial system, the gyros and accelerometers are mounted on a "stable platform". The gyro outputs are used to drive gimbal motors in such a way as to maintain the platform in a known orientation, irresspective of the vehicle motion, and the accelerometers measure the acceleration of the vehicle along all three axes, with reference to the platform itself. Integrating the specific-force indications from the accelerometers, with a correction of gravity, yields the velocity and position information of the vehicle. By contrast, the gyros and accelerometers of a strapdown system are rigidly attached to the structure of the vehicle, and therefore provide signals proportional to specific-force and angular rate with respect to inertial space coordinates in the vehicle body axes. The transformation from the sensor body axes to inertial reference coordinates is digitally processed by a computer rather than gimbal mechanized. The advent of the microprocessor has made it attractive to replace highly complex and fallible moving components by computer power inherent in silicon chips. However, the simplicity of the strapdown system is gained at the expense of subjecting the inertial sensors to a more severe vehicle motion environment.

With the aim of developing navigation technology where new capabilities are emerging rather than in those areas which are technically mature, the DREO Navigation sub-program has undertaken some research and development work in the following areas of strapdown navigation technology:

i) The development analysis of an aiding GPS/Strapdown inertial navigation system [1-2].

ii) The development of a Strapdown Aircraft Heading and Reference System [3], and

iii) A Spot-light Mode Synthetic Aperture Radar motion sensing technique.

Since the structure of the navigation algorithms depends upon the coordinate frame chosen for navigation computation, the above activities necessitate the development of two distinct sets of navigation software mechanization schemes, in the following two coordinate frames:

i) Inertial, and

ii) Local-level geographic.

This report presents some of the design rational and analytical descriptions of these two sets of navigation algorithms for deriving attitude, heading and navigation information from a strapdown system.

Simulation results are presented to illustrate typical test results and the achievable levels of performance accuracy.

Both of these mechanization algorithms called navigators from now on, were programmed in a Xerox Sigma 9 Fortran environment. Program listings appear in Appendices 1 and 3. The navigators were driven by a modified version of the flight profile simulator [4] called PROFGEN, implemented at DREO. PROFGEN generates profiles assuming that the aircraft is an ideal point mass without any side-slipping or crabbing motion. It also generates the corresponding ideal position and velocity information to use as a reference for comparison of the navigators' outputs (Fig. 1).

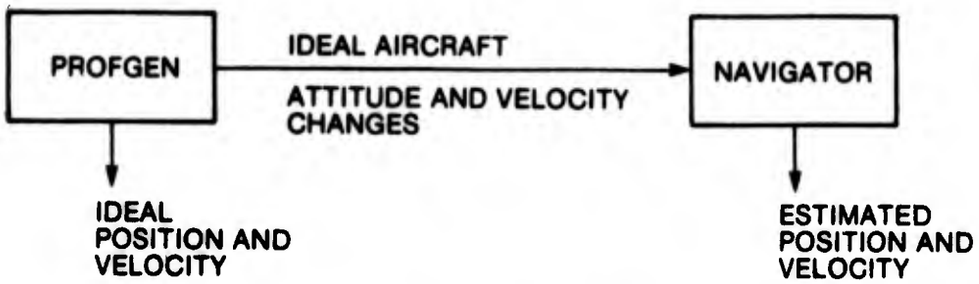


FIGURE 1: NAVIGATOR SIMULATION

1.2 General Description of Coordinate Frames

For a strapdown system, there are several algorithms possible to compute navigation quantities of position and velocity from the outputs of the inertial measurement unit (IMU). These algorithms in general correspond to the choice of Cartesian reference frame in which the algorithm is mechanized. AT DREO, two mechanization schemes were chosen for implementation. Namely, one inertial frame and another one in local-level geographic frame.

For strapdown systems, since the gyros measure body angular rate with respect to inertial space, the inertial frame (Fig. 2) is most appropriate in navigation computations. The origin of this frame is taken to be the earth's centre, its x-axis through the equator at the Greenwich meridian at time zero, its z-axis through the north pole and the y-axis through the equator completing the right-handed system. It is also assumed that the inertial frame is non-rotating relative to the stars, i.e. non-rotating in the inertial space.

The local-level geographic frame (Fig. 2) is typically adopted, when the navigation information output in geographic coordinates is desired by the user. The origin of this navigator is the location of the strapdown system, with its x-axis pointing north, its z-axis normal to the earth's reference ellipsoid and its y-axis pointing in an easterly direction

completing the right-handed system. Clearly the local level frame rotates in inertial space because of the earth's rotation and because of the system's own motion over the earth's surface.

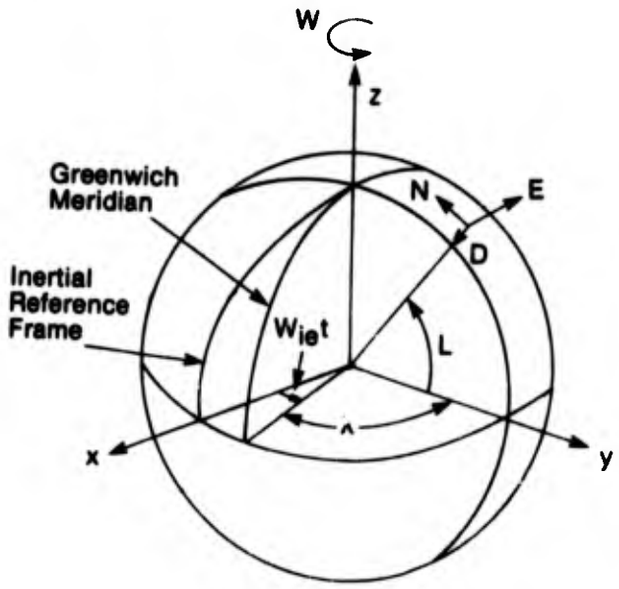


FIGURE 2: INERTIAL AND LOCAL-LEVEL GEOGRAPHIC FRAME

Even though a navigator software designer is at liberty to select the coordinate frame of his navigation computations, the coordinates of the IMU are rigidly attached to the body of the vehicle. Ideally speaking, these three axes correspond to the three input axes of the strapdown gyros and accelerometers. Therefore, it is necessary to define the body reference frame, so that the orientation of the computational reference can be determined with respect to this reference frame. The body frame has its origin at the strapdown IMU, its x-axis along the longitudinal direction of the vehicle body, its y-axis pointing out through the right side, and the z axis down through the vehicle's underside.

2.0 INERTIAL-FRAME NAVIGATOR

2.1 Introduction

Conceptually, the inertial-frame navigator is perhaps the simplest for a strapdown system. The reason for this is that the strapdown sensors, being rigidly fixed to the vehicle body, directly sense any rotational or linear acceleration of the vehicle body with respect to the inertial space. Hence, any angular motion sensed by the gyros of the IMU can be employed in a straight-forward manner to keep track of the vehicle's relative orientation to the inertial reference frame. Using this relative orientation information, the velocity increments sensed by the accelerometers can be transformed to the inertial-frame (with a correction made to remove acceleration components due to gravity), and finally integrated to provide velocity and position information.

2.2 Attitude Computation

The distinguishing algorithm of strapdown inertial systems, as opposed to gimballed inertial systems, is the calculation of 3 x 3 direction-cosine transformation matrix that relates the orientation of the accelerometer measurement axes to the selected inertial navigation frame. This matrix is updated continuously using gyro-sensed attitude increments $\Delta\theta^B$, by a third-order quaternion scheme which is generally regarded in the literature [5, 6] as being more accurate and requiring less computer time than the corresponding direct direction cosine algorithm when operated at the same sampling rate. If $q_{t+\Delta t}$ is considered as the updated quaternion and q_t is the previous quaternion, we have

$$q_{t+\Delta t} = (q_1, q_2, q_3, q_4)$$

$$= q_t\left[1 + \frac{\Delta\theta^B}{2} - \frac{\Delta\theta^B \cdot \Delta\theta^B}{8} - \frac{\Delta\theta^B \cdot \Delta\theta^B}{48}\Delta\theta^B - \frac{\Delta\theta^B \times \Delta\theta^{B*}}{24}\right] \qquad (1)$$

where $\Delta\theta^{B*}$ is the attitude increment of the previous interval, and the multiplication of q_t and various terms in the brackets are derived on the basis of the following rule:

If $\quad q_1 = \lambda_1 + \overline{P_1}$

$$q_2 = \lambda_2 + \overline{P_2}$$

then

$$q_1 q_2 = \lambda_1 \lambda_2 + \lambda_1\overline{P_2} + \lambda_2\overline{P_1} - \overline{P_1} \cdot \overline{P_2} + \overline{P_1} \times \overline{P_2} \qquad (2)$$

In order to minimize scale and skew error effects, the quaternion q must be periodically renormalized to satisfy the constraint

$$q^T q = 1$$

which only requires a vector multiplication and a simple scalar division

$$q_n = \frac{q}{(q^T q)^{\frac{1}{2}}} = \frac{q}{\sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2}} \qquad (3)$$

Then the updated direction cosine matrix C_B^I can be constructed from the elements of q at any time using the relation:

$$C_B^I = \begin{bmatrix} 1 - 2(q_3^2 + q_4^2) & 2(q_2q_3 - q_1q_4) & 2(q_2q_4 + q_1q_3) \\ 2(q_2q_3 + q_1q_4) & 1 - 2(q_2^2 + q_4^2) & 2(q_4q_3 - q_1q_2) \\ 2(q_2q_4 - q_1q_3) & 2(q_4q_3 + q_1q_2) & 1 - 2(q_2^2 + q_3^2) \end{bmatrix} \qquad (4)$$

2.3 Gravity Model

The gravity model implemented is taken from Britting's work [7], and is a function only of the inertial system's inertial frame coordinates (x, y, z).

The radial component is given by:

$$G_r = \frac{-\mu}{R^2}\left[1 - \frac{3}{2} J_2 \left(\frac{R_e}{R}\right)^2 (3\cos^2\phi - 1) - 2J_3 \left(\frac{R_e}{R}\right)^3 \cos\phi\,(5\cos^2\phi - 3)\right.$$

$$\left. - \frac{5}{8} J_4 \left(\frac{R_e}{R}\right)^4 (35\cos^4\phi - 30\cos^2\phi + 3)\right] \qquad (5)$$

and the colatitude component by:

$$G_\phi = 3\frac{\mu}{R^2}\left(\frac{R_e}{R}\right)^2 \sin\phi\cos\phi$$

$$x\left[J_2 + \frac{1}{2} J_3 \left(\frac{R_e}{R}\right) \sec\phi\,(5\cos^2\phi - 1) + \frac{5}{6} J_4 \left(\frac{R_e}{R}\right)^2 (7\cos^2\phi - 3)\right] \qquad (6)$$

where $R = \sqrt{x^2 + y^2 + z^2}$

$R_e = 2092.5640$ ft. = equatorial radius

ϕ = geocentric colatitude

$\mu = 1.407645 \times 10^{16}$ ft.^{3}/sec^2

$J_2 = .0010823$

$J_3 = -2.3 \times 10^{-6}$

$J_4 = -1.8 \times 10^{-6}$

Hence $G_x = (G_r + G\phi \tan L_c)x/R$

$G_y = (G_r + G\phi \tan L_c)y/R$

$G_z = (G_r - G\phi \cot L_c)z/R$ Where L_c = geocentric latitude

(G_x, G_y, G_z) is the inertial frame vector which ideally gives the gravitational force of mass attraction exerted by the earth and sensed by the accelerometers.

2.4 Position and Velocity Computations

The relations between position and inertial measurements, expressed in an inertial coordinate frame, are

$$\dot{R}_I = V^I \tag{7}$$

and

$$\dot{V}^I = A^I - G^I \tag{8}$$

where R and V are respectively, inertial frame position and velocity coordinates, and A^I is the non-gravitational specific force measured by the accelerometers. The gravitational effect G^I due to mass attraction which is sensed by the accelerometers, must be subtracted in the computation of the total rate of change of velocity.

Solutions of Equations (7) and (8) yield:

$$V^I(t \quad \Delta t) = V^I(t) + \Delta V^I - G^I \Delta t$$

and

$$R^I(t \quad \Delta t) = R^I(t) + V^I(t)\Delta t$$

$$= \frac{1}{2} \Delta t(\Delta V^I - G^I \Delta t)$$

2.5 Coriolis Correction

If the strapdown system is rotating in inertial space while velocity increments are being sensed by the body-mounted accelerometers, certain errors referred to from now on as "coriolis" effects are introduced into the transformed velocity increments in inertial frame. Some correction for these error terms, however small, should be made. Instead of a detailed, rigourous approach as in reference [8], a simpler but nonetheless effective method was followed. This method effectively transforms the sensed velocity changes in body frame by the direction cosine matrix C_B^I corresponding to the midpoint of the update interval. Thus, most of the dissymmetries caused by transforming body velocity changes via a direction cosine matrix at either end of the update interval are avoided.

Let C_1 and C_2 be the C_B^I matrices at the beginning and end of an update interval, i.e. at times t and t + Δt respectively. Let ΔV^B be the sensed velocity increments and $\Delta\theta^B$ the sensed angular increments in body frame.

Then, to a first order approximation, $C_2 = C_1(I + M)$

where

$$M = \begin{bmatrix} 0 & -\Delta\theta_z^B & \Delta\theta_y^B \\ \Delta\theta_z^B & 0 & -\Delta\theta_x^B \\ -\Delta\theta_y^B & \Delta\theta_x^B & 0 \end{bmatrix}$$

Hence,

$$\frac{C_1\,\Delta V^B + C_2\,\Delta V^B}{2} = \frac{C_1\,\Delta V^B + C_1(I + M)\Delta V^B}{2}$$

$$= \frac{C_1\,\Delta V^B + C_1\,\Delta V^B + C_1 M \Delta V^B}{2}$$

$$= C_1\,\Delta V^B + C_1\,\frac{M}{2}\,\Delta V^B$$

$$= C_1(I + \frac{M}{2})\Delta V^B$$

This shows that transforming ΔV^B by C_1 and C_2 and then averaging is equivalent (to a first order approximation) to transforming ΔV^B by the direction cosine matrix at $t + \frac{\Delta t}{2}$. Transforming by C_1 and C_2 and averaging is what is carried out in the implemented coriolis correction algorithm. For test results of the correction, see Figs. 6, 7 and 8 in Section 2.9.

2.6 Computation of Navigation Quantities

2.6.1 Latitude-Altitude Algorithm

Since the inertial frame navigator so far described provides position and velocity information in inertial frame, and these quantities by themselves are of little interest to the average human navigator, transformation to more usable geographic referenced quantities is therefore necessary. This Section describes a new algorithm developed at DREO which converts inertial frame coordinates (x, y, z) into latitude, longitude and altitude.

An arbitrary point P(x, y, z) is assummed to be located above the reference ellipsoid, in inertial frame coordinates. In Fig. 3, the plane containing P and the polar Z-axis is presented, where it is apparent that

$$w = \sqrt{x^2 + y^2} \tag{10}$$

The altitude h is the distance between P and its projection Q(w, z) on the reference ellipsoid, and can be expressed as

$$h^2 = (w_1 - w)^2 + (z_1 - z)^2 \tag{11}$$

and the ellipsoidal equation is:

$$w_1 = \sqrt{\frac{R_e^2\,R_p^2 - R_e^2\,z_1^2}{R_p^2}} \tag{12}$$

where R_e = earth equatorial radius

and R_p = earth polar radius

L = geographic latitude

Substituting Eq. (12) into Eq. (11) gives

$$h^2 = R_e^2\left(1 - \frac{z_1^2}{R_p^2}\right) - 2w\sqrt{R_e^2\left(1 - \frac{z_1^2}{R_p^2}\right)} + w^2 + z_1^2 - 2zz_1 + z^2 \qquad (13)$$

Since Q is the point on the ellipse with the minimum distance from P, then differentiating Eq. (13) w.r.t. z_1 and setting it to zero yields

$$\sqrt{1 - \frac{z_1^2}{R_p^2}} = \frac{-w\,R_p\,z_1}{R_p^2(z_1 - z) - R_e^2 z_1} \qquad (14)$$

Squaring both sides of Eq. (14) gives us the following quartic expression

$$-\left(\frac{R_p^2 - R_e^2}{R_p^2}\right)^2 z_1^4 + \frac{2(R_p^2 - R_e^2)}{R_p^2} z z_1^3 \qquad (15)$$

$$+\left[\frac{(R_p^2 - R_e^2)^2}{R_p^2} - z^2 - \frac{w^2 R_e^2}{R_p^2}\right] z_1^2 - 2(R_p^2 - R_e^2) z z_1 + R_p^2 z^2 = 0$$

Hence z_1 can be solved for exactly using the standard quartic root formulae or it can be found using the Newton-Raphson iterative technique.

Using the iterative method and a starting value of

$$z_1 = \frac{R_p}{\sqrt{1 + \left(\frac{R_p\,w}{R_e\,z}\right)^2}}$$

Corresponding to the point p' in Fig. 3, it was found that only 2 or 3 iterations are necessary at most to find z_1. Starting with Eq. (12) and then using Eq. (11) the altitude h of the point P can be solved for.

Finally since P and Q have the same geographic latitude L, the latitude of P is obtained based upon point Q, which implies

$$L = \tan^{-1}\left(\frac{R_e^2\,z_1}{R_p^2\,w}\right)$$

Furthermore, for any point inside the ellipse, the quartic relationship of Eq. (15) also generates distance-to-ellipse extrema and it can be shown geometrically that for depths smaller than half of the earth radius, the Newton-Raphson technique will converge to the proper (minimum distance-producing) root of Eq. (15).

In order to test this latitude-altitude algorithm, it was programmed in a stand-alone fashion along with Bowring's algorithm [8] for comparison. The FORTRAN program listings are presented in Appendix 2. At a single iteration both algorithms have comparable accuracy and take comparable lengths of CPU time. At 2 iterations both have comparable accuracy improvements, but the new algorithm takes significantly (approx. 20%) less CPU time for execution than Bowring's. Aberrations in the behaviour of the algorithms occurred at the equator and at the earth's poles. At the equator, the algorithm presented here takes about twice as much CPU time as normal. At the poles, Bowring's algorithm breaks down and gives wrong answers. Essentially, the DREO algorithm has the advantage of being simple in concept and provides an analytically exact transformation from inertial frame coordinates to geographic position.

2.6.2 Velocity in Local-Navigational Frame

To compute the velocity information in the local navigational frame, it is necessary to take into consideration the effects of the earth's rotation. This relation is simply:

$$V^N = C_I^N \, (V^I - \Omega_{ie}^I \, R^I)$$

where C_I^N is a standard function of longitude λ and latitude L.

2.7 Navigator Initialization

To initialize the operation of the navigator, it is essential to insert externally derived navigation information on latitude, longitude and altitude. The following algorithm summarizes the initialization process:

a) Direction Cosine Matrix C_B^I

$$C_B^I = C_N^I \, C_B^N,$$

where

$$C_N^I = \begin{bmatrix} -\sin L \cos\lambda & -\sin\lambda & -\cos L \cos\lambda \\ -\sin L \sin\lambda & \cos\lambda & -\cos L \sin\lambda \\ \cos L & 0 & -\sin L \end{bmatrix}$$

$$C_B^N = \begin{bmatrix} \cos H \cos P & \cos H \sin P \sin R - \sin H \cos R & \cos H \sin P \cos R + \sin H \sin R \\ \sin H \cos P & \sin H \sin P \sin R + \cos H \cos R & \sin H \sin P \cos R - \cos H \sin R \\ -\sin P & \cos P \sin R & \cos P \cos R \end{bmatrix}$$

and L = latitude, λ = longitude, H = heading, P = pitch, R = roll

b) Quaternion q

$$q_1 = (\sqrt{1 + C_{11} + C_{22} + C_{33}})/2$$

$$q_2 = (C_{32} - C_{23})/4q_1$$

$$q_3 = (C_{13} - C_{31})/4q_1$$

$$q_4 = (C_{21} - C_{12})/4q_1 \qquad \text{where } (C_{ij}) = C_B^I$$

Then normalizing, $q_i = q_i/\sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2}$ $\qquad (i = 1,2,3,4)$

This method is due to Grubin [9, 10].

c) Inertial Frame Position Vector (x,y,z)

Let $R = R_e/\sqrt{1 - e^2 \sin^2 L}$ $\qquad$ where R_e = equatorial radius,

e = earth's eccentricity

Then $x = (R + h)\cos L \cos\lambda$

$y = (R + h)\cos L \sin\lambda$

$$z = \left(\frac{R R_p^2}{R_e^2} + h\right)\sin L$$

where R_p = polar radius, h = altitude

d) Inertial Frame Velocity Vector V^I

Letting $w_{ie} = .7292115147 \times 10^{-4}$ rad/sec (earth's rotational rate) and

$$V = w_{ie}\sqrt{x^2 + y^2},$$

then $V_x^I = -V \sin\lambda + v_1$

$V_y^I = V \cos\lambda + v_2$

$V_z^I = v_3$

where (v_1, v_2, v_3) is the inertial frame velocity due to the body possibly not having zero velocity relative to the earth at initialization time. Usually $(v_1, v_2, v_3) = (0, 0, 0)$, i.e. the system is stationary at initialization.

2.8 Navigator Block Diagram

In Fig. 4 is shown a block diagram of the inertial frame navigator with angular increments $\Delta\theta^B$ and velocity increments ΔV^B as inputs. Note that the latitude-altitude algorithm implemented is a new and efficient algorithm developed at DREO.

2.9 Test Results

All testing done using a flight profile simulated by PROFGEN [4] which corresponded to a C141 "racetrack" flight of 1750 seconds duration as shown in Fig. 5. The navigator was iterating at a 50 Hz rate, i.e. $\Delta t = 0.02$ sec.

The starting conditions for this scenario were 45° latitude, -60° longitude, 0 ft. altitude, 90° heading, 0° pitch, 0° roll and 0 ft/s velocity. Total RSS position error of the navigator as a function of time for the whole run is shown in Fig. 6. This is the error with respect to the ideal position generated by PROFGEN, and after 1000 seconds of flight time, the error is only about 15m, which clearly shows the accuracy of this navigator.

In order to demonstrate the improvement due to the coriolis correction, the same scenario was repeated another 2 times without the correction. In the first repetition, the ΔV^B's were transformed to ΔV^I's only before C_B^I was updated. The resulting error is shown in Fig. 7. In the second repetition, the ΔV^B's were transformed only after C_B^I was updated with corresponding error shown in Fig. 8. Clearly, the coriolis correction improves position estimates especially during turning manoeuvres.

3.0 LOCAL-LEVEL NAVIGATOR

3.1 Introduction

In some applications, such as marine navigation, it may be necessary to eliminate the vertical direction from the dynamics of the

navigator. This requirement can be easily met with a coordinate frame which has one axis along the local-vertical, and the other two axes in the local-horizontal plane. For the present purpose, it is convenient to define a local-level geographic coordinate frame which is aligned with the local north-east-down coordinate system. Several features of the inertial frame navigator were carried over to the local-level version, in particular, the coriolis correction and the quaternion attitude update algorithm.

For the purpose of completeness in presentation, the DREO software implementation assumed the presence of all three accelerometers to measure the geographic referenced specific force. A complete software program print out is presented in Appendix III. In view of the intention to use the local-level navigator chiefly for two dimensional (i.e. earth's surface) navigator, it is justifiable to select a simpler and less accurate gravity model. This gravity model is entirely different from that of the inertial frame navigator, since the concept of inertial frame does not have any implication in the local-level navigator.

3.2 Gravity Model

For the local-level navigator, a crude gravity model [7] coordinized in north-east-down coordinates was used. The component, g in the down direciton, accurate up to 20 μg's is

$$g_D = \frac{\mu}{R^2}\left[1 - \frac{3}{4} J_2(1 - 3\cos 2L)\right] - R\, w_{ie}^2 \cos L \cos L_c$$

the north component is -1.63×10^{-8} h sin L cos L and the east component is zero. These components model the earth's mass attraction force plus the centrifugal force due to the earth's rotation.

The geocentric radius is approximated by

$$R \simeq R_o + h.$$

The error in this approximation increases with increasing altitude, but is only about 1 foot even at the altitude of 200,000 ft.

The radius of the reference ellipsoid can be computed from the equation of an ellipse, and is approximately

$$R_o = R_e\left[1 - \frac{\varepsilon}{2}(1 - \cos 2L) + \frac{5}{16}\varepsilon^2 (1 - \cos 4L)\right]$$

where the first and the second term are, respectively accurate to 150 ft and 1 ft.

The geocentric latitude can be written in terms of the geographic latitude L, which is

$$L_c = L - \varepsilon\left(1 - \frac{h}{R_o}\right) \sin 2L$$

3.3 Quaternion Attitude Update Algorithms

The directional-cosine matrix to transform angular velocity from the body frame to local-level geographical frame is given by

$$C_B^N = C_B^N \, \Omega_{NB}^B$$

where

$$\Omega_{NB}^B = \Omega_{IB}^B - \Omega_{IN}^B$$

or simply

$$\Delta\phi^B = \Delta\phi_{NB}^B = \Delta\theta_{IB}^B - \Delta\psi_{IN}^B$$

Then the quaternions of C_B^N are represented by

$$q_{t+\Delta t} = (q_1, q_2, q_3, q_4)$$

$$= q_t\left[1 + \frac{\Delta\phi^B}{2} - \frac{\Delta\phi^B \cdot \Delta\phi^B}{8} - \frac{\Delta\phi^B \cdot \Delta\phi^B}{48}\,\Delta\phi^B - \frac{\Delta\phi^B \times \Delta\phi^{B*}}{24}\right]$$

where $\Delta\phi^{B*} = \Delta\phi^B$ of previous interval.

Hence the DCM matrix C_B^N can be constructed from the quaternions as:

$$C_B^N = \begin{bmatrix} 1 - 2(q_3^2 + q_4^2) & 2(q_2q_3 - q_1q_4) & 2(q_2q_4 + q_1q_3) \\ 2(q_2q_3 + q_1q_4) & 1 - 2(q_2^2 + q_4^2) & 2(q_4q_3 - q_1q_2) \\ 2(q_2q_4 - q_1q_3) & 2(q_4q_3 + q_1q_2) & 1 - 2(q_2^2 + q_3^2) \end{bmatrix}$$

3.4 Mechanization Equations

The rate of change of earth-referenced velocity as seen in the local-level geographical frame is

$$(\dot{V}_E)^N = \dot{V}_E + \Omega_{NE} \times V_E$$

$$= \dot{V}_E - \Omega_{EN} \times V_E$$

where the rate of change of earth-fixed veolicty is:

$$\dot{V}_E = a + g - 2\,\Omega_{IE} \times V_E$$

It is well known that angular velocities Ω_{IE} and Ω_{EN} are given by:

$$\Omega_{IE} = \begin{bmatrix} w_{ie_0} & \text{Cos L} \\ -w_{ie} & \text{Sin L} \end{bmatrix}$$

and

$$\Omega_{EN} = \begin{bmatrix} \dot{l} \cos L \\ -L \\ -\dot{l} \text{ Sin L} \end{bmatrix}$$

We also have:

$$V_E^N = \begin{bmatrix} v_N \\ v_E \\ v_D \end{bmatrix}$$

$$g = \begin{bmatrix} g_N \\ 0 \\ g_N \end{bmatrix}$$

Therefore the rate of change of velocity components are

$$\dot{v}_N = a_N - 1.63 \times 10^{-8} h \text{ Sin L Cos L} + v_D \dot{L} - v_E(\dot{l} + 2 w_{ie}) \text{ Sin L}$$

$$\dot{v}_E = a_E + (\dot{l} + 2 w_{ie}) (v_N \text{ Sin L} + v_D \text{ Cos L})$$

and

$$\dot{v}_D = a_D + g_D - v_N \dot{L} - v_E(\dot{l} + 2 w_{ie}) \text{Cos L}$$

where

$$a_N = C_N \Delta v_N$$

$$a_E = C_E \Delta v_E$$

$$a_D = C_D \Delta v_D$$

The latitude, longitude, and altitude are related to the earth-referenced velocities as:

$$\dot{L} = \frac{V_N}{(r_L + h)}$$

$$\dot{l} = \frac{V_E}{(r_L + h)\cos L}$$

and

$$\dot{h} = -v_D$$

3.5 Navigator Block Diagram

A complete local-level navigator mechanization diagram is presented in Fig. 9. As can be seen, there are two timing loops, a fast inner loop iterated at 50 Hz and a slower outer loop at 2 Hz.

The fast loop continues to update the quaternions and C_B^N matrix to maintain the correct body to navigation frame orientation using the outputs of strapdown gyros, corrected by $\Delta\Psi_{IN}^B$. The velocity increments Δ_V^B in body frame are also transformed to navigation frame velocity increments Δ_V^B, which are then followed by a coriolis correction.

The slow loop iterates once every "nav interval" i.e. every 0.5 seconds. In the middle of the nav interval, the quantities C_N, C_E, C_D are computed for use at the end of each nav interval where position and velocity are calculated. Also, since the $\Delta\psi_N^B$ used over the previous nav interval t to t_{k+} was a prediction based on $w_{IN}^N(k)$, a correction is made at the end of the nav interval using the new knowledge of $w_{IN}^N(k+1)$. Further elaboration and a full analytical development of the local level navigator algorithm is given by Van der Velde [11].

To initialize the navigator, C_B^N is computed from input position and attitude (see Section 2.7, same as the inertial frame navigator). Then the quaternion is calculated from C_B^N using the Grubin [9, 10] method once more.

3.6 Test Results

The Racetrack scenario described earlier was used to test the local level navigator. In Fig. 10, the solid line shows the total RSS position error in meters of this navigator as a function of time. The dotted line is the corresponding position error for the inertial fame navigator repeated for comparision. Most of the local-level navigator's error is due to the cruder gravity model employed. This is demonstrated by the plots of altitude error in meters for both navigators in Fig. 11 (the local level navigator is represented by the solid line) where it can be seen that the local level error in altitude accounts almost entirely for the total position error.

4.0 CONCLUSIONS

This report presents the derivation of two strapdown navigation algorithms for deriving navigation information from a strapdown inertial measurement unit. The navigation processing algorithms are implemented in the inertial navigation frame and the local-level geographic frame. In the inertial frame mechanization, a new exact algorithm was derived to transform inertially referenced data into geographic coordinates. Simulation results and complete listings of software programs are also presented.

5.0 REFERENCES

1. Liang, D.F., Johnson, R.H., "Development of Aiding GPS/Strapdown Inertial Navigation Systems", NATO AGARD Symposium on Advances in Guidance and Control Systems Using Digital Techniques, CP No. 272, Ottawa, May 1979.

2. Dyment, M.J., and Liang, D.F., "Dynamic Performance Analysis of Navstar/GPS Navigation Filters", NATO AGARD Symposium on Precision Positioning and Inertial Guidance Sensors, London, Sept. 1980.

3. Reid, D.B., and Liang, D.F., "Development Analysis of a Strapdown Aircraft Heading and Reference System", 1981 National Aerospace Symposium of the Institute of Navigation, Philadelphia, April 1981.

4. Vinnins, M., "Profgen: An Aircraft Flight Profile Generation Program", DREO Technical Note No. 78-14, Ottawa, Sept. 1978.

5. McKern, R.A., "A Study of Transformation Algorithms For Use In a Digital Computer", M.I.T. Technical Report T-493, Jan. 1968.

6. Bortx, J.E., "A New Concept in Strapdown Inertial Navigation", NASA TR-R-329, Washington, March 1970.

7. Britting, K.R., Inertial Navigation Systems Analysis, Wiley-Interscience, New York, 1971.

8. Levinson, E., "Laser-gyro Strapdown Inertial System Application", AGARD-LS-95 on Strapdown Inertial Systems, May 1978.

9. Grubin, C., "Derivation of the Quaternion Scheme via the Euler Axis and Angle", J. Spacecraft, Vol. 7, No. 10, Oct. 1970.

10. Grubin, C., "Attitude Determination for a Strapdown Inertial System Using the Euler Axis/Angle and Quaternion Parameters", AIAA Guidance and Control Conference, Key Biscayne, Florida, Aug. 1973.

11. Van der Velde, W.E., M.I.T., Unpublished Lecture Note on Strapdown Navigation System.

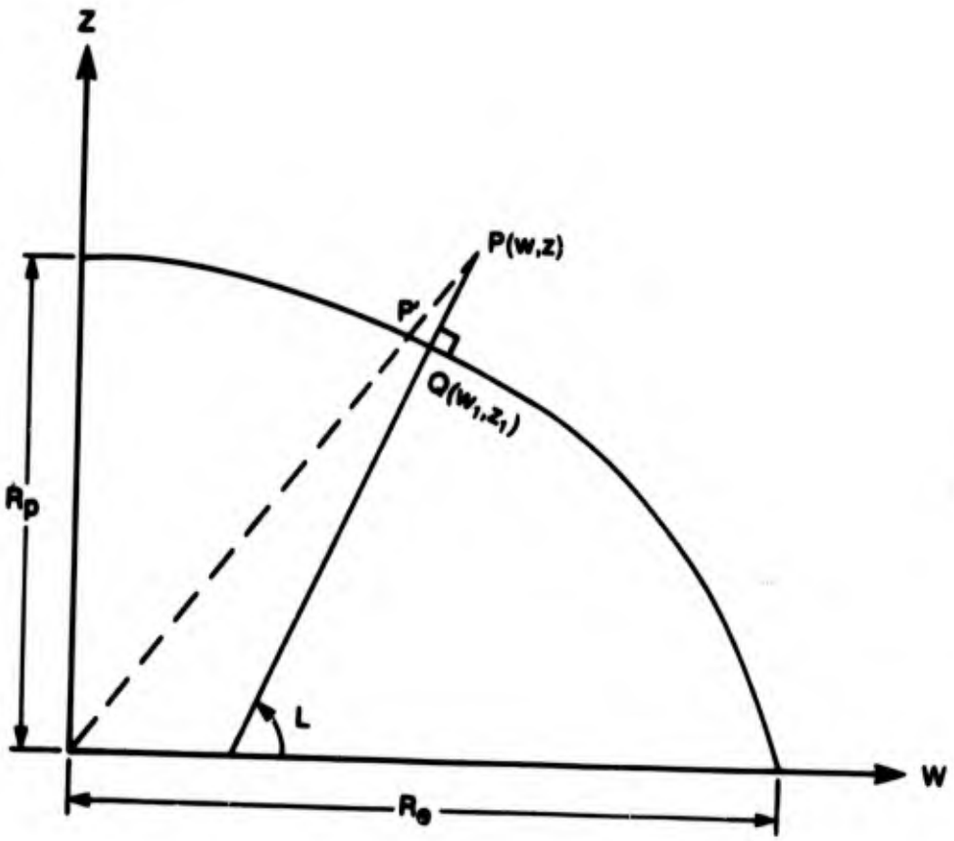


FIGURE 3 - LATITUDE-ALTITUDE REPRESENTATION

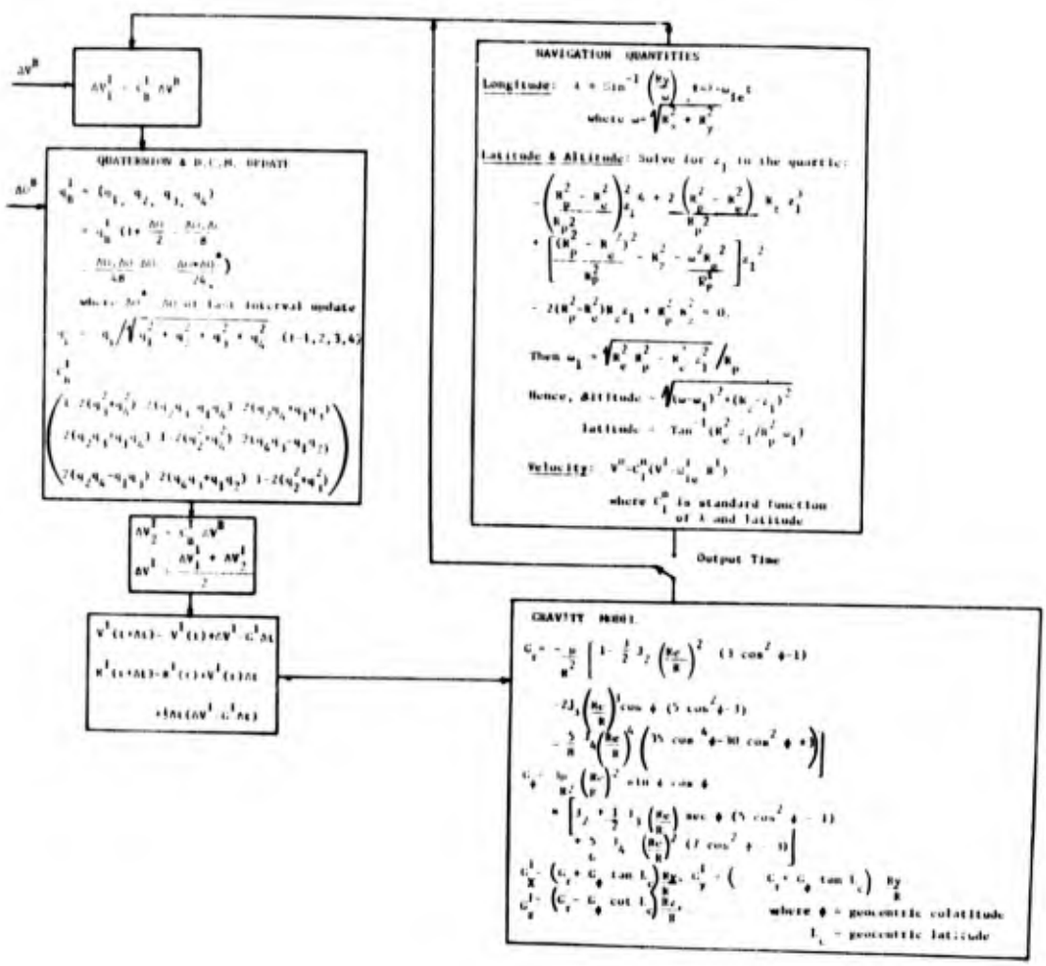


FIGURE 4 - INERTIAL-FRAME NAVIGATOR

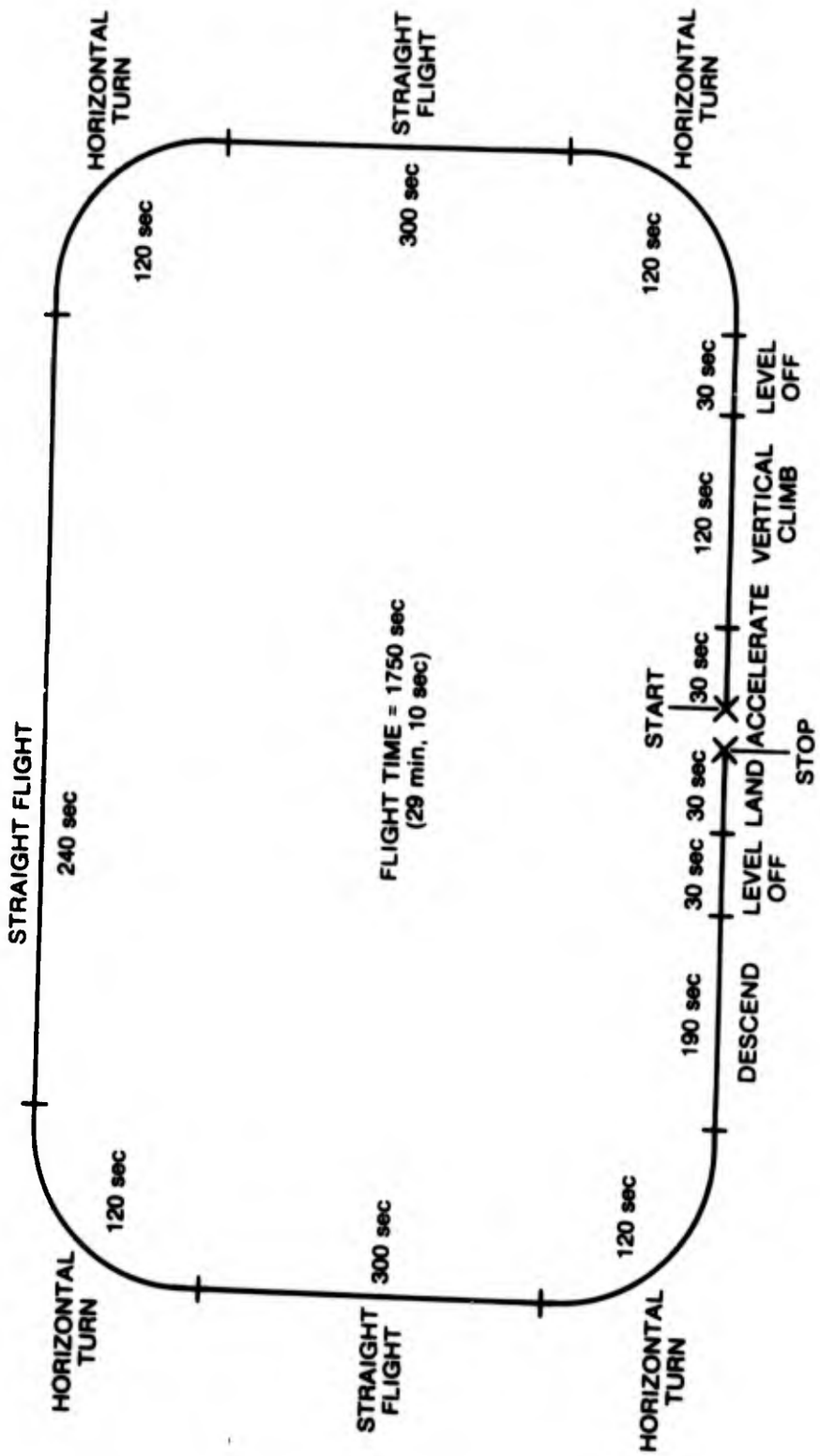


FIGURE 5 - 'RACETRACK' FLIGHT PROFILE

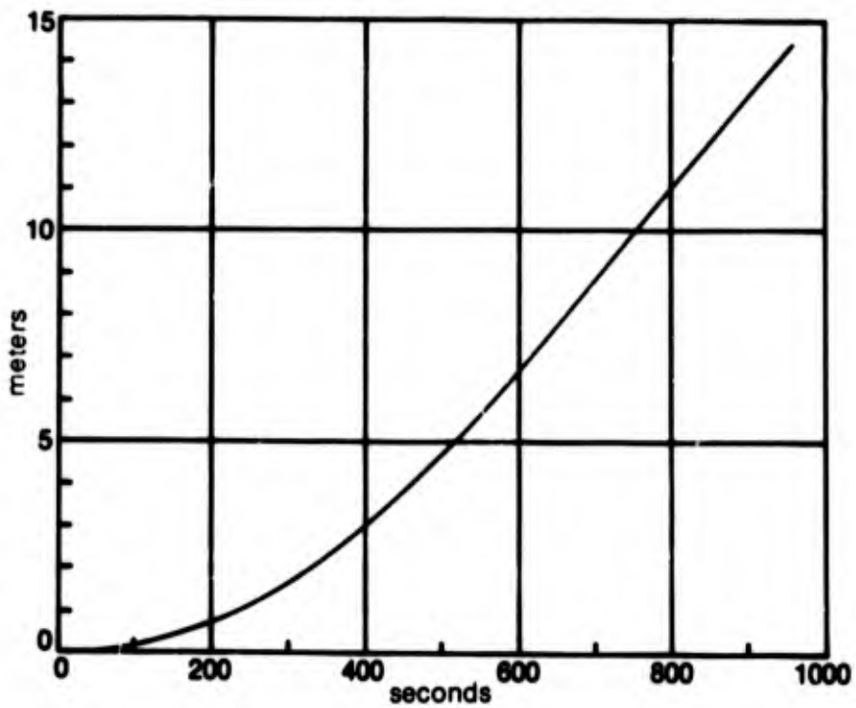


FIGURE 6 - NAVIGATOR POSITION ERROR (METERS) VS. TIME (SEC)

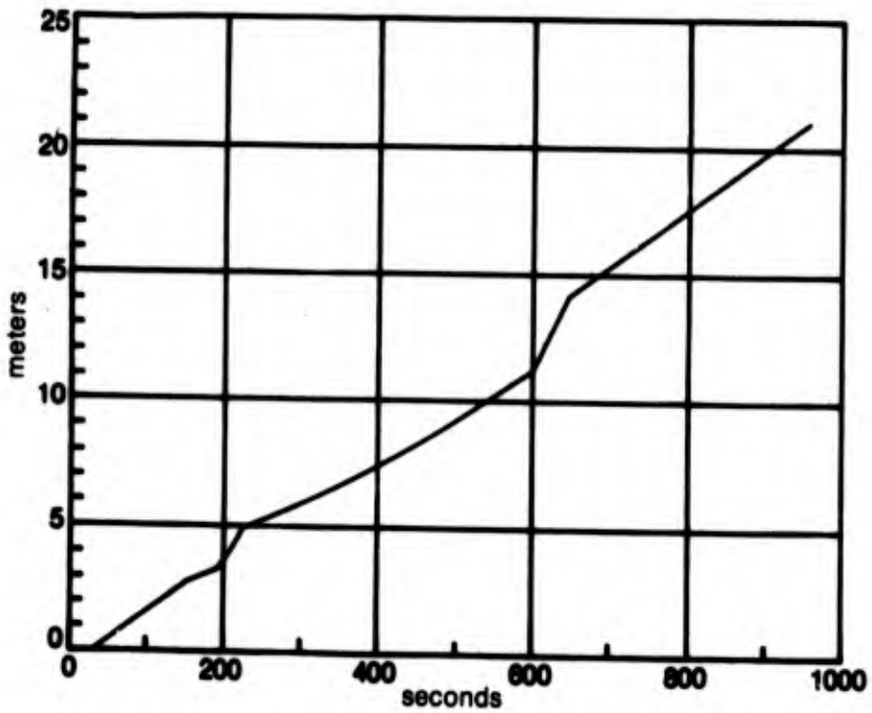


FIGURE 7 - POSITION ERROR - TRANSFORMATION BEFORE UPDATE

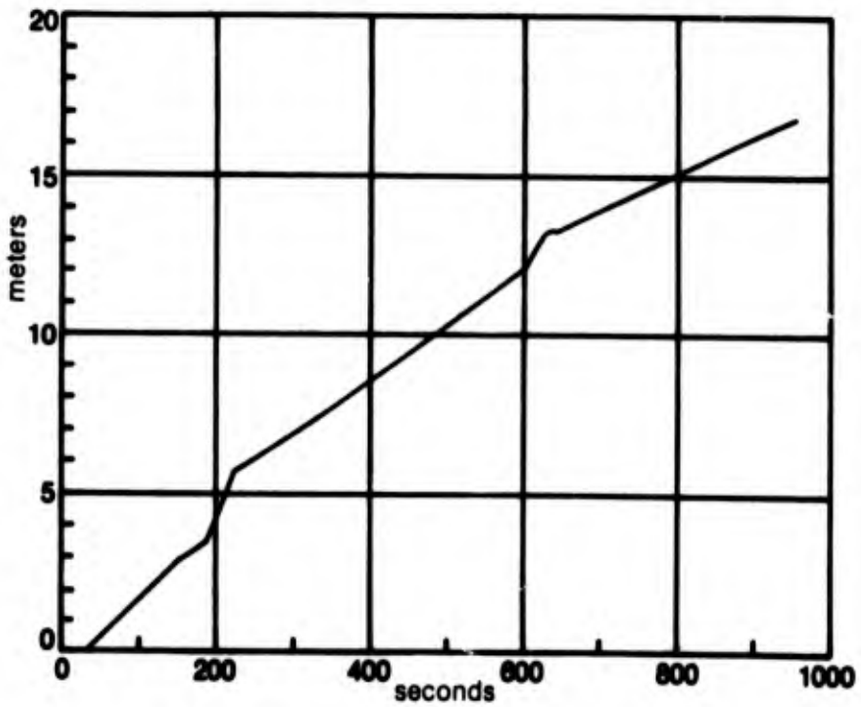


FIGURE 8 - POSITION ERROR - TRANSFORMATION AFTER UPDATE

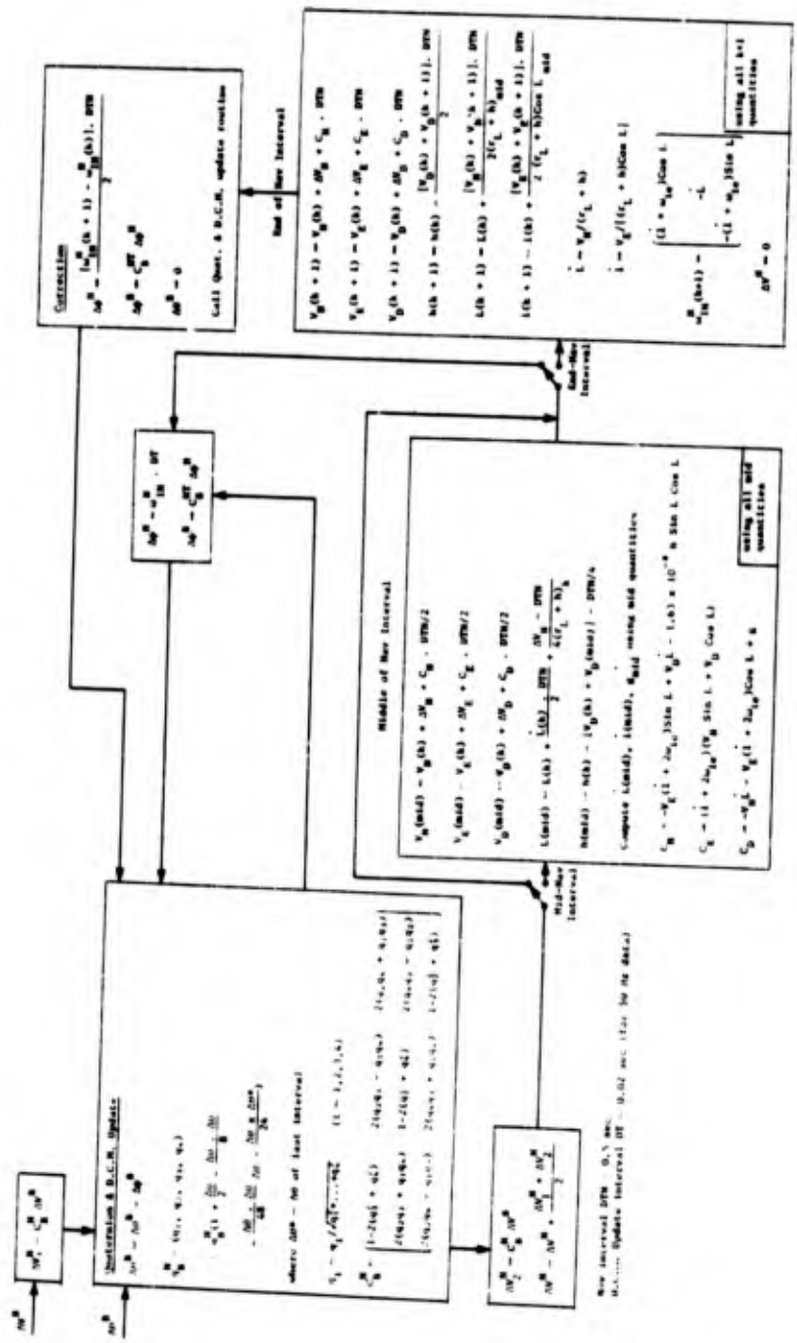


FIGURE 9 - LOCAL-LEVEL NAVIGATOR BLOCK DIAGRAM

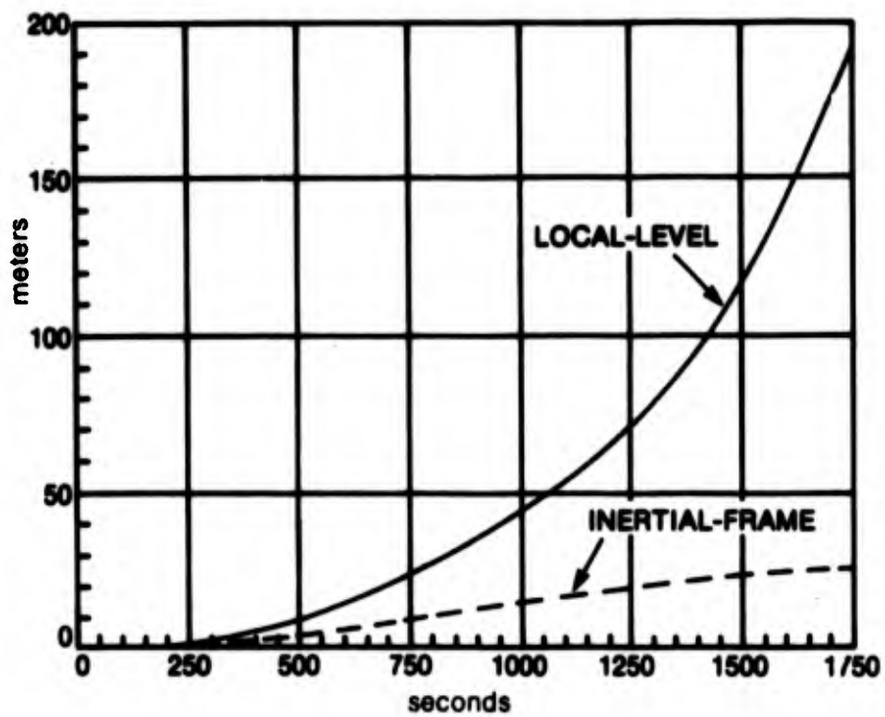


FIGURE 10 - NAVIGATOR'S POSITION ERROR (METERS) VS. TIME (SEC)

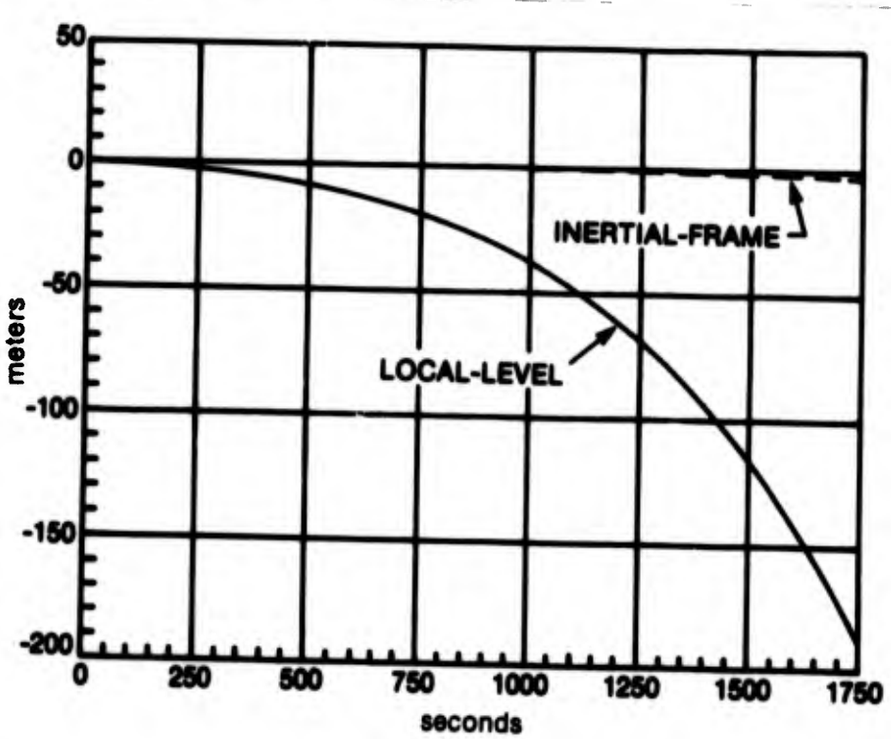


FIGURE 11 - NAVIGATOR'S ALTITUDE ERROR (METERS) VS. TIME (SEC)

APPENDIX I

APPENDIX 1

```
1 -     1.000 C  PROGRAM 'PROFIN' IS DRIVER FEEDING OUTPUTS FROM PROFGEN DIRECTLY
2 -     2.000 C  INTO NAV EQUATIONS, BYPASSING GYRO & ACCELEROMETER MODELS.
3 -     3.000 C
4 -     3.100 C      F:1 PROFGEN DATA FILE
5 -     3.200 C      F:2 PRINTED OUTPUT
6 -     3.300 C      F:3 DISK OUTPUT FILE
7 -     3.400 C
8 -     4.000        COMMON /TIME/T,DT/DEL/DEL(3),DELANG(3),DELANGL(3)
9 -     5.000        DIMENSION BUF(322),ANG(3),ANGL(3),F(3),FL(3)
10 -    6.000        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
11 -    7.000 C
12 -    7.100        OUTPUT (102) 'ENTER 1 FOR PRINTED OUTPUT'
13 -    7.200        OUTPUT (102) '       0 FOR NONE'
14 -    7.300        INPUT (101) NPRNT
15 -    7.400        OUTPUT (102) 'ENTER 1 FOR DISK OUTPUT'
16 -    7.500        OUTPUT (102) '       0 FOR NONE'
17 -    7.600        INPUT (101) NDISK
18 -    7.700 C
19 -    8.000        T=0.0D0;DT=.02D0
20 -    9.000        DELANGL(1)=DELANGL(2)=DELANGL(3)=0.0D0
21 -   10.000        CALL INIT
22 -   11.000 C
23 -   12.000 C  DISPOSE OF 1ST INPUT FILE RECORD:
24 -   13.000        CALL BUFFER IN(1,1,N,1,N1)
25 -   14.000        IF (N1 .NE. 2) WRITE (102,10);STOP
26 -   15.000 C
27 -   16.000        TLAST=-10.0D0
28 -   17.000        FL(1)=FL(2)=FL(3)=0.0D0
29 -   18.000        ANGL(1)=ANGL(2)=ANGL(3)=0.0D0
30 -   19.000 C
31 -   20.000     5  CALL BUFFER IN(1,1,BUF,644,N1)
32 -   21.000        IF (N1 .EQ. 2) GO TO 20
33 -   22.000        WRITE (102,10);END FILE 3;STOP
34 -   23.000    10  FORMAT ('* * * DISK READ ERROR * * *')
35 -   24.000 C
36 -   25.000    20  DO 100 N=0,45
37 -   26.000        T=BUF(1+N*7)
38 -   26.500 C      CHECK FOR BLANK BUFFER ELTS.:
39 -   26.600        IF (T .LT. 1.0D-5 .AND. N .GE. 1) GO TO 5
40 -   27.000 C      CHECK FOR DUPLICATE RECORDS:
41 -   28.000        IF (DABS(T-TLAST) .LT. 1.0D-5)  GO TO 100
42 -   29.000           DO 50 M=1,3
43 -   30.000           ANG(M)=BUF(M+1+N*7)
```

```
 44 -    31.000              F(M)=BUF(M+4+N*7)
 45 -    32.000              DELANG(M)=(ANG(M)+ANGL(M))*0.5*DT
 46 -    33.000              DEL(M)=(F(M)+FL(M))*0.5*DT*.3048
 47 -    34.000              ANGL(M)=ANG(M)
 48 -    35.000     50       FL(M)=F(M)
 49 -    36.000           TLAST=T
 50 -    37.000           IF (T .LT. 1.0D-5)  GO TO 100
 51 -    38.000           CALL INFRAM
 52 -    40.000           CALL GRAV
 53 -    41.000           ITIM=T+1.0D-5
 54 -    42.000           IF (T-ITIM .GT. 1.0D-3) GO TO 100
 55 -    42.500           CALL NQUANT
 56 -    42.600           IF (NPRNT .EQ. 1) CALL OUTP
 57 -    43.000           IF (NDISK .EQ. 1) CALL OUTD
 58 -    44.000   100     CONTINUE
 59 -    45.000 C
 60 -    46.000           GO TO 5
 61 -    47.000           END
```

```
1 -    1.000          SUBROUTINE INIT
2 -    1.500 C        SUBROUTINE INITIALIZES D.C.M. C, QUATERNION Q,
3 -    1.600 C        GRAVITY VECTOR G, INERTIAL POSITION RI, & INERTIAL VELOCITY VI.
4 -    3.000 C
5 -    3.500          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
6 -    4.000          COMMON /C/C(3,3)/Q/Q(4)
7 -    4.100          COMMON /OUTPUT/VI(3),RI(3)/G/G(3)
8 -    4.200          COMMON /LAST/VILAST(3),RILAST(3)
9 -    4.500          COMMON /POS/SLAT,SLONG,ALAT,ALONG,ALT
10 -   4.700          DIMENSION C1(3,3),C2(3,3)
11 -   5.500          PI=DACOS(-1.0D0)
12 -   6.000 C
13 -   7.000          WRITE (102,10)
14 -   8.000    10    FORMAT ('ENTER INITIAL LAT., LONG., HEADING(DEGREES)')
15 -   9.000          INPUT (101) SLAT,SLONG,HEAD
16 -   9.050          SLAT=SLAT*PI/180.0
17 -   9.052          SLONG=SLONG*PI/180.0
18 -   9.060          HEAD=HEAD*PI/180.0
19 -  10.000          WRITE (102,20)
20 -  10.200    20    FORMAT ('ENTER INITIAL ROLL, PITCH (DEGREES)')
21 -  10.400          INPUT (101) ROLL,PITCH
22 -  10.600          ROLL=ROLL*PI/180.D0
23 -  10.700          PITCH=PITCH*PI/180.D0
24 -  11.000          WRITE (102,25)
25 -  11.200    25    FORMAT ('ENTER INITIAL ALTITUDE (FEET)')
26 -  11.400          INPUT (101) ALT
27 -  12.000          WRITE (102,30)
28 -  12.200    30    FORMAT ('ENTER INITIAL INERTIAL VELOCITY COMPONENTS'/
29 -  12.400         $   '    DUE TO VEHICLE MOTION (FT/SEC)')
30 -  12.600          INPUT (101) VX,VY,VZ
31 -  13.000          ALT=ALT*0.3048
32 -  13.200          VX=VX*.3048
33 -  13.300          VY=VY*.3048
34 -  13.400          VZ=VZ*.3048
35 -  18.100 C
36 -  18.130 C   INITIALIZE D.C.M.:
37 -  18.150          CHEAD=DCOS(HEAD)
38 -  18.155          CROLL=DCOS(ROLL)
39 -  18.160          CPITCH=DCOS(PITCH)
40 -  18.165          SHEAD=DSIN(HEAD)
41 -  18.170          SROLL=DSIN(ROLL)
42 -  18.175          SPITCH=DSIN(PITCH)
43 -  18.190          C1(1,1)=CHEAD*CPITCH
```

```
 89 -    23.200          DO 150 JJ=1,3
 90 -    23.400      150 RILAST(JJ)=RI(JJ)
 91 -    26.000 C
 92 -    27.000 C        INITIALIZE G:
 93 -    28.000          CALL GRAV
 94 -    29.000 C
 95 -    30.000 C        INITIALIZE VI:
 96 -    31.000          WIE=.7292115147D-4
 97 -    32.000          V=WIE*DSQRT((RI(1)**2.0)+(RI(2)**2.0))
 98 -    33.000          VILAST(1)=-V*SINLG+VX
 99 -    34.000          VILAST(2)=V*COSLG+VY
100 -    35.000          VILAST(3)=VZ
101 -    36.000          RETURN
102 -    37.000          END
```

```
44 -    18.210          C1(1,2)=CHEAD*SPITCH*SROLL-SHEAD*CROLL
45 -    18.230          C1(1,3)=CHEAD*SPITCH*CROLL+SHEAD*SROLL
46 -    18.250          C1(2,1)=SHEAD*CPITCH
47 -    18.270          C1(2,2)=SHEAD*SPITCH*SROLL+CHEAD*CROLL
48 -    18.290          C1(2,3)=SHEAD*SPITCH*CROLL-CHEAD*SROLL
49 -    18.310          C1(3,1)=-SPITCH
50 -    18.330          C1(3,2)=CPITCH*SROLL
51 -    18.350          C1(3,3)=CPITCH*CROLL
52 -    18.370 C
53 -    18.390          SINL=DSIN(SLAT)
54 -    18.395          COSL=DCOS(SLAT)
55 -    18.410          SINLG=DSIN(SLONG)
56 -    18.415          COSLG=DCOS(SLONG)
57 -    18.430          C2(1,1)=-SINL*COSLG
58 -    18.450          C2(1,2)=-SINLG
59 -    18.470          C2(1,3)=-COSL*COSLG
60 -    18.490          C2(2,1)=-SINL*SINLG
61 -    18.510          C2(2,2)=COSLG
62 -    18.530          C2(2,3)=-COSL*SINLG
63 -    18.550          C2(3,1)=COSL
64 -    18.570          C2(3,2)=0.0D0
65 -    18.590          C2(3,3)=-SINL
66 -    18.610 C
67 -    18.630          DO 100 I=1,3
68 -    18.650             DO 90 J=1,3
69 -    18.670    90       C(I,J)=C2(I,1)*C1(1,J)+C2(I,2)*C1(2,J)+C2(I,3)*C1(3,J)
70 -    18.690   100     CONTINUE
71 -    18.710 C
72 -    18.730 C  INITIALIZE QUATERNION (FROM GRUBIN, JOUR. SPACECRAFT, OCT. 1970):
73 -    18.750          Q(1)=DSQRT(1.0+C(1,1)+C(2,2)+C(3,3))/2.0
74 -    18.770          Q(2)=(C(3,2)-C(2,3))/(4.0*Q(1))
75 -    18.790          Q(3)=(C(1,3)-C(3,1))/(4.0*Q(1))
76 -    18.810          Q(4)=(C(2,1)-C(1,2))/(4.0*Q(1))
77 -    18.830          QL=DSQRT(Q(1)**2+Q(2)**2+Q(3)**2+Q(4)**2)
78 -    18.850          DO 110 J=1,4
79 -    18.870    110 Q(J)=Q(J)/QL
80 -    19.400 C
81 -    19.500 C        INITIALIZE RI:
82 -    20.500          RE=20925640.D0*.3048D0
83 -    20.600          RP=20855481.D0*.3048D0
84 -    21.000          E=(RE*RE-RP*RP)/(RE*RE)
85 -    21.500          RN=RE/DSQRT(1.0-E*SINL**2)
86 -    22.000          RI(1)=(RN+ALT)*COSL*COSLG
87 -    22.500          RI(2)=(RN+ALT)*COSL*SINLG
88 -    23.000          RI(3)=(RN*RP*RP/(RE*RE)+ALT)*SINL
```

```
      ISTEP=ISTEP+1
      IF (ISTEP .LT. 10) GO TO 150
      QL=DSQRT(Q(1)**2+Q(2)**2+Q(3)**2+Q(4)**2)
      DO 130 J=1,4
  130 Q(J)=Q(J)/QL
      ISTEP=0
C
C  UPDATE D.C.M.:
  150 C(1,1)=1.0-2.0*(Q(3)**2+Q(4)**2)
      C(2,1)=2.0*(Q(2)*Q(3)+Q(1)*Q(4))
      C(3,1)=2.0*(Q(2)*Q(4)-Q(1)*Q(3))
      C(1,2)=2.0*(Q(2)*Q(3)-Q(1)*Q(4))
      C(2,2)=1.0-2.0*(Q(2)**2+Q(4)**2)
      C(3,2)=2.0*(Q(4)*Q(3)+Q(1)*Q(2))
      C(1,3)=2.0*(Q(2)*Q(4)+Q(1)*Q(3))
      C(2,3)=2.0*(Q(4)*Q(3)-Q(1)*Q(2))
      C(3,3)=1.0-2.0*(Q(2)**2+Q(3)**2)
C
      DO 155 I=1,3
  155 DANGL(I)=DELANG(I)
C
C  COMPUTE DELVIN'S, AVERAGE & GET VI'S, RI'S:
      DO 160 I=1,3
      DELVIN(I)=C(I,1)*DEL(1)+C(I,2)*DEL(2)+C(I,3)*DEL(3)
      DELVI(I)=(DELVI(I)+DELVIN(I))*0.5-G(I)*DT
      VI(I)=VILAST(I)+DELVI(I)
  160 RI(I)=RILAST(I)+VILAST(I)*DT+0.5*DT*DELVI(I)
C
C      SAVE VI & RI FOR NEXT TIME AROUND:
      DO 170 I=1,3
      VILAST(I)=VI(I)
  170 RILAST(I)=RI(I)
C
      RETURN
      END
```

```
  1 -    1.000          SUBROUTINE INFRAM
  2 -    1.500 C
  3 -    2.000 C       SUBROUTINE CALCULATES INERTIAL VELOCITY & POSITION,
  4 -    3.000 C       AND UPDATES QUATERNION & D.C.M. QUATERNION IS
  5 -    3.500 C       NORMALIZED EVERY 10TH TIME.
  6 -    4.000 C
  7 -    4.100 C       CORIOLIS CORRECTION IS MADE BY AVERAGING TRANSFORMED
  8 -    4.150 C       VELOCITY CHANGES OBTAINED USING D.C.M. AT START & END
  9 -    4.200 C       OF UPDATE INTERVAL. CORRECTION DUE TO BOB JOHNSON, OCT. 1978.
 10 -    4.250 C
 11 -    4.300          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
 12 -    4.400 C
 13 -    4.500          COMMON /C/C(3,3)/Q/Q(4)
 14 -    6.000          COMMON /G/G(3)/TIME/T,DT
 15 -    7.000          COMMON /LAST/VILAST(3),RILAST(3)
 16 -    8.000          COMMON /OUTPUT/VI(3),RI(3)
 17 -    8.500          COMMON /DEL/DEL(3),DELANG(3),DANGL(3)
 18 -    8.600 C
 19 -    9.000          DIMENSION DELVI(3),CHI(4),QQ(4)
 20 -    9.500          DIMENSION DELVIN(3)
 21 -   10.000 C
 22 -   13.000          DATA ISTEP/0/
 23 -   17.000 C
 24 -   29.000 C  COMPUTE DELVI'S BEFORE UPDATING MATRIX:
 25 -   32.000            DO 110 I=1,3
 26 -   33.000            DELVI(I)=C(I,1)*DEL(1)+C(I,2)*DEL(2)+C(I,3)*DEL(3)
 27 -   36.000   110      CONTINUE
 28 -   38.000 C
 29 -   44.000 C  UPDATE QUATERNION:
 30 -   45.000            DOT=DELANG(1)**2+DELANG(2)**2+DELANG(3)**2
 31 -   45.100            CHI(1)=1.0-DOT/8.0
 32 -   45.200            CHI(2)=(0.5-DOT/48.0)*DELANG(1)+
 33 -   45.300           A(DANGL(2)*DELANG(3)-DANGL(3)*DELANG(2))/24.0
 34 -   46.000            CHI(3)=(0.5-DOT/48.0)*DELANG(2)+
 35 -   46.100           A(DANGL(3)*DELANG(1)-DANGL(1)*DELANG(3))/24.0
 36 -   47.000            CHI(4)=(0.5-DOT/48.0)*DELANG(3)+
 37 -   47.100           A(DANGL(1)*DELANG(2)-DANGL(2)*DELANG(1))/24.0
 38 -   48.000            QQ(1)=Q(1)*CHI(1)-Q(2)*CHI(2)-Q(3)*CHI(3)-Q(4)*CHI(4)
 39 -   49.000            QQ(2)=Q(1)*CHI(2)+CHI(1)*Q(2)+Q(3)*CHI(4)-Q(4)*CHI(3)
 40 -   50.000            QQ(3)=Q(1)*CHI(3)+CHI(1)*Q(3)-Q(2)*CHI(4)+Q(4)*CHI(2)
 41 -   51.000            QQ(4)=Q(1)*CHI(4)+CHI(1)*Q(4)+Q(2)*CHI(3)-Q(3)*CHI(2)
 42 -   51.010            DO 128 J=1,4
 43 -   51.020   128      Q(J)=QQ(J)
```

```
      SUBROUTINE NQUANT
C
C  SUBROUTINE COMPUTES CURRENT LAT, LONG, ALTITUDE,
C  AND VELOCITY IN NAVIGATION FRAME.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      COMMON /OUTPUT/VI(3),RI(3)
      COMMON /POS/SLAT,SLONG,ALAT,ALONG,ALT
      COMMON /VNAV/VN(3)/TIME/T,DT
      COMMON /D/ D(3,3)
C
      DIMENSION TT(3)
C
      DATA NPLACE/0/
      IF (NPLACE .EQ. 1) GO TO 5
C
C  SET CONSTANTS:
         RE=20925640.0D0*0.3048D0
         RP=20855481.0D0*0.3048D0
         RE2=RE*RE
         RP2=RP*RP
         PI=DACOS(-1.0D0)
         TER1=RP2-RE2
         TER2=TER1/(RP2)
         WIE=.7292115147D-4
         TER3=-2.D0*TER1
         TER4=TER1*TER2
         TER5=2.D0*TER2
         TER6=-TER2**2
         RERP=RE/RP
         TER8=RE2/RP2
         TER9=RP2/RE2
C
    5    NPLACE=1
         WO=DSQRT(RI(1)**2+RI(2)**2)
         ALAMB=DASIN(RI(2)/WO)
         IF (RI(1) .LT. 0.0D0 .AND. RI(2) .GE. 0.0C0) ALAMB=PI-ALAMB
         IF (RI(1) .LT. 0.0D0 .AND. RI(2) .LT. 0.0C0) ALAMB=-PI-ALAMB
C
C  COMPUTE LONGITUDE:
         ALONG=ALAMB-WIE*T
         IF (ALONG .LE. -PI) ALONG=ALONG+2.0*PI
```

```
  1 -     1.000         SUBROUTINE GRAV
  2 -     2.000 C       SUBROUTINE CALCULATES COMPONENTS OF GRAVITY IN INERTIAL FRAME
  3 -     2.500 C       USING ADVANCED MODEL FROM BRITTING.
  4 -     3.000 C
  5 -     3.500         IMPLICIT DOUBLE PRECISION (A-H,O-Z)
  6 -     3.600 C
  7 -     4.000         COMMON /G/G(3)/OUTPUT/VI(3),RI(3)
  8 -     6.000 C
  9 -     7.000         R=DSQRT((RI(1)**2)+(RI(2)**2)+(RI(3)**2))
 10 -     8.000         U=1.407645D16*.3048D0**3
 11 -     9.000         AJ2=.001082300
 12 -    10.000         RE=20925640.0D0*.3048D0
 13 -    11.000 C
 14 -    12.000         AJ3=-2.3D-6
 15 -    12.500         AJ4=-1.8D-6
 16 -    13.000         R1=DSQRT(RI(1)**2+RI(2)**2)
 17 -    14.000         SINPHI=R1/R
 18 -    14.200         COSPHI=RI(3)/R
 19 -    14.500 C
 20 -    15.000         GRAD=-(U/(R*R))*(1.0-1.5*AJ2*((RE/R)**2)
 21 -    16.000        $    *(3.0*(COSPHI**2)-1.0)-2.0*AJ3*((RE/R)**3)
 22 -    17.000        $    *COSPHI*(5.0*(COSPHI**2)-3.0)
 23 -    18.000        $    -.625*AJ4*((RE/R)**4)*(35.0*(COSPHI**4)
 24 -    19.000        $    -30.0*(COSPHI**2)+3.0))
 25 -    19.500 C
 26 -    20.000         GPHI=3.0*U*RE*RE*SINPHI*COSPHI/(R**4)
 27 -    21.000         GPHI=GPHI*(AJ2+(0.5*AJ3*RE*(5.0*(COSPHI**2)-1.0)
 28 -    22.000        $    /(R*COSPHI))+(5.0/6.0)*AJ4*
 29 -    23.000        $    (RE*RE/(R*R))*(7.0*(COSPHI**2)-3.0))
 30 -    23.500 C
 31 -    24.000         G(1)=-RI(1)*(GRAD+(GPHI*RI(3)/R1))/R
 32 -    25.000         G(2)=-RI(2)*(GRAD+(GPHI*RI(3)/R1))/R
 33 -    26.000         G(3)=RI(3)*((GPHI*R1/RI(3))-GRAD)/R
 34 -    27.000 C
 35 -    28.000         RETURN
 36 -    29.000         END
```

```
      30.400          D(3,3)=-SINLAT
      31.000          TT(1)=VI(1)+WIE*RI(2)
      31.200          TT(2)=VI(2)-WIE*RI(1)
      31.400          TT(3)=VI(3)
      32.000          DO 20 J=1,3
      33.000    20    VN(J)=D(J,1)*TT(1)+D(J,2)*TT(2)+D(J,3)*TT(3)
      34.000 C
      35.000          RETURN
      36.000          END
```


37

```
C
C  * * * * * * * * LATITUDE - ALTITUDE ALGORITHM * * * * * * * *
C                    DUE TO BOB JOHNSON, SEPTEMBER 1978
C
      Z0=RI(3)
      Z02=Z0*Z0
      W02=W0*W0
      X1=RP2*Z02
      X2=TER3*Z0
      X3=TER4-Z02-W02*TER8
      X4=TER5*Z0
      X5=TER6
      Y1=X2
      Y2=2.D0*X3
      Y3=3.D0*X4
      Y4=4.D0*X5
C
      ROOT=DSQRT(RP2/(1.0+TER9*W02/Z02))
      IF (Z0 .LT. 0.0D0) ROOT=-ROOT
C
      DO 50 L=1,3
      F=X1+ROOT*(X2+ROOT*(X3+ROOT*(X4+ROOT*X5)))
      DERIV=Y1+ROOT*(Y2+ROOT*(Y3+ROOT*Y4))
   50 ROOT=ROOT-F/DERIV
C
      W1=RERP*DSQRT(RP2-ROOT**2)
      ALT=DSQRT((W0-W1)**2+(Z0-ROOT)**2)
      IF (W0+DABS(Z0) .LT. W1+DABS(ROOT)) ALT=-ALT
      ALAT=DATAN(TER8*ROOT/W1)
C
C  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C  COMPUTE VELOCITY IN NAV FRAME:
      SINLAT=DSIN(ALAT)
      SINLAM=DSIN(ALAMB)
      COSLAT=DCOS(ALAT)
      COSLAM=DCOS(ALAMB)
      D(1,1)=-SINLAT*COSLAM
      D(1,2)=-SINLAT*SINLAM
      D(1,3)=COSLAT
      D(2,1)=-SINLAM
      D(2,2)=COSLAM
      D(2,3)=0.D0
      D(3,1)=-COSLAM*COSLAT
      D(3,2)=-COSLAT*SINLAM
```

```
  1 -     1.000       SUBROUTINE OUTO
  2 -     2.000 C
  3 -     3.000 C  SUBROUTINE WRITES TO DISK FOR INERTIAL FRAME NAVIGATOR.
  4 -     4.000 C
  5 -     5.000       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
  6 -     6.000 C
  7 -     7.000       COMMON /TIME/T,DT
  8 -     8.000       COMMON /POS/SLAT,SLONG,ALAT,ALONG,ALT
  9 -     9.000       COMMON /VNAV/VN(3)
 10 -    10.000 C
 11 -    11.000       DIMENSION BUF(8)
 12 -    12.000 C
 13 -    12.100       RADDEG=180.DO/DACOS(-1.DO)
 14 -    12.300 C
 15 -    13.000       BUF(1)=T
 16 -    14.000       BUF(2)=ALAT*RADDEG
 17 -    15.000       BUF(3)=ALONG*RADDEG
 18 -    16.000       BUF(4)=ALT
 19 -    17.000       BUF(5)=VN(1)
 20 -    18.000       BUF(6)=VN(2)
 21 -    19.000       BUF(7)=VN(3)
 22 -    20.000       BUF(8)=DSQRT(VN(1)**2 + VN(2)**2 + VN(3)**2)
 23 -    21.000 C
 24 -    22.000       CALL BUFFER OUT(3,1,BUF,16,N1,N2,N3)
 25 -    23.000       IF (N3 .NE. 0) OUTPUT (102) 'DISK WRITE ERROR';STOP
 26 -    24.000 C
 27 -    25.000       RETURN
 28 -    26.000       END
```


39

```
      SUBROUTINE OUTP
C     SUBROUTINE WRITES TO TTY OR LP.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /TIME/T,DT
      COMMON /POS/SLAT,SLONG,ALAT,ALONG,ALT
      COMMON /VNAV/VN(3)
C
      DATA NPLACE/0/
      IF (NPLACE .EQ. 1) GO TO 8
      NPLACE=1
      PI=DACOS(-1.0D0)
      WRITE (2,5)
    5 FORMAT ('TIME',T10,'LAT',T26,'LONG',T42,'ALTITUDE',T58,'VN(1)',
     1T74,'VN(2)',T90,'VN(3)',T106,'VPATH'//)
C
    8 WRITE (2,10) T,ALAT*180.0/PI,ALONG*180.0/PI,ALT,
     1VN(1),VN(2),VN(3),DSQRT(VN(1)**2+VN(2)**2+VN(3)**2)
   10 FORMAT (F6.1,7G16.8)
      RETURN
      END
```

APPENDIX II

APPENDIX 2

```
      C  PROGRAM "ALGER " RUNS 2 ALGORITHMS TO FIND GEODETIC
      C  LATITUDE & ALTITUDE FROM INERTIAL FRAME COORDINATES.
      C  BOTH ALGORITHMS ARE RUN AT 1 ITERATION.
      C
      C  1ST ALGORITHM DUE TO BOB JOHNSON, SEPTEMBER 1978.
      C  2ND ALGORITHM DUE TO BOWRING.
      C
      C  FUNCTION MTIMER RETURNS ELAPSED CPU TIME IN UNITS OF
      C  2 MILLISECONDS.
      C
      C
            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      C
      C  SET CONSTANTS:
            RE=20925640.DO
            RP=20855481.0DO
            RE2=RE*RE;RP2=RP*RP
            RERP=RE/RP
            E=(RE2-RP2)/RE2
            PI=DACOS(-1.0DO)
            TERM1=RP2-RE2
            TERM2=TERM1/RP2
            TERM12=-2.DO*TERM1
            TERM13=TERM1*TERM2
            TERM4=RE2/RP2
            TERM5=4.DO*TERM2
            TERM6=-4.DO*TERM2**2
            TERM7=RE2*RP2
            TERM8=RE2/RP2
            TERM9=RP2/RE2
            EPS=RE2/RP2-1.DO
            CREF1=EPS*RP
            CREF2=E*RE
      C
      C  GET INERTIAL COORDINATES:
            OUTPUT (102) 'ENTER 1 FOR INERTIAL COORDINATES'
            OUTPUT (102) '      2 FOR LATITUDE & HEIGHT'
            INPUT (101) NN
            IF (NN .EQ. 2) GO TO 10
      C
            OUTPUT (102) 'ENTER XO, ZO'
            INPUT (101) XO,ZO
            GO TO 15
```

```
                TANP=(ZO+COEF1*SINU**3)/(RO-COEF2*COSU**3)
                ALAT2=DATAN(TANP)*180.0D0/PI
                COSP=1.0D0/DSQRT(1.0D0+TANP**2)
                HT2=RC/COSP-RE/DSQRT(1.0D0-E*(1.0D0-COSP**2))
  200 CONTINUE
                KK=MTIMER(1)
                WRITE (102,210) HT2,ALAT2,KK
  210 FORMAT (/'BOWRING ALGORITHM:'/'ALTITUDE =',G22.15/
     $   'LATITUDE =',G22.15/'CPU TIME =',I5)
C
                STOP
                END
```

43

```
      C
   10 WRITE (102,160) HT1,ALAT1,KK
  160 FORMAT (/'NEW ALGORITHM:'/'ALTITUDE =',G22.15/
     $   'LATITUDE =',G22.15/'CPU TIME =',I5)
      C
C  PERFORM 2ND ALGORITHM 1000 TIMES:
      KK=KTIMER(0)
      DO 200 J=1,1000
      TANU=RERP*Z0/W0
      C
      COSU=1.0D0/DSQRT(1.0D0+TANU**2)
      SINU=DSQRT(1.0D0-COSU**2)
      IF (Z0 .LT. 0.0D0) SINU=-SINU
```

```
      RN=RE/DSQRT(1.0D0-E*DSIN(RLAT)**2)
      RC=(RN+H)*DCOS(RLAT)
      ZC=(RN*RP*RP/(RE*RE)+H)*DSIN(RLAT)
C
C  PERFORM 1ST ALGORITHM 1000 TIMES:
   15 KK=LTIMER(0)
      DO 150 J=1,1000
      ZC2=ZC*ZC;WC2=WC*WC
      X1=RP2*ZC2
      X2=TER3*ZC
      X3=TER4*ZC2-WC2*TER8
      X4=TER5*ZC
      X5=TER6
      Y1=X2;Y2=2.0*X3;Y3=3.0*X4;Y4=4.0*X5
C
      RGGT=DSQRT(RP2/(1.0+TER9*WC2/ZC2))
      IF (ZC .LT. 0.0D0) RGGT=-RGGT
C
      DO 50 L=1,2
      F=X1+RGGT*(X2+RGGT*(X3+RGGT*(X4+RGGT*X5)))
      DERIV=Y1+RGGT*(Y2+RGGT*(Y3+RGGT*Y4))
   50 RGGT=RGGT-F/DERIV
C
      A1=PERP*DSQRT(RP2-RGGT**2)
      HT1=DSQRT((WC-W1)**2+(ZC-RGGT)**2)
      IF (WC+DABS(ZC) .LT. A1+DABS(RGGT)) HT1=-HT1
      ALAT1=DATAN(TER8*RGGT/A1)*180.0/PI
  150 CONTINUE
      KK=LTIMER(1)
      WRITE (102,160) HT1,ALAT1,KK
  160 FORMAT (//'NEW ALGORITHM:'//'ALTITUDE =',G22.15/
     $   'LATITUDE =',G22.15/'CPU TIME =',I5)
C
C  PERFORM 2ND ALGORITHM 1000 TIMES:
      KK=LTIMER(0)
      DO 220 J=1,1000
      TANU=PERP*ZC/WC
C
      DO 180 K=1,2
      COSU=1.0D0/DSQRT(1.0D0+TANU**2)
      SINU=DSQRT(1.0D0-COSU**2)
      IF (ZC .LT. 0.0D0) SINU=-SINU
      TANP=(ZC+CHEF1*SINU**3)/(WC-CHEF2*COSU**3)
  180 TANU=PRR*TANP
```

```
      C  PROGRAM "ALGOR2" RUNS 2 ALGORITHMS TO FIND GEODETIC
      C  LATITUDE & ALTITUDE FROM INERTIAL FRAME COORDINATES.
      C  BOTH ALGORITHMS ARE RUN AT 2 ITERATIONS.
      C
      C  1ST ALGORITHM DUE TO BOB JOHNSON, SEPTEMBER 1978.
      C  2ND ALGORITHM DUE TO BOWRING.
      C
      C  FUNCTION NTIMER RETURNS ELAPSED CPU TIME IN UNITS OF
      C  2 MILLISECONDS.
      C
      C
          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      C
      C  SET CONSTANTS:
          RE=20925640.0D0
          RP=20855481.0D0
          RE2=RE*RE;RP2=RP*RP
          RERP=RE/RP
          E=(RE2-RP2)/RE2
          PI=DACOS(-1.0D0)
          TER1=RP2-RE2
          TER2=TER1/RP2
          TER3=-2.0*TER1;TER4=TER1*TER2
          TER5=2.0*TER2;TER6=-TER2**2
          TER7=RE2*RP2;TER8=RE2/RP2;TER9=RP2/RE2
          EPS=RE2/RP2-1.0D0
          COEF1=EPS*RP;COEF2=E*RE
          RPRE=RP/RE
      C
      C  GET INERTIAL COORDINATES:
          OUTPUT (102) 'ENTER 1 FOR INERTIAL COORDINATES'
          OUTPUT (102) '      2 FOR LATITUDE & HEIGHT'
          INPUT (101) NN
          IF (NN .EQ. 2) GO TO 10
      C
          OUTPUT (102) 'ENTER XO, ZO'
          INPUT (101) XO,ZO
          GO TO 15
      C
   10     OUTPUT (102) 'ENTER LAT, HEIGHT'
          INPUT (101) PLAT,H
          PLAT=PLAT*PI/180.0D0
      C  COMP TO INERTIAL COORDINATES USING EXACT FORMULAE:
```

```
      ALAT2=DATAN(TANP)*180.0D0/PI
      COSP=1.0D0/DSQRT(1.0D0+TANP**2)
      HT2=WO/COSP-RE/DSQRT(1.0D0-E*(1.0D0-COSP**2))
  200 CONTINUE
      KK=NTIMER(1)
      WRITE (102,210) HT2,ALAT2,KK
  210 FORMAT (//'BOWRING ALGORITHM:'/'ALTITUDE =',G22.15/
     $   'LATITUDE =',G22.15/'CPU TIME =',I5)
C
      STOP
      END
```

p. 48 Blank

APPENDIX III

```
C     PROGRAM "LOCLEV" IS LOCAL LEVEL NAVIGATOR.
C     OUTPUTS FROM PROFGEN GO DIRECTLY INTO NAV
C     EQUATIONS, BYPASSING GYRO & ACCEL. MODELS.
C
      COMMON /DELV/DELVN,DELVE,DELVD/C/C(3,3)
      COMMON /ANGLES/DELANG(3),DELPSI(3)/TIME/T,DT,DTN
      COMMON /ISTEP/ISTEP
      COMMON /RATES/DLAT,DLONG,VN,VE,VD,WIN(3)
      DIMENSION BUF(322),ANG(3),ANGL(3),F(3),FL(3)
      DIMENSION DEL(3),DELPSIN(3)
      DIMENSION DV1(3),DV2(3)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C     INITIALIZATION:
      OUTPUT (102) 'ENTER 1 FOR HARDCOPY OUTPUT, ELSE 0'
      INPUT (101) NFLAGH
      OUTPUT (102) 'ENTER 1 FOR DISK OUTPUT, ELSE 0'
      INPUT (101) NFLAGD
      TSTART=0.D0
      DTN=0.5D0
      DT=.02D0
      NFIN=DTN/DT+.00001D0;MIDPT=NFIN/2
      NCOUNT=0
      ISTEP=0
      DELVN=DELVE=DELVD=0.0D0
      CALL INIT
      DO 4 J=1,3
    4 DELPSIN(J)=WIN(J)*DT
C
C     DISPOSE OF 1ST DTODTA RECORD:
      CALL BUFFER IN(1,1,N,1,N1)
      IF (N1 .NE. 2) WRITE (102,10);STOP
C
      TLAST=-10.D0
      FL(1)=FL(2)=FL(3)=0.D0
      ANGL(1)=ANGL(2)=ANGL(3)=0.D0
C
    5 CALL BUFFER IN(1,1,BUF,644,N1)
      IF (N1 .EQ. 2) GO TO 20
      WRITE (102,10);END FILE 3;STOP
   10 FORMAT ('* * * DISK READ ERROR * * *')
C
```

```
89 -    66.900          IF (NFLAGD .EQ. 1) CALL LOUTD
90 -    67.000  100     CONTINUE
91 -    68.000 C
92 -    69.000          GO TO 5
93 -    70.000          END
```

```
44 -    35.000    20  DO 100 N=0,45
45 -    36.000        T=BUF(1+N*7)
46 -    36.500 C      CHECK FOR BLANK BUFFER ELTS.:
47 -    36.600        IF (T .LT. 1.D-5  .AND. N .GE. 1) GO TO 5
48 -    37.000 C      CHECK FOR DUPLICATE RECORDS:
49 -    38.000        IF (DABS(T-TLAST) .LT. 1.D-5)   GO TO 100
50 -    39.000           DO 50 M=1,3
51 -    40.000           ANG(M)=BUF(M+1+N*7)
52 -    41.000           F(M)=BUF(M+4+N*7)
53 -    42.000           DELANG(M)=(ANG(M)+ANGL(M))*0.5*DT
54 -    43.000           DEL(M)=(F(M)+FL(M))*0.5*DT*.3048D0
55 -    44.000           ANGL(M)=ANG(M)
56 -    45.000    50     FL(M)=F(M)
57 -    46.000        TLAST=T
58 -    47.000        IF (T .LT. TSTART+1.D-5)  GO TO 100
59 -    47.100        DO 55 JJ=1,3
60 -    47.200    55  DV1(JJ)=C(JJ,1)*DEL(1)+C(JJ,2)*DEL(2)+C(JJ,3)*DEL(3)
61 -    48.000        CALL UPDATE
62 -    48.100        DO 60 JJ=1,3
63 -    48.200    60  DV2(JJ)=C(JJ,1)*DEL(1)+C(JJ,2)*DEL(2)+C(JJ,3)*DEL(3)
64 -    49.000        DELVN=DELVN+0.5D0*(DV1(1)+DV2(1))
65 -    50.000        DELVE=DELVE+0.5D0*(DV1(2)+DV2(2))
66 -    51.000        DELVD=DELVD+0.5D0*(DV1(3)+DV2(3))
67 -    51.500        NCOUNT=NCOUNT+1
68 -    51.600 C
69 -    52.000 C      CHECK FOR MIDPOINT OF NAVIGATION INTERVAL:
70 -    53.000        IF (NCOUNT .NE. MIDPT) GO TO 70
71 -    55.000        CALL MID
72 -    55.200 C
73 -    55.500 C      CHECK FOR END OF NAVIGATION INTERVAL:
74 -    56.000    70  IF (NCOUNT .NE. NFIN) GO TO 90
75 -    58.000        CALL NAV
76 -    59.000        CALL ANGCAL
77 -    61.000        DELVN=DELVE=DELVD=0.D0
78 -    61.500        NCOUNT=0
79 -    63.000        DO 85 J=1,3
80 -    64.000    85  DELPSIN(J)=WIN(J)*DT
81 -    64.500 C
82 -    65.000    90  DO 97 J=1,3
83 -    66.000    97  DELPSI(J)=C(1,J)*DELPSIN(1)+C(2,J)*DELPSIN(2)+C(3,J)*DELPSIN(3)
84 -    66.200 C
85 -    66.500 C      CHECK FOR OUTPUT TIME:
86 -    66.600        ITIM=T+1.D-5
87 -    66.700        IF (T-ITIM .GT. 1.D-4) GO TO 100
88 -    66.800        IF (NFLAGH .EQ. 1) CALL LOUT
```

```
44 -    21.800          VD=VD*.3048D0
45 -    22.000 C
46 -    23.000 C        INITIALIZE D.C.M.:
47 -    24.000          CHEAD=DCOS(HEAD)
48 -    24.200          CROLL=DCOS(ROLL)
49 -    24.400          CPITCH=DCOS(PITCH)
50 -    24.600          SHEAD=DSIN(HEAD)
51 -    24.800          SROLL=DSIN(ROLL)
52 -    25.000          SPITCH=DSIN(PITCH)
53 -    26.000          C(1,1)=CHEAD*CPITCH
54 -    27.000          C(1,2)=CHEAD*SPITCH*SROLL-SHEAD*CROLL
55 -    28.000          C(1,3)=CHEAD*SPITCH*CROLL+SHEAD*SROLL
56 -    29.000          C(2,1)=SHEAD*CPITCH
57 -    30.000          C(2,2)=SHEAD*SPITCH*SROLL+CHEAD*CROLL
58 -    31.000          C(2,3)=SHEAD*SPITCH*CROLL-CHEAD*SROLL
59 -    32.000          C(3,1)=-SPITCH
60 -    33.000          C(3,2)=CPITCH*SROLL
61 -    34.000          C(3,3)=CPITCH*CROLL
62 -    35.000 C
63 -    36.000 C        INITIALIZE QUATERNION:
64 -    37.000          Q(1)=DSQRT(1.0+C(1,1)+C(2,2)+C(3,3))/2.0
65 -    38.000          Q(2)=(C(3,2)-C(2,3))/(4.0*Q(1))
66 -    39.000          Q(3)=(C(1,3)-C(3,1))/(4.0*Q(1))
67 -    40.000          Q(4)=(C(2,1)-C(1,2))/(4.0*Q(1))
68 -    44.000          QL=DSQRT(Q(1)**2+Q(2)**2+Q(3)**2+Q(4)**2)
69 -    45.000          DO 50 J=1,4
70 -    46.000    50    Q(J)=Q(J)/QL
71 -    47.000 C
72 -    48.000 C        INITIALIZE REST:
73 -    60.000          COS2L=DCOS(2.0*ALAT)
74 -    60.500          COSL=DCOS(ALAT)
75 -    60.600          SINL=DSIN(ALAT)
76 -    61.000          R0=RE*(1.0-0.5*E*(1.0-COS2L))
77 -    61.500          T1=DSQRT(1.0-ECCEN*SINL*SINL)
78 -    62.000          RADLAT=RE*(1.0-ECCEN)/(T1**3)+ALT
79 -    62.500          RADLNG=RE/T1
80 -    62.600          RADLNG=(RADLNG+ALT)*COSL
81 -    63.000          GLAT=ALAT-E*(1.0-ALT/R0)*DSIN(2.0*ALAT)
82 -    64.000          R=R0+ALT
83 -    65.000          G=(U/(R*R))*(1.0-0.75*AJ2*(1.0-3.0*COS2L))
84 -    66.000         A=R*WIE*WIE*COSL*DCOS(GLAT)
85 -    67.000 C
86 -    67.500          DLAT=VN/RADLAT
87 -    67.600          DLONG=VE/RADLNG
88 -    68.000          CN=1.63D-8*ALT*SINL*COSL
```

```
  1 -     1.000          SUBROUTINE INIT
  2 -     2.000 C        SUBROUTINE INITIALIZES D.C.M., QUATERNION
  3 -     3.000 C        AND OTHER VARIABLES.
  4 -     4.000 C
  5 -     4.200          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
  6 -     4.400 C
  7 -     5.000          COMMON /TIME/T,DT,DTN/C/C(3,3)/Q/Q(4)
  8 -     6.000          COMMON /POS/ALAT,ALONG,ALT/RATES/DLAT,DLONG,VN,VE,VD,WIN(3)
  9 -     7.000          COMMON /ANGLES/DELANG(3),DELPSI(3)
 10 -     8.000          COMMON /ORIENT/HEAD,ROLL,PITCH
 11 -     8.100          COMMON /LAST/DANGL(3)/COEF/CN,CE,CD
 12 -     8.200          COMMON /RAD/RADLAT,RADLTM,RADLNG,RADLGM
 13 -     8.300 C
 14 -     8.500          DIMENSION DPSIN(3)
 15 -     9.000 C
 16 -    11.000          PI=DACOS(-1.D0)
 17 -    11.100          WIE=.7292115147D-4
 18 -    11.500          RE=20925640.0D0*.3048D0
 19 -    11.600          U=1.407645D16*.3048D0**3
 20 -    11.650          AJ2=1.0823D-3
 21 -    11.700          RP=20855481.0D0*.3048D0
 22 -    11.800          ECCEN=(RE*RE-RP*RP)/(RE*RE)
 23 -    11.900          E=(RE-RP)/RE
 24 -    12.000 C
 25 -    13.000          WRITE (102,10)
 26 -    14.000    10    FORMAT ('ENTER INITIAL LAT., LONG., ALTITUDE')
 27 -    15.000          INPUT (101) ALAT,ALONG,ALT
 28 -    16.000          WRITE (102,20)
 29 -    17.000    20    FORMAT ('ENTER INITIAL HEADING,ROLL,PITCH (DEGREES)')
 30 -    18.000          INPUT (101) HEAD,ROLL,PITCH
 31 -    18.100          WRITE (102,30)
 32 -    18.200    30    FORMAT ('ENTER VN, VE, VD (FT/SEC)')
 33 -    18.300          INPUT (101) VN,VE,VD
 34 -    19.000 C
 35 -    19.500 C        CONVERT UNITS:
 36 -    20.000          ALAT=ALAT*PI/180.D0
 37 -    20.200          ALONG=ALONG*PI/180.D0
 38 -    21.000          HEAD=HEAD*PI/180.D0
 39 -    21.100          ROLL=ROLL*PI/180.D0
 40 -    21.200          PITCH=PITCH*PI/180.D0
 41 -    21.500          ALT=ALT*.3048D0
 42 -    21.600          VN=VN*.3048D0
 43 -    21.700          VE=VE*.3048D0
```

```
      SUBROUTINE UPDATE
C     SUBROUTINE UPDATES QUATERNION & D.C.M.
C     USING DELANG'S & DELPSI'S.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      COMMON /Q/Q(4)/C/C(3,3)
      COMMON /ANGLES/DELANG(3),DELPSI(3)/LAST/DANGL(3)
      COMMON /ISTEP/ISTEP
C
      DIMENSION CHI(4),QQ(4)
C
      DO 20 K=1,3
   20 DELANG(K)=DELANG(K)-DELPSI(K)
C
C     UPDATE QUATERNION:
      DOT=DELANG(1)**2+DELANG(2)**2+DELANG(3)**2
      CHI(1)=1.0D0-DOT/8.0D0
      CHI(2)=(0.5D0-DOT/48.0D0)*DELANG(1)+
     A(DANGL(2)*DELANG(3)-DANGL(3)*DELANG(2))/24.CDC
      CHI(3)=(0.5D0-DOT/48.0D0)*DELANG(2)+
     A(DANGL(3)*DELANG(1)-DANGL(1)*DELANG(3))/24.CDC
      CHI(4)=(0.5D0-DOT/48.0D0)*DELANG(3)+
     A(DANGL(1)*DELANG(2)-DANGL(2)*DELANG(1))/24.CDC
      QQ(1)=Q(1)*CHI(1)-Q(2)*CHI(2)-Q(3)*CHI(3)-Q(4)*CHI(4)
      QQ(2)=Q(1)*CHI(2)+CHI(1)*Q(2)+Q(3)*CHI(4)-Q(4)*CHI(3)
      QQ(3)=Q(1)*CHI(3)+CHI(1)*Q(3)-Q(2)*CHI(4)+Q(4)*CHI(2)
      QQ(4)=Q(1)*CHI(4)+CHI(1)*Q(4)+Q(2)*CHI(3)-Q(3)*CHI(2)
      DO 50 J=1,4
   50 Q(J)=QQ(J)
      ISTEP=ISTEP+1
      IF (ISTEP .LT. 10) GO TO 150
      QL=DSQRT(Q(1)**2+Q(2)**2+Q(3)**2+Q(4)**2)
      DO 130 J=1,4
  130 Q(J)=Q(J)/QL
      ISTEP=0
C
C     UPDATE D.C.M.:
  150 C(1,1)=1.0D0-2.0D0*(Q(3)**2+Q(4)**2)
      C(2,1)=2.0D0*(Q(2)*Q(3)+Q(1)*Q(4))
      C(3,1)=2.0D0*(Q(2)*Q(4)-Q(1)*Q(3))
      C(1,2)=2.0D0*(Q(2)*Q(3)-Q(1)*Q(4))
      C(2,2)=1.0D0-2.0D0*(Q(2)**2+Q(4)**2)
```

```
 89 -    68.500          CN=CN-VE*(DLONG+2.0*WIE)*SINL+VD*DLAT
 90 -    69.000          CE=(DLONG+2.0*WIE)*(VN*SINL+VD*COSL)
 91 -    70.000          CD=-VN*DLAT-VE*(DLONG+2.0*WIE)*COSL+G
 92 -    71.000 C
 93 -    72.000          WIN(1)=(DLONG+WIE)*COSL
 94 -    73.000          WIN(2)=-DLAT
 95 -    74.000          WIN(3)=-(DLONG+WIE)*SINL
 96 -    75.000 C
 97 -    76.000          DO 80 I=1,3
 98 -    77.000    80    DPSIN(I)=WIN(I)*DT
 99 -    78.000 C
100 -    79.000          DO 90 I=1,3
101 -    80.000          DELPSI(I)=C(1,I)*DPSIN(1)+C(2,I)*DPSIN(2)+C(3,I)*DPSIN(3)
102 -    81.000    90    DANGL(I)=DELPSI(I)
103 -    82.000 C
104 -    83.000          RETURN
105 -    84.000          END
```

```
  1 -     1.000         SUBROUTINE MID
  2 -     2.000 C       SUBROUTINE CALCULATES TERMS CN, CE, CD
  3 -     3.000 C       AT MIDPOINT OF NAVIGATION INTERVAL.
  4 -     4.000 C
  5 -     4.200         IMPLICIT DOUBLE PRECISION (A-H,O-Z)
  6 -     4.400 C
  7 -     5.000         COMMON /COEF/CN,CE,CD/TIME/T,DT,DTN
  8 -     6.000         COMMON /POS/ALAT,ALONG,ALT/RATES/DLAT,DLONG,VN,VE,VD,WIN(3)
  9 -     7.000         COMMON /DELV/DELVN,DELVE,DELVD
 10 -     8.000         COMMON /RAD/RADLAT,RADLTM,RADLNG,RADLGM
 11 -     9.000 C
 12 -    12.000         RE=20925640.0D0*.3048D0
 13 -    12.500         RP=20855481.0D0*.3048D0
 14 -    12.600         ECCEN=(RE*RE-RP*RP)/(RE*RE)
 15 -    12.700         E=(RE-RP)/RE
 16 -    13.000         U=1.407645D16*.3048D0**3
 17 -    14.000         AJ2=1.0823D-3
 18 -    14.200         WIE=.7292115147D-4
 19 -    15.000 C
 20 -    16.000         VNM=VN+DELVN+CN*DTN/2.0
 21 -    17.000         VEM=VE+DELVE+CE*DTN/2.0
 22 -    18.000         VDM=VD+DELVD+CD*DTN/2.0
 23 -    19.000         ALATM=ALAT+DLAT*DTN/2.0+DELVN*DTN/(4.0*RADLAT)
 24 -    20.000 C
 25 -    21.000         COS2L=DCOS(2.0*ALATM)
 26 -    22.000         COSL=DCOS(ALATM)
 27 -    22.200         SINL=DSIN(ALATM)
 28 -    23.000         RO=RE*(1.0-0.5*E*(1.0-COS2L))
 29 -    23.500         T1=DSQRT(1.0-ECCEN*SINL*SINL)
 30 -    24.000         RADLTM=RE*(1.0-ECCEN)/(T1**3)
 31 -    25.000         RADLGM=RE/T1
 32 -    26.000         ALTM=ALT-(VD+VDM)*DTN*0.25
 33 -    27.000         GLATM=ALATM-E*(1.0-ALTM/RO)*DSIN(2.0*ALATM)
 34 -    28.000         R=RO+ALTM
 35 -    29.000         RADLTM=RADLTM+ALTM
 36 -    30.000         RADLGM=(RADLGM+ALTM)*COSL
 37 -    31.000         DLAT=VNM/RADLTM
 38 -    32.000         DLONG=VEM/RADLGM
 39 -    33.000 C
 40 -    34.000 C       EVALUATE GRAVITY:
 41 -    35.000         G=(U/(R*R))*(1.0-0.75*AJ2*(1.0-3.0*COS2L))
 42 -    36.000        A=R*WIE*WIE*COSL*DCOS(GLATM)
 43 -    37.000 C
```

```
44 -    37.000          C(3,2)=2.0D0*(Q(4)*Q(3)+Q(1)*Q(2))
45 -    38.000          C(1,3)=2.0D0*(Q(2)*Q(4)+Q(1)*Q(3))
46 -    39.000          C(2,3)=2.0D0*(Q(4)*Q(3)-Q(1)*Q(2))
47 -    40.000          C(3,3)=1.0D0-2.0D0*(Q(2)**2+Q(3)**2)
48 -    41.000 C
49 -    42.000          DO 160 J=1,3
50 -    42.500   160    DANGL(J)=DELANG(J)
51 -    43.000 C
52 -    44.000          RETURN
53 -    45.000          END
```

```
 1 -     1.000         SUBROUTINE NAV
 2 -     2.000 C       SUBROUTINE CALCULATES NAVIGATION QUANTITIES OF
 3 -     3.000 C       VN, VE, VD, LATITUDE, LONGITUDE, & ALTITUDE.
 4 -     4.000 C
 5 -     4.500         IMPLICIT DOUBLE PRECISION (A-H,O-Z)
 6 -     5.000         COMMON /RATES/DLAT,DLONG,VN,VE,VD,WIN(3)
 7 -     6.000         COMMON /POS/ALAT,ALONG,ALT/COEF/CN,CE,CD
 8 -     7.000         COMMON /TIME/T,DT,DTN/DELV/DELVN,DELVE,DELVD
 9 -     8.000         COMMON /RAD/RADLAT,RADLTM,RADLNG,RADLGM
10 -     9.000 C
11 -    10.500         PI=DACOS(-1.DO)
12 -    11.000 C
13 -    12.000 C       CALCULATE NEW VELOCITY COMPONENTS:
14 -    13.000         VNNEW=VN+DELVN+CN*DTN
15 -    14.000         VENEW=VE+DELVE+CE*DTN
16 -    15.000         VDNEW=VD+DELVD+CD*DTN
17 -    16.000 C
18 -    17.000 C       CALCULATE POSITION:
19 -    17.500         ALT=ALT-(VD+VDNEW)*DTN*0.5
20 -    18.000         ALAT=ALAT+(VN+VNNEW)*DTN*0.5/RADLTM
21 -    19.000         ALONG=ALONG+(VE+VENEW)*DTN*0.5/RADLGM
22 -    19.100 C
23 -    19.200 C       CORRECT FOR INTERNATIONAL DATE LINE:
24 -    19.500         IF (ALONG .LE. -PI) ALONG=ALONG+2.0*PI
25 -    19.700         IF (ALONG .GT. PI) ALONG=ALONG-2.0*PI
26 -    21.000 C
27 -    22.000         VN=VNNEW
28 -    22.200         VE=VENEW
29 -    22.400         VD=VDNEW
30 -    23.000 C
31 -    24.000         RETURN
32 -    25.000         END
```

```
      38.000 C      EVALUATE CN, CE, CD AT MIDPOINT:
      39.000        CN=-VEM*(DLONG+2.0*WIE)*SINL+VDM*DLAT-1.63D-8*ALTM*SINL*COSL
      40.000        CE=(DLONG+2.0*WIE)*(VNM*SINL+VDM*COSL)
      41.000        CD=-VNM*DLAT-VEM*(DLONG+2.0*WIE)*COSL+G
      42.000 C
      43.000        RETURN
      44.000        END
```

```
  1 -     1.000          SUBROUTINE ANGCAL
  2 -     2.000 C        SUBROUTINE CALCULATES WIN & MAKES ANGLE CORRECTION
  3 -     3.000 C        AT EACH NAVIGATION INTERVAL.
  4 -     4.000 C
  5 -     4.200          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
  6 -     4.400 C
  7 -     5.000          COMMON /RATES/DLAT,DLONG,VN,VE,VD,WIN(3)
  8 -     6.000          COMMON /POS/ALAT,ALONG,ALT/TIME/T,DT,DTN
  9 -     7.000          COMMON /ANGLES/DELANG(3),DELPSI(3)
 10 -     8.000          COMMON /RAD/RADLAT,RADLTM,RADLNG,RADLGM
 11 -     9.000          COMMON /C/C(3,3)
 12 -     9.100 C
 13 -     9.500          DIMENSION DPSIN(3),WINNEW(3)
 14 -    10.000 C
 15 -    13.000          RE=20925640.0D0*.3048D0
 16 -    13.500          RP=20855481.0D0*.3048D0
 17 -    13.600          ECCEN=(RE*RE-RP*RP)/(RE*RE)
 18 -    14.000          WIE=.7292115147D-4
 19 -    15.000 C
 20 -    16.000          COSL=DCOS(ALAT)
 21 -    16.500          SINL=DSIN(ALAT)
 22 -    17.500          T1=DSQRT(1.0-ECCEN*SINL*SINL)
 23 -    18.000          RADLAT=RE*(1.0-ECCEN)/(T1**3)+ALT
 24 -    19.000          RADLNG=RE*T1
 25 -    20.000          RADLNG=(RADLNG+ALT)*COSL
 26 -    21.000 C
 27 -    22.000 C        CALCULATE RATES:
 28 -    23.000          DLAT=VN/RADLAT
 29 -    24.000          DLONG=VE/RADLNG
 30 -    25.000          WINNEW(1)=(DLONG+WIE)*COSL
 31 -    26.000          WINNEW(2)=-DLAT
 32 -    27.000          WINNEW(3)=-(DLONG+WIE)*SINL
 33 -    28.000 C
 34 -    29.000 C        MAKE CORRECTION IN ANGLE CHANGE OVER NAVIGATION INTERVAL:
 35 -    30.000          DO 50 I=1,3
 36 -    31.000    50    DPSIN(I)=0.5*(WINNEW(I)-WIN(I))*DTN
 37 -    32.000          DO 60 I=1,3
 38 -    33.000          DELPSI(I)=C(1,I)*DPSIN(1)+C(2,I)*DPSIN(2)+C(3,I)*DPSIN(3)
 39 -    34.000    60    DELANG(I)=0.D0
 40 -    35.000          CALL UPDATE
 41 -    36.000 C
 42 -    37.000          DO 70 I=1,3
 43 -    38.000    70    WIN(I)=WINNEW(I)
```

```
44 -    39.000 C
45 -    40.000          RETURN
46 -    41.000          END
```


62

```
  1 -     1.000          SUBROUTINE LOUTD
  2 -     1.500 C
  3 -     2.000 C        SUBROUTINE WRITES NAV OUTPUTS ONTO DISK
  4 -     2.500 C        FOR LOCAL LEVEL NAVIGATOR.
  5 -     3.000 C
  6 -     3.100          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
  7 -     3.200 C
  8 -     4.000          COMMON /TIME/T,DT,DTN
  9 -     5.000          COMMON /POS/ALAT,ALONG,ALT
 10 -     6.000          COMMON /RATES/DLAT,DLONG,VN,VE,VD,WIN(3)
 11 -     6.500 C
 12 -     7.000          DIMENSION BUF(8)
 13 -     8.000 C
 14 -     9.500          RADDEG=180.D0/DACOS(-1.D0)
 15 -    10.000 C
 16 -    14.000          BUF(1)=T
 17 -    15.000          BUF(2)=ALAT*RADDEG;BUF(3)=ALONG*RADDEG
 18 -    15.100          BUF(4)=ALT
 19 -    16.000          BUF(5)=VN;BUF(6)=VE;BUF(7)=VD
 20 -    17.000          BUF(8)=DSQRT(VN**2+VE**2+VD**2)
 21 -    20.000          CALL BUFFER OUT(2,1,BUF,16,N1,N2,N3)
 22 -    21.000          IF (N3 .NE. 0) WRITE (102,30);STOP
 23 -    23.000    30    FORMAT ('* * * DISK WRITE ERROR * * *')
 24 -    23.200 C
 25 -    23.500          RETURN
 26 -    24.000          END
```

```
 1 -     1.000         SUBROUTINE LOUT
 2 -     2.000 C       SUBROUTINE WRITES TO TTY OR LP.
 3 -     3.000 C
 4 -     3.500         IMPLICIT DOUBLE PRECISION (A-H,O-Z)
 5 -     4.000         COMMON /TIME/T,DT,DTN
 6 -     5.000         COMMON /POS/ALAT,ALONG,ALT
 7 -     6.000         COMMON /RATES/DLAT,DLONG,VN,VE,VD,WIN(3)
 8 -     7.000 C
 9 -     9.000         PI=DACOS(-1.DO)
10 -     9.500         DATA NPLACE/0/
11 -    10.000 C
12 -    11.000         IF (NPLACE .EQ. 1) GO TO 8
13 -    12.000         WRITE (3,5)
14 -    13.000     5   FORMAT ('TIME',T10,'LAT',T26,'LONG',T42,'ALTITUDE',T58,'VN',
15 -    14.000        1T74,'VE',T90,'VD',T106,'VPATH'//)
16 -    15.000     8   WRITE (3,10) T,ALAT*180.0/PI,ALONG*180.0/PI,ALT,
17 -    16.000        1VN,VE,VD,DSQRT(VN**2+VE**2+VD**2)
18 -    17.000    10   FORMAT (F6.1,7G16.8)
19 -    17.500         NPLACE=1
20 -    18.000         RETURN
21 -    19.000         END
```