TE-43

COMPUTATIONAL COMPARISON OF STRAPDOWN SYSTEM ATTITUDE ALGORITHMS

by

Fred J. Marcus

June 1971

Approved: C. Marbey

Director
Measurement Systems Laboratory

Massachusetts Institute of Technology
Measurement Systems Laboratory
Cambridge, Massachusetts 02139

TE-43

COMPUTATIONAL COMPARISON OF STRAPDOWN SYSTEM ATTITUDE ALGORITHMS

by

Fred J. Marcus

B.S. (1968)

Massachusetts Institute of Technology

SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE

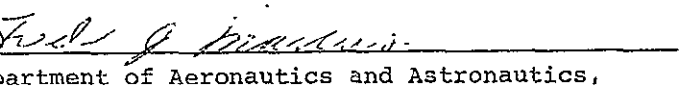
DEGREE OF MASTER OF

SCIENCE

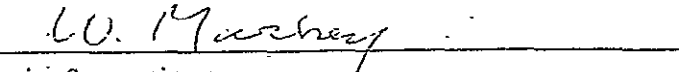
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

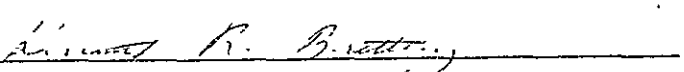
June 1971

Signature of Author 

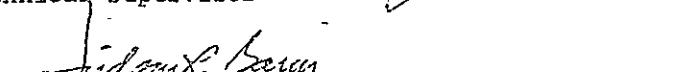
Department of Aeronautics and Astronautics, June 1971

Certified by 

Thesis Supervisor



Technical Supervisor

Accepted by 

Chairman, Departmental Committee on Graduate Students

Abstract

The accuracies of two strapdown system attitude algorithms are compared by digital computer simulation of the algorithms' response to specified angular input rates. For constant rate cases, both the direction cosine matrix (D.C.M.) technique and the Euler parameter technique gave rise to linearly growing drift angle errors. The errors incurred by the Euler parameter technique were significantly less than those incurred by the direction cosine matrix scheme.

For the cases where the simulated body motions were purely sinusoidal, the drift errors incurred by both techniques were bounded sinusoids having the same periods as the applied angular inputs. The magnitudes of the bounds were again less for those routines using Euler parameters rather than D.C.M. schemes. The magnitudes of the bounds for this case were also found to be proportional to the input angular rate raised to the power of the order of the numerical integration scheme employed.

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to his thesis supervisor, Professor Winston R. Markey, and to his technical supervisor, Kenneth R. Britting, whose generous advice and criticisms were very helpful in this effort. Mr. Edmund J. Koenke must also be thanked for the assistance he rendered during the preparation of this thesis.

Acknowledgement is also made to the M.I.T. Computation Center for its assistance in overcoming programming difficulties.

Grateful thanks are also extended to the author's wife for the patience and encouragement she offered.

This thesis was prepared under DSR project 70343, sponsored by the National Aeronautics and Space Administration, Electronics Research Center, Cambridge, Massachusetts, through NASA Grant Number NGR 22-009-229 and under Department of Transportation Contract DOT-TSC-143.

The publication of this thesis does not constitute approval by the National Aeronautics and Space Administration or by the M.I.T. Measurement Systems Laboratory of the findings or the conclusions contained therein. It is published only for the exchange and stimulation of ideas.

TABLE OF CONTENTS

CHAPTER 1.

Introduction

Strapdown navigation systems enable the determination of vehicle motions, position, and attitude with respect to some reference frame by using gyros and accelerometers which are fixed to the vehicle. Because these instruments are fixed to the vehicle, they are always in the same orientation with respect to any coordinate frame also fixed in the vehicle. This hypothetical frame fixed in the body is called the body frame.

In order to use the gyro and accelerometer measurements, it is necessary to have an accurate measurement of the orientation of the body frame with respect to the reference frame. The most common representation of this orientation is that of the direction cosine matrix. This is a matrix whose elements are the cosines of the angles between the axes of the body and reference frame. There are, however, other representations which can be used. Specifically, two of these are: (1) Euler parameters; and (2) Cayley-Klein parameters. The three techniques mentioned above are all governed by differential equations which are functions of vehicle rotations, and these equations can be solved by numerical integration on a computer using the gyro signals to indicate changes in vehicle orientation. The equations programmed in the computer for this purpose are called attitude algorithms.

It will be the purpose of this thesis to evaluate, by use of computer simulation, the relative merits of the three techniques mentioned above. The algorithms will be simulated as if they were computed on a general purpose computer. Another technique employing Euler angles to represent the relative orientation of the coordinate frames was not included in this analysis due to the singularities inherent with three parameter schemes. These singularities are the equivalent of "gimbal lock" situations found in physical represen-

tations employing a three gimbal system.[1] The merits of each algorithm can be determined by comparing the relative orientation computed by using each of the techniques (when specific vehicle maneuvers are prescribed) with the closed form solution of the direction cosine matrix. The accuracy of each of the algorithms will depend not only on vehicle maneuvers, but also on the integration schemes used to evaluate the differential equations and the rate extraction techniques used to determine the angular velocity of the vehicle. It will be assumed that integrating gyros which are sampled a number of times over each integration step will be utilized. These changes in accuracy by use of more sampling times or higher order numerical integration techniques will, of course, increase the computer time needed for each integration interval. The algorithms will be compared by: (1) determining relative accuracy for each algorithm using different integration routines (Runge-Kutta first order, second order, and fourth order), and specified vehicle motions; (2) determining relative accuracy for each algorithm when combined with an orthonormalization scheme; and (3) determining relative accuracy for each algorithm using different specified vehicle motions.

The accuracy will be determined by putting the results of all the techniques into their equivalent direction cosine matrices (D.C.M.'s). Because of the errors incurred in numerical integration, these D.C.M.'s may not be orthonormal; and different orthonormalization techniques will be used to determine which yields the best results. After orthonormalization, the D.C.M.'s calculated by using the two different algorithms will be compared by determining how large a rotation must be made with each computed reference frame to align it with the true reference frame.

The vehicle motions which will be used for this analysis are those for which closed form solution can be evaluated. These motions

will include: (1) single axis constant rotations; (2) three axis constant rotations; (3) single axis sinusoidal rotations; and (4) general coning motion. For each of these motions and different angular velocities, the accuracies of each algorithm will be compared.

The direction cosine algorithm computation using $\dot{B} = B\Omega$, (where Ω is the skew-symmetric form of the angular velocity of the body with respect to the reference frame, coordinatized in the body frame) seems to be the one most frequently used in strapdown systems. This thesis will try to either confirm the value of this algorithm or suggest the use of an alternative algorithm in order to yield the best results in a strapdown system.

CHAPTER 2.

Transformation Algorithms

The purpose of a transformation matrix is to transform the coordinates of a vector in one frame to their associated values in a second frame.

The transformation matrix accomplishes this transformation by a simple multiplication. If we let $\underline{V}^b$ be the coordinates of a vector in the body frame, $\underline{V}^i$ be the coordinates of the same vector in the inertial reference frame, and C_b^i be the coordinate transformation from the body frame to the reference frame, then:

$$\underline{V}^i = C_b^i \, \underline{V}^b$$

where each element of the transformation matrix C_b^i is the cosine of the angle between an axis of the reference frame and an axis of the body frame.[5]

The purpose of a transformation algorithm is to determine the elements of the transformation matrix as a function of time, given only the value of the elements of the transformation matrix at the initial time (t_0), and the angular motions of the body with respect to the reference frame coordinatized in the body frame. The latter can be obtained directly from the gyros mounted on the vehicle.

If the angular rotation of the body is:

$$\underline{\omega}(t) = \begin{bmatrix} \omega_x(t) \\ \omega_y(t) \\ \omega_z(t) \end{bmatrix}$$

where ω_x, ω_y, ω_z are the angular rates about the x, y, z body axis respectively, then

$$\dot{C}_b^i(t) = C_b^i(t)\ \Omega(t) \tag{2.1}$$

where Ω is the skew symmetric form of the angular rate vector, i.e.,

$$\Omega = \begin{pmatrix} 0 & -\omega_z(t) & \omega_y(t) \\ \omega_z(t) & 0 & -\omega_x(t) \\ -\omega_y(t) & \omega_x(t) & 0 \end{pmatrix}$$

Equation 2.1 actually represents nine first order coupled differential equations. It is assumed that $C_b^i\ (t_0)$ and $\underline{\omega}(t)$ are known. Therefore, $C_b^i\ (t)$ can be obtained for all t by numerically integrating these equations on a general purpose computer. This scheme will be identified as the "Direction Cosine Matrix Algorithm," and should not be confused with the direction cosine matrix itself.

The Euler parameter algorithm uses a four parameter technique.[1] When Euler formulated this technique, he found that by using three parameters instead of two to represent the components of an axis about which one coordinate frame was rotating with respect to another, he could avoid the problem of singularities. A complete derivation of this technique can be found in Reference 2.

Instead of having nine differential equations to solve, the Euler parameter technique has only four. The Euler parameter technique can be summarized by the following equations:

Let θ be the vector whose components are the Euler parameters

$$\underline{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}$$

This vector will then satisfy the following differential equation

$$\frac{d\underline{\theta}}{dt} = \frac{1}{2} \begin{bmatrix} 0 & \omega_z & -\omega_y & -\omega_x \\ -\omega_z & 0 & \omega_x & -\omega_y \\ \omega_y & -\omega_x & 0 & -\omega_z \\ \omega_x & \omega_y & \omega_z & 0 \end{bmatrix} \underline{\theta}$$

and the direction cosine matrix can be obtained from purely algebraic manipulations as follows

$$C_b^i = \begin{bmatrix} \theta_1^2+\theta_4^2-\theta_2^2-\theta_3^2 & 2(\theta_1\theta_2+\theta_3\theta_4) & 2(\theta_1\theta_3-\theta_2\theta_4) \\ 2(\theta_1\theta_2-\theta_3\theta_4) & -\theta_1^2+\theta_2^2+\theta_4^2-\theta_3^2 & 2(\theta_2\theta_3+\theta_1\theta_4) \\ 2(\theta_1\theta_3+\theta_2\theta_4) & 2(\theta_2\theta_3-\theta_1\theta_4) & -\theta_1^2-\theta_2^2+\theta_3^2+\theta_4^2 \end{bmatrix}$$

Using this technique, we have to integrate only four differential equations. However, these four values must be put into the D.C.M. form in order to use their information as a transformation from the body frame to the reference frame. The effort involved in accomplishing

this task is comprised solely of simple algebraic manipulations and is not an iterative procedure, as is the integration of the differential equations.

Finally, the third algorithm to be mentioned is that using Cayley-Klein parameters. It is also a four parameter technique satisfying the differential equation.

$$\frac{d}{dt}\begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\\ \alpha_4\end{bmatrix} = \frac{1}{2}\begin{bmatrix}0 & -\omega_z & -\omega_y & -\omega_x\\ \omega_z & 0 & -\omega_x & \omega_y\\ \omega_y & \omega_x & 0 & -\omega_z\\ \omega_x & -\omega_y & \omega_z & 0\end{bmatrix}\begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\\ \alpha_4\end{bmatrix}$$

and the direction cosine matrix can be formed according to the following equation.

$$C_b^i = \begin{bmatrix}\alpha_1^2-\alpha_2^2-\alpha_3^2+\alpha_4^2 & 2(-\alpha_1\alpha_2+\alpha_3\alpha_4) & 2(\alpha_1\alpha_3+\alpha_2\alpha_4)\\ 2(\alpha_1\alpha_2+\alpha_3\alpha_4) & \alpha_1^2-\alpha_2^2+\alpha_3^2-\alpha_4^2 & 2(-\alpha_1\alpha_4+\alpha_2\alpha_3)\\ 2(-\alpha_1\alpha_3+\alpha_2\alpha_4) & 2(\alpha_1\alpha_4+\alpha_2\alpha_3) & \alpha_1^2+\alpha_2^2-\alpha_3^2-\alpha_4^2\end{bmatrix}$$

If we let

$$\alpha_1 = \theta_4$$
$$\alpha_2 = -\theta_3$$
$$\alpha_3 = -\theta_2$$
$$\alpha_4 = -\theta_1$$

we then have identical differential equations and identical forms for the transformation matrix in both the Euler parameter and the Cayley-Klein parameter techniques.

Because these two techniques are really the same, the computer analyses run on the IBM 360 were only programmed for Euler parameters. The results for the Cayley-Klein parameters would have been identical to these. There is no practical reason why one of the four parameter techniques should be favored over the other, with respect to accuracy, ease of programming, or computer time needed to process the algorithm.

CHAPTER 3.

Rate Extraction, Integration Routines, and Orthonormalization Routines

A. Rate Extraction

It was shown that the attitude algorithm requires continual knowledge of $\underline{\omega}(t)$. However, most systems use pulse torqued integrating gyros whose output is not in the form of angular velocities, but rather in the incremental form $\Delta\theta(t)$, where $\Delta\theta(t)$ is an incremental change in angular orientation. It is necessary to be able to extract rate information $\underline{\omega}(t)$ from the series of pulse trains $\Delta\theta(t)$ for the Runge-Kutta integration routines which were used to integrate the algorithms' differential equations. A first order approach to rate extraction would be to let

$$\omega(t) = \frac{d\theta}{dt} = \lim_{\Delta t \to 0} \frac{\Delta\theta(t+\Delta t) - \Delta\theta(t)}{\Delta t}$$

Some integration routines, however, also need rate information at the midpoint of the integration interval, i.e., $\omega(t+\frac{t}{2})$. For this reason, a second order rate extraction scheme was used in the computer simulations which were implemented.

This second order scheme samples the gyro output at times t, $t + \frac{\Delta t}{2}$, and $t + \Delta t$. Using these three points, and simulating the gyro output increments as

$$\Delta\theta_1(t) = \theta(t + \frac{\Delta t}{2}) - \theta(t)$$

$$\Delta\theta_2(t) = \theta(t + \Delta t) - \theta(t + \frac{\Delta t}{2}),$$

where $\theta(t)$ is the actual simulated rotation, and fitting the points with a second order polynomial, yields after differentiation:[1]

$$\omega(t) = \frac{3\Delta\theta_1(t) - \Delta\theta_2(t)}{\Delta t}$$

$$\omega(t + \frac{\Delta t}{2}) = \frac{\Delta\theta_1(t) + \Delta\theta_2(t)}{\Delta t}$$

$$\omega(t + \Delta t) = \frac{-\Delta\theta_1(t) + 3\Delta\theta_2(t)}{\Delta t}$$

Other techniques using higher sampling rates for the gyros can yield better approximations to $\omega(t)$ than the method just described. The advantages and disadvantages of these techniques well be discussed later.

B. Integration Schemes

The rate information determined in the previous section can now be used as the input to the integration routines, which were used to get solutions from the algorithms' differential equations. In the computer simulations performed, it was arbitrarily decided to let the initial conditions be set by the fact that at $t = 0$ the body frame of the vehicle is exactly aligned with the reference frame. The magnitudes and general form of the errors associated with the algorithms would not be changed by using other initial conditions, and exact initial alignment was chosen merely for convenience. For the direction cosine matrix technique $\dot{C}_b^i(t) = C_b^i(t)\Omega(t)$, the initial conditions imply that $C_b^i(t_0)$ equals the identity matrix. For the Euler parameter technique, the initial conditions implying exact alignment are:

$$\theta_1 = 0.0$$
$$\theta_2 = 0.0$$
$$\theta_3 = 0.0$$
$$\theta_4 = 1.0$$

For the purposes of this simulation, it was decided to test both algorithms using a first, second, and fourth order Runge-Kutta integration routine with fixed time steps of integration. Third order routines were not employed, due to the fact that some of the cases simulated in this analysis have been previously analyzed using closed-form analytic techniques when first, second, and third order integration routines were employed.[2] It was hoped that use of first, second and fourth order techniques in this analysis would computationally confirm and extend the results of previous analytic comparisons. On an actual flight, it would be advantageous to be able to sample the gyros at a predetermined rate and also compute over predetermined intervals of integration, so that the amounts of computer time needed to implement a particular algorithm would be known exactly. The use of a variable step size integration routine would not permit the evaluation of the computation time needed because it would be impossible to determine beforehand what integration step size would be used.

Runge-Kutta integration techniques were employed because they do not require past histories of the dependent variables, and they have an accuracy equivalent to the accuracy of a Taylor series solution of the same order.

The equations governing these integration routines are shown below.[3,4]

Letting $\dot{\underline{x}}(t) = \underline{f}[x(t), \underline{\omega}(t)]$, the equations become:

First-Order Runge-Kutta

$$\underline{x}(t + \Delta t) = \underline{x}(t) + \Delta t\, [f(\underline{x}(t), \underline{\omega}(t))]$$

Second-Order Runge-Kutta

$$\underline{y} = \underline{x}(t) + \Delta t\, [\underline{f}(\underline{x}(t), \underline{\omega}(t))]$$

$$\underline{x}(t + \Delta t) = \underline{x}(t) + \frac{\Delta t}{2} [\underline{f}(\underline{y}, \underline{\omega}(t + \Delta t)) + \underline{f}(\underline{x}(t), \underline{\omega}(t))]$$

Fourth-Order Runge-Kutta

$$\underline{A} = \Delta t\ \underline{f}[\underline{x}(t),\ \underline{\omega}(t))]$$

$$\underline{B} = \Delta t\ \underline{f}[\underline{x}(t) + \frac{\underline{A}}{2},\ \underline{\omega}(t + \frac{\Delta t}{2})]$$

$$\underline{C} = \Delta t\ \underline{f}[\underline{x}(t) + \frac{\underline{B}}{2},\ \underline{\omega}(t + \frac{\Delta t}{2})]$$

$$\underline{D} = \Delta t\ \underline{f}[\underline{x}(t) + \underline{C},\ \underline{\omega}(t + \Delta t)]$$

$$\underline{x}(t + \Delta t) = \underline{x}(t) + \frac{\underline{A}}{6.0} + \frac{\underline{B}}{3.0} + \frac{\underline{C}}{3.0} + \frac{\underline{D}}{6.0}$$

C. Orthonormalization

The elements of the direction cosine matrices computed by either the direction cosine matrix technique or the Euler parameter technique will contain errors due to the fact that numerical integration routines are not exact. Because of these errors, the computed D.C.M.'s will not necessarily have the property of being orthonormal.

Orthonormality is a requisite of a true D.C.M. and can be expressed as follows. If $\underline{C}_i$; i = 1,2,3; are the three rows of the D.C.M., normality requires that

$$|\underline{C}_i| = 1 \quad i = 1,2,3$$

and orthogonality requires that

$$\underline{C}_1 \times \underline{C}_2 = \underline{C}_3$$

$$\underline{C}_2 \times \underline{C}_3 = \underline{C}_1$$

$$\underline{C}_3 \times \underline{C}_1 = \underline{C}_2$$

where "x" denotes the cross product operator. The algorithms used in this analysis were also tested with the orthonormalization routines included, to determine how much improvement could be obtained in the

For the Euler parameter technique, orthonormalization merely requires that:

$$\sum_{i=1}^{4} (\theta_i)^2 = 1$$

If this requirement is met, the rows and columns of the D.C.M. formed from the Euler parameters have magnitude equal to one, and they are mutually orthogonal. Therefore, the D.C.M. formed by these parameters will be automatically orthonormal. Therefore, in the computed simulation, orthonormalization was achieved by dividing each of the computed parameters by the square root of the sum of the squares of the computed parameters.

Orthonormalization for the direction cosine matrix technique is not nearly as simple as for the four parameter algorithm. For this case, a technique was used which was not too complicated but does normalize the computed D.C.M. and orthogonalizes the computed D.C.M. to second-order in the original orthogonality errors(see Appendix B).

The procedure used was to let $\underline{B}_i$; i = 1,2,3; be the rows of the computed D.C.M.

Then form:

$$\underline{C}_i = \underline{B}_j \times \underline{C}_k$$

where i = 1, j = 2, k = 3; i = 2, j = 3, k = 1; and i = 3, j = 1, k = 2. Next let

$$\hat{\underline{B}}_i = \frac{\underline{B}_i + \underline{C}_i}{|\underline{B}_i + \underline{C}_i|}$$

and the $\hat{B}_i$ are now the rows of the orthonormalized D.C.M.

The direction cosine matrices were orthonormalized each time the direction cosine matrix was formed, i.e. every ten seconds for the simulations performed.

CHAPTER 4.

Error Quantities and Truth Models

A. Error Quantities

Use of a computer to integrate the differential equations of the attitude algorithms discussed in the previous chapter, yields a computed direction cosine matrix. The elements of the computed direction cosine matrix will differ from those of the true direction cosine matrix representing the relative orientation of the body frame and the reference frame, due to errors in the numerical integration techniques, errors in the rate extraction schemes and numerical round off in the computer. In the present analysis, the errors due to numerical round off were reduced enough to become insignificant by using double precision in all the programs run on the computer. Merely comparing the difference of the elements of these two matrices does not, however, yield a good physical representation of the errors in the computed D.C.M.

In order to obtain a somewhat physical representation of the errors associated with the computed direction cosine matrix, the following scheme can be used.[1,2]

Let $\hat{C}$ be the computed D.C.M. between the body and reference frames, C be the exact D.C.M. between the two frames, and C^T be the transpose of C.

Then

$$\hat{C} = C + \delta C$$

or

$$\hat{C} = (I + E)C$$

where

$$E = \delta CC^T$$

and the fact that $CC^T = I$ yields $E = \hat{C}C^T - I$. The error matrix, E, will be of the form:

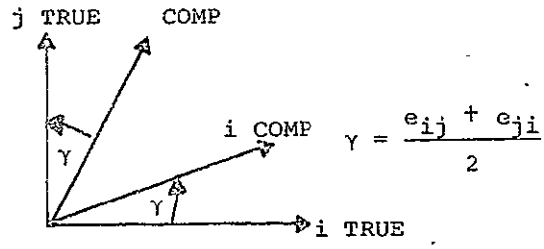
$$E = \begin{pmatrix} \varepsilon_{11} & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & \varepsilon_{22} & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & \varepsilon_{33} \end{pmatrix}$$

The elements of the error matrix can be given a physical interpretation.[1] The elements along the main diagonal indicate to first order the growth or decrement, of the column they are in, from unity. This error is called the "scale error" and is a dimensionless quantity.

Letting ε_{ij} be the element in the i^{th}column and i^{th} row of the error matrix, "skew errors" can be defined as the average of the off diagonal elements of the error matrix, i.e.

$$(\text{Skew})_k = \frac{\varepsilon_{ij} + \varepsilon_{ji}}{2} \quad \text{for} \quad \begin{cases} k = 1, i = 2, j = 3 \\ k = 2, i = 1, j = 3 \\ k = 3, i = 1, j = 2 \end{cases}$$

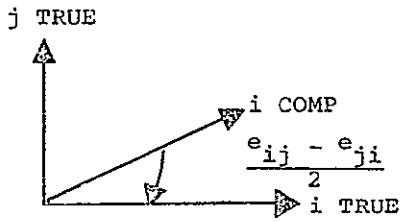
Physically the skew error is a measure of the perpendicularity of the i and j axes, as shown by the following figure:



Finally, a "drift error" can be associated with the error matrix.

$$\text{Drift Error}_k = \frac{\varepsilon_{ij} - \varepsilon_{ji}}{2} \quad \text{for} \quad \begin{cases} k = 1,\ i = 2,\ j = 3 \\ k = 2,\ i = 3,\ j = 1 \\ k = 3,\ i = 1,\ j = 2 \end{cases}$$

The drift error is a measure of how much the computed frame must be rotated about the K^{th} axis to align it with the true frame.
A geometric interpretation of the drift error appears below.



The scale errors as defined are dimensionless quantities. The skew and drift errors are in radians, but can easily be converted to degrees, as was done for this simulation.

Using these errors as a measure of accuracy for the different algorithms tested easily lends physical insight to the determination of the merits of each algorithm.

B. Truth Models and Input Rates

In order to calculate the error matrix $\underline{E}$, described in the previous section, it is necessary to have knowledge of the true D.C.M. Four basic types of vehicle rotations were used in this computer analysis, and a truth model which could compute the true D.C.M. in closed form was needed.

1. Single Axis, Constant Rate

For the single axis constant rate rotations, the D.C.M. is merely a function of the angle through which the body axis was rotated. Assuming an x, y, z coordinate system, the simulated rotations were input about the y axis, i.e.

$$\omega(t) = \begin{Bmatrix} 0 \\ k \\ 0 \end{Bmatrix} \qquad k = \text{constant}$$

then

$$\theta_y(t) = \theta_0 + kt \qquad \theta_0 = 0$$

$$\theta_x(t) = 0$$

$$\theta_z(t) = 0$$

and

$$C_{B\ \text{True}}^{I}(t) = \begin{pmatrix} \cos\theta_y(t) & 0 & \sin\theta_y(t) \\ 0 & 1 & 0 \\ -\sin\theta_y(t) & 0 & \cos\theta_y(t) \end{pmatrix}$$

2. Single Axis, Sinusoidal Rates

The single axis, sinusoidal rates case also used the y body axis as the axis of rotation. The form of the sinusoidal input used was:

$$\omega(t) = \alpha\cos\beta t$$

As for the single axis case, the true D.C.M. is only a function of $\theta_y(t)$, which was evaluated according to the following equation:

$$\theta_y(t) = \theta_0(t) + \int_{t_0}^{t} \alpha\cos\beta t = 0 + \frac{\alpha}{\beta}\sin\beta\tau \Big]_{t_0}^{t}$$

Therefore

$$\theta_y(t) = \frac{\alpha}{\beta}\sin\beta t \qquad 4.1$$

and the construction of the true D.C.M. is the same as for the single axis case with equation 3.1 being used to evaluate $\theta_y(t)$.

3. Three Axes, Constant Rates

For this case, a constant angular rate was input into all three axes. To evaluate the truth model for this case, we let

$$\dot{C}(t)_{true} = C(t)\Omega(t)_{true}$$

which is the differential equation used in the direction cosine matrix technique.

If we let $\Phi(t,t_0)$ be the transition matrix for this equation, it can be shown (see Appendix A) that the true direction cosine matrix can be represented as:

$$C_{B_{true}}^{I}(t) = \left[I + \frac{\Omega^2}{\omega^2}(1 - \cos \omega t) - \frac{\Omega}{\omega} \sin \omega t\right]^T$$

where $\omega = [\omega_x^2(t) + \omega_y^2(t) + \omega_z^2(t)]^{\frac{1}{2}}$

and $\Omega(t)$ = skew symmetric form of $\underline{\omega}(t)$.

4. Coning Motion Superimposed on Constant Pitch Rate

The angular input for this case is:

$$\underline{\omega}(t) = \begin{Bmatrix} \alpha \sin \beta t \\ \gamma \\ \alpha \cos \beta t \end{Bmatrix}$$

For this case, it can be shown (see Appendix A) that the true D.C.M. as a function of time is as follows:

$$C_B^I(t) = \left[I + \frac{\Omega^2}{\omega^2}(1 - \cos \omega t) + \frac{\Omega}{\omega} \sin \omega t\right] C_B^A(t)$$

where

$$\underline{\omega}(t) \begin{bmatrix} 0 \\ \beta + \gamma \\ \alpha \end{bmatrix} ; \omega = (\alpha^2 + (\beta + \gamma)^2)^{\frac{1}{2}}$$

$\Omega(t)$ is the skew symmetric form of $\omega(t)$

$$\text{and } C_B^A(t) = \begin{pmatrix} \cos \beta t & 0 & -\sin \beta t \\ 0 & 1 & 0 \\ \sin \beta t & 0 & \cos \beta t \end{pmatrix}$$

Besides evaluating truth models for the analysis, we must provide our computational algorithms with pulses $\Delta\underline{\theta}(t)$, in order to enable evaluation of the angular velocity, $\underline{\omega}(t)$, as it would be accomplished by measuring $\Delta\underline{\theta}(t)$ from the gyro outputs. The $\Delta\underline{\theta}(t)$ used for the analysis was not quantized, as would be the case with a physical system, although evaluation of errors from this source might be desirable if the level of quantization were so large as to compete with the drift errors which come about from only approximate integration of the algorithm's differential equations.

a. For the single axis constant rate case

$$\underline{\omega}(t) = \begin{Bmatrix} 0 \\ k \\ 0 \end{Bmatrix}$$

$$\Delta\underline{\theta}_1(t) = \Delta\underline{\theta}_2(t) = \int_t^{t+\frac{\Delta t}{2}} \underline{\omega}(t) = \begin{Bmatrix} 0 \\ k\frac{\Delta t}{2} \\ 0 \end{Bmatrix}$$

Thus $k\Delta t$ was the pulse input to the algorithm which then used the rate extraction routine described in Chapter 2 to derive rate information.

b. For the three axes constant rate case

$$\omega(t) = \begin{Bmatrix} \alpha \\ \beta \\ \gamma \end{Bmatrix}$$

and the pulses used by the algorithms were

$$\Delta\underline{\theta}_1(t) = \Delta\underline{\theta}_2(t) = \int_t^{t+\frac{\Delta t}{2}} \underline{\omega}(t) = \begin{Bmatrix} \frac{\alpha\Delta t}{2} \\ \frac{\beta\Delta t}{2} \\ \frac{\gamma\Delta t}{2} \end{Bmatrix}$$

c. For the single axis sinusoidal case,

$$\underline{\omega}(t) = \begin{Bmatrix} 0 \\ \alpha \cos \beta t \\ 0 \end{Bmatrix}$$

and therefore

$$\Delta\underline{\theta}_1(t) = \begin{bmatrix} 0 \\ \int_t^{t+\frac{\Delta t}{2}} \alpha \cos \beta t \, dt \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{\alpha}{\beta}\left[\sin \beta(t + \frac{\Delta t}{2}) - \sin \beta t\right] \\ 0 \end{bmatrix}$$

$$\Delta\underline{\theta}_2(t) = \begin{bmatrix} \\ \int_{t+\frac{\Delta t}{2}}^{t+\Delta t} \alpha \cos \beta t \, dt \\ \\ 0 \end{bmatrix} = \begin{bmatrix} \\ \frac{\alpha}{\beta}[\sin\beta(t+\Delta t) - \sin\beta(t+\frac{\Delta t}{2})] \\ \\ 0 \end{bmatrix}$$

d. For the coning motion superimposed on a pitch rate, the angular rate is

$$\underline{\omega}(t) = \begin{Bmatrix} \alpha \sin \beta t \\ \gamma \\ \alpha \cos \beta t \end{Bmatrix}$$

and therefore

$$\Delta\underline{\theta}_1(t) = \begin{bmatrix} \frac{\alpha}{\beta}[\cos \beta(t) - \cos \beta(t+\frac{\Delta t}{2})] \\ \gamma \frac{\Delta t}{2} \\ \frac{\alpha}{\beta}[\sin \beta(t+\frac{\Delta t}{2}) - \sin \beta t] \end{bmatrix}$$

$$\Delta\underline{\theta}_2(t) = \begin{bmatrix} \frac{\alpha}{\beta}[\cos \beta(t+\frac{\Delta t}{2}) - \cos \beta(t+\Delta t)] \\ \gamma \frac{\Delta t}{2} \\ \frac{\alpha}{\beta}[\sin \beta(t+\Delta t) - \sin \beta(t+\frac{\Delta t}{2})] \end{bmatrix}$$

It should be reemphasized that the angular rates could have been supplied to the algorithm directly; but doing this would have neglected the errors incurred by the algorithms due to the fact that they do not have exact means of rate extraction.

CHAPTER 5

Results of the Computer Analysis

In order to compare the accuracies of the previously discussed algorithms, four types of vehicle motions were simulated. For each simulated vehicle maneuver, both the direction cosine matrix and Euler parameter techniques were employed using first, second and fourth order integration routines. The integration interval was kept at a constant value of 1/8 second, and all of the test runs were simulated for a period of three minutes.

The drift, scale, and skew errors found in each of the computed D.C.M.'s were then compared in an attempt to evaluate the relative merits of each of the algorithms tested. Although each type of vehicle maneuver was input separately, the generality of the programs allow any combination of vehicle maneuvers to be input, as long as it is still possible to formulate a closed form solution of the D.C.M. to use as a truth model. It was felt, however, that the maneuvers which were used for the present analysis are representative of common motions an actual vehicle would undergo during flight.

A. Single Axis Constant Rate Case

As previously discussed, a constant angular velocity of 10°/second around only one body axis was the simulated vehicle maneuver for this test case. For this type of input, the rate extraction routine will yield perfect rate information in the absence of gyro pulse quantization. Therefore, the errors found in the computed direction cosine matrices will only depend upon the type of attitude algorithm employed, and upon the order of the integration technique used to integrate the differential equations of the algorithm. Errors which might have resulted due to the fact that digital computers only have finite word length, were made insignificant with the simulations

performed by using double precision word length with all of the computer programs.

For this simulated vehicle maneuver, it was found that both the scale and drift errors grew linearly as a function of time, independent of the algorithm and integration routine employed to compute the D.C.M. Also, the skew errors were identically zero over the complete simulation period.

These same simulations were also run with an orthonormalization routine included in the algorithm. The orthonormalization routines were used to orthonormalize the computed D.C.M. every five seconds during the three minute simulated maneuver time. It was found that the use of an orthonormalization scheme decreased the scale errors by a factor on the order of 10^{-4} for those algorithms using a first or second order integration routine. The addition of an orthonormalization scheme to the algorithms using a fourth order integration scheme did not yield a significant improvement in the scale errors. It is important to note that the orthonormalization schemes did not improve the drift errors incurred by the algorithms, and in some cases they actually increased the drift errors(see Appendix B).

Orthonormalization routines which were only employed at the end of the three minute simulation period were also analyzed. It was found that orthonormalization for those cases yielded much less significant improvement in scale errors than did orthonormalization routines which were used continuously throughout the simulations.

If, however, it is required that the drift errors should be small, i.e., on the order of the size of the quantization level of a typical gyro pulse (approximately .007°), it is found that the addition of an orthonormalization routine is of no advantage. The drift errors are not improved, and the scale errors are already in the 10^{-10} range.

The single axis constant input test case showed the drift errors

incurred by the Euler parameter technique with any order integration routine tested were always lower than the drift errors incurred by the direction cosine matrix technique using the same order integration routine. Euler parameters yield 1/4 the drift errors that the direction cosine matrix scheme does for both the first and second order integration techniques, and 1/16 the drift error for the fourth order integration techniques.

With orthonormalization every 5 seconds included, the drift errors at the end of three minutes, normalized with respect to the drift error incurred by the direction cosine matrix technique using a first order integration routine, (0.285°), can be summarized as follows:

Table 1.

Drift Error	1st Order	2nd Order	4th Order
$\dot{B} = B\Omega$	1	.5	$.12 \times 10^{-4}$
E.P.	.25	.125	$.75 \times 10^{-6}$

$\dot{B} = B\Omega$ represents the direction cosine matrix technique.

E.P. represents the Euler parameter technique.

$\omega = 10°/sec.$; $\Delta t = 1/8$ sec.

The elements of Table 1 were found to be constant for first and second order integration techniques, for varying integration interval, Δt, and for varying angular rate, $\underline{\omega}$. However, the fourth order techniques were shown to improve relative to the first and second order techniques as the ratio $\omega/\Delta t$ was decreased. The fourth order Euler parameters continued, however, to incur only 1/16 the drift error incurred by fourth order direction cosine matrix techniques as $\omega/\Delta t$ was decreased.

B. Three Axis Constant Rate Case

In this test case, constant angular rates of 10°/second were input along the three orthogonal body axes. The resulting error analysis showed the scale, skew, and drift errors to be the same along all three axes. As in the single axis case, it was found that requiring the drift errors to be small made the use of an orthonormalization scheme to reduce scale and skew errors unnecessary. The drift for this case using D.C.M. technique with a first order integration scheme was found to be 2.391°.

A summary of the normalized drift errors can be seen in Table 2.

Table 2.

Drift Order	1st Order	2nd Order	4th Order
$\dot{B} = B\Omega$	1	.5	$.26 \times 10^{-4}$
E.P.	.25	.125	$.225 \times 10^{-5}$

Again it can be seen that for first and second order integration routines, the Euler parameter technique yields a drift error of 1/4 the drift error incurred by the direction cosine matrix technique; and for fourth order integration routines, the drift error ratio is 1/16. It should also be noted that the fourth order integration techniques in this case do not have the same relative improvement over first and second order techniques as they did in the single axis constant rate constant rate test case. Their relative improvement was reduced by a factor of three from its value in the single axis case, although for the value of $\omega/\Delta t$ used they still held a commanding advantage.

If the results of these first two test cases are compared with the analytical work of McKern,[2] it can be seen that using a second order integration routine with either Euler parameters or direction

cosine matrix schemes yields drift errors of half those which would result from using a first order integration routine with the algorithms, independent of the value $\omega/\Delta t$. Use of a fourth order routine, however, yields only 1/4 the drift error of a third order routine, also independent of the value $\omega/\Delta t$.

C. Single Axis Sinusoidal Rate Case

For this test case, the simulated vehicle angular rate was of the form $\omega = \beta \cos\beta t$. For computer runs made, using a simulated vehicle rate of this form meant that by varying β, the vehicle would perform constant amplitude but varying frequency oscillations.

The results of the error analysis showed that there was no skew error incurred, and that the scale error did not need improvement by an orthonormalization routine when the drift errors were kept small. Most important was the result that the drift errors for this case did not grow linearly with time as in the constant rate test cases, but instead were bounded sinusoids having the same period as did $\sin\beta t$.

The drift errors for this case had another interesting dependence upon β. For the first order integration schemes, using either Euler parameters or the direction cosine matrix technique algorithms, drift errors were directly proportional to β. With second order integration techniques, the drift errors for both algorithms were proportional to β^2, and for fourth order integration techniques the drift errors were proportional to β^4. No computer runs were made with a third order integration routine, but it seems reasonable to speculate that the drift errors for a third order routine would be proportional to β^3.

The ratios of Euler parameter drift errors vs. direction cosine matrix drift errors for this case is shown in Table 3.

Table 3.

Drift Errors	1st Order	2nd Order	4th Order
$E.P./\dot{B} = B\Omega$	1	1/4	1/16

These ratios held constant as the value of β was varied. These ratios are the same as they are for both the single and three axis constant rate cases when second or fourth order integration routines are used. However, the advantage held by the Euler parameters decreased sharply for the first order integration routines.

D. Coning Motion Superimposed on a Constant Pitch Rate

For this simulation, the simulated angular rate of the vehicle was:

$$\underline{\omega}(t) = \begin{bmatrix} \alpha \sin \beta t \\ \gamma \\ \alpha \cos \beta t \end{bmatrix}$$

As in the sinusoidal rate case, the magniutdes of α and β were kept the same.

The results of the computer analysis showed the drift errors incurred along all three axes to be irregular functions of time, i.e., they were not linear and they did not have any simple sinusoidal form. In order to compare the accuracies of the algorithms tested, a mean drift was defined as follows:

$$\text{Mean Drift} = (D_x^2 + D_y^2 + D_z^2)^{\frac{1}{2}}$$

where D_x, D_y, D_z was the maximum drift achieved during the three minute simulations about the x, y, and z axes respectively. Two basic test inputs were simulated for this case. The first used

$$\alpha = \gamma = 1°/\text{sec.} \qquad \beta = .1$$

and the second used

$$\alpha = \gamma = 10°/\text{sec.} \qquad \beta = 10.0$$

The normalized mean drift errors is shown in Tables 4 and 5. The mean drifts for these cases when the D.C.M. technique with first order integration was employed were 6.25×10^{-2} and 10.2×10^{-1} degrees respectively.

Table 4.

Normalized Mean Drift Errors

	1st Order	2nd Order	4th Order
$\dot{B} = B\Omega$	1	$.71 \times 10^{-2}$	$.30 \times 10^{-8}$
E.P.	1	$.27 \times 10^{-2}$	$.16 \times 10^{-9}$

$\alpha = \beta = \gamma = 1$

Table 5.

Normalized Mean Drift Errors

	1st Order	2nd Order	4th Order
$\dot{B} = B\Omega$	1	.44	$.19 \times 10^{-4}$
E.P.	.66	.16	$.23 \times 10^{-6}$

$\alpha = \beta = \gamma = 10$

The most significant result of the analysis is that again Euler parameters had mean drift errors less than or equal to those incurred by the direction cosine matrix technique, when the algorithms were tested with the same simulated input and evaluated with the same order integration technique.

The coupling of the algorithms' differential equations, and dissimilarity of the angular rates imposed upon each axis for this test case, made it difficult to derive a concise correlation between the magnitudes of the inputs and the resulting mean drift errors incurred by the algorithms.

CHAPTER 6

Conclusions

For every case tested, the Euler parameter algorithm proved to be at least as good as, and in most cases superior to the direction cosine matrix algorithm, when both used the same order integration techniques. It would seem wiser, therefore, to choose a four parameter technique in favor of the direction cosine matrix scheme which employs nine differential equations.

Although the orthonormalization algorithm used with the direction cosine matrix algorithm was only an approximate scheme, the method employed with the Euler parameter algorithm was exact, and showed that orthonormalization was useful only for reducing scale and skew errors. It further proved orthonormalization to be completely unnecessary when the algorithm employed uses an integration interval small enough to maintain the drift errors at a magnitude on the order of the quantization level of a typical gyro pulse. For those reasons, it would seem appropriate not to waste computer time by employing an orthonormalization routine when the above conditions are met.

It was mentioned in a previous chapter that the rate extraction routine used for these simulations was not the only choice available. One could in fact sample the gyro outputs more often over the integration interval in order to derive more accurate rate information. For example, the gyros' outputs could be sampled more frequently and be fitted with a third rather than a second order polynomial. However, choice of the optimum rate extraction scheme to be used would depend upon a priori knowledge of the angular rates to be undergone by the vehicle. The simulations run for this paper were not employed to explore the advantages of more accurate rate extraction routines. However, the generality of the computer programs written for this

analysis could easily handle changes such as this. It might be useful, therefore, in future work to explore the effect of rate extraction upon the accuracies of the attitude algorithms.

It must also be reemphasized that the simulation runs employed double precision throughout. This yields an accuracy of 15 decimal digits for each number stored in the computer. When some of the simulations were carried out in single precision (accuracy 7 decimal digits), it was found that computer round off became one of the most prominent sources of inaccuracy in the algorithms. Typical on-board computers do not have as long a word length as does the I.B.M. 360 with double precision. The accuracy of the on-board computer will, therefore, be a great source of concern when attempting to implement an attitude algorithm with a strapdown inertial navigation system.

Another important aspect concerning the accuracy of an attitude algorithm is the fact that each pulse from an integrating gyro is obtained only after the vehicle has undergone a certain minimum change in angular orientation. This minimum change to produce a pulse can be identified as the pulse quantization level of the gyro being considered. This simulation did not take this quantization into account, but again the programs could easily be modified to do so. The pulse quantization level would set a minimum magnitude on the drift accuracy of the algorithms due to the fact that the algorithms cannot be expected to have accuracies greater than the information supplied to them. The build-up of errors due to gyro quantization could be a study by itself, but for the purpose of these simulations it was ignored. It would, however, be reasonable to assume that gyro quantization errors would only be important when algorithm drift errors are about the same order of magnitude as the quantization levels. For much larger drift errors, the errors caused by quantization would in all likelihood be overshadowed by the errors incurred due to only

finite order numerical integration and errors in the rate extraction schemes. It might be possible to analyze effects of gyro quantization by merely looking at the effect they would have on rate extraction information.

Finally, it must be noted that although higher order integration routines yield better accuracies, they also take more time for computation. In some applications it might be more commendable to use a low order integration routine and small integration interval rather than a high order integration routine and large integration interval. To make a good choice would depend upon knowledge of typical vehicle maneuvers for the application of the strapdown system computation times required by the general purpose computer being used, and the accuracies required of the algorithm.

Although this study does not indicate what is the optimum rate extraction or optimum order integration routine to employ, it does strongly indicate that the four parameter techniques have a distinct advantage over the direction cosine matrix method, both of which yield attitude algorithms with no singularities.

REFERENCES

1. Koenke, E.J., and Downing, D.R., Evaluating Strapdown Algorithms: A Unified Approach, Guidance and Control Programs Office, Electronics Research Center, N.A.S.A., Cambridge, Mass., 1968.

2. McKern, R.A., A Study of Transformation Algorithms for Use in a Digital Computer, Aero. Dept., M.I.T., Cambridge, Mass., 1968.

3. Arden, D.W., An Introduction to Digital Computing, Addison-Wesley, Reading, Mass., 1963.

4. Macon, N., Numerical Analysis, John Wiley & Sons, New York, N.Y., 1963.

5. Broxmeyer, C., Inertial Navigation Systems, McGraw-Hill, New York, N.Y., 1964.

6. Ogata, K., State Space Analysis of Control Systems, Prentice-Hall, Englewood Cliffs, N.J., 1967.

APPENDIX A.

Closed Form Solutions for the Direction Cosine Matrix

The differential equation for the D.C.M. is

$$\dot{C}(t) = C(t)\Omega(t). \qquad A.1$$

This matrix equation actually represents nine first order differential equations and can be rewritten as

$$\dot{C}^T(t) = \Omega^T(t)C^T(t). \qquad A.2$$

The solution to Equation A.1 can be expressed using state space techniques as[6]

$$C^T(t) = \Phi(t,t_0)C^T(t_0). \qquad A.3$$

Throughout the remainder of this appendix the initial values will be set as follows

$$t_0 = 0,$$

$$C(t_0) = C(0) = C_0.$$

The differential equation for the state transition matrix in Equation A.3 can be expressed as

$$\dot{\Phi}(t,0) = -\Omega(t)\Phi(t,0), \qquad A.4$$

because $\Omega(t)$ is skew symmetric, i.e.

$$\Omega^T(t) = -\Omega(t); \qquad A.5$$

and the boundary condition for the differential equation is

$$\Phi(0,0) = I \qquad \text{A.6}$$

The solution for the state transition matrix, when $\Omega(t)$ is constant, is

$$\Phi(t,0) = e^{-\Omega t} = e^{-\frac{\Omega}{\omega}(\omega t)} = I + \frac{\Omega}{\omega}(\omega t) + (\frac{\Omega}{\omega})^2(\frac{\omega t}{2!})^2 + (\frac{\Omega}{\omega})^3(\frac{\omega t}{3!})^3 + \ldots \qquad \text{A.7}$$

where $\omega = (\omega_x^2 + \omega_y^2 + \omega_z^2)^{\frac{1}{2}}$. A.8

Noting that

$$\frac{\Omega^3}{\omega^3} = -\frac{\Omega}{\omega}; \quad \frac{\Omega^4}{\omega^4} = -\frac{\Omega^2}{\omega^2}, \text{ etc.} \qquad \text{A.9}$$

and substituting A.9 into A.7 yields

$$\Phi(t,0) = I - (\frac{\Omega}{\omega})\left[(\omega t) - \frac{(\omega t)^3}{3!} + \frac{(\omega t)^5}{5!} + \ldots\right] + (\frac{\Omega^2}{\omega^2})\left[\frac{(\omega t)^2}{2!} - \frac{(\omega t)^4}{4!} + \frac{(\omega t)^6}{6!} + \ldots\right], \qquad \text{A.10}$$

Now

$$\Phi(t,0) = I - \frac{\Omega}{\omega}[\sin \omega t] - \frac{\Omega^2}{\omega^2}\left[1 - \frac{(\omega t)^2}{2!} + \frac{(\omega t)^4}{4!} + \ldots\right] + \frac{\Omega^2}{\omega^2} \qquad \text{A.11}$$

which results in

$$\Phi(t,0) = I - \frac{\Omega}{\omega}[\sin \omega t] + \frac{\Omega^2}{\omega^2}[1 - \cos \omega t]. \qquad \text{A.12}$$

From Equation A.3, the D.C.M. is shown to be

$$C(t) = C_0 \Phi^T(t,0) \qquad \text{A.13}$$

so that

$$C(t) = C_0 [I - \frac{\Omega}{\omega} \sin \omega t + \frac{\Omega^2}{\omega^2}(1 - \cos \omega t)]^T. \qquad \text{A.14}$$

For the case where $C_0 = I$

$$C(t) = [I + \frac{\Omega}{\omega} \sin \omega t + \frac{\Omega^2}{\omega^2}(1 - \cos \omega t)]. \qquad \text{A.15}$$

For the case of coning motion superimposed upon a constant pitch rate, the body angular velocity is given by

$$\underline{\omega}^b_{ib} = \begin{Bmatrix} \alpha \sin \beta t \\ \gamma \\ \alpha \cos \beta t \end{Bmatrix} \qquad \text{A.16}$$

which can be expressed as

$$\underline{\omega}^b_{ib} = C^b_a \underline{\omega}^a_{ib} \qquad \text{A.17}$$

where

$$C^b_a(t) = \begin{bmatrix} \cos \beta t & 0 & \sin \beta t \\ 0 & 1 & 0 \\ -\sin \beta t & 0 & \cos \beta t \end{bmatrix} \qquad \text{A.18}$$

and

$$\underline{\omega}^{a}_{ib} = \begin{bmatrix} 0 \\ \gamma \\ \alpha \end{bmatrix}. \qquad \text{A.19}$$

Now

$$C^{i}_{b}(t) = C^{i}_{a}(t) C^{a}_{b}(t) \qquad \text{A.20}$$

which is the matrix of interest for this case.

It is now necessary to determine the solution for $C^{i}_{a}(t)$. From Equation A.18, the value for $\underline{\omega}^{a}_{ab}$ can be determined as

$$\underline{\omega}^{a}_{ab} = \underline{\omega}^{b}_{ab} = \begin{Bmatrix} 0 \\ -\beta \\ 0 \end{Bmatrix}. \qquad \text{A.21}$$

$\underline{\omega}^{b}_{ib}$ can be rewritten as

$$\underline{\omega}^{b}_{ib} = C^{b}_{a} \left\{ \begin{pmatrix} 0 \\ \beta + \gamma \\ \alpha \end{pmatrix} + \begin{pmatrix} 0 \\ -\beta \\ 0 \end{pmatrix} \right\} \qquad \text{A.22}$$

and because

$$\underline{\omega}^{a}_{ib} = \underline{\omega}^{a}_{ia} + \underline{\omega}^{a}_{ab} \qquad \text{A.23}$$

it must follow that

$$\underline{\omega}^{a}_{ia} = \begin{Bmatrix} 0 \\ \beta + \gamma \\ \alpha \end{Bmatrix}. \qquad \text{A.24}$$

$\underline{\omega}^{a}_{ia}$ is of constant value and

$$\dot{C}^{i}_{a}(t) = C^{i}_{a}(t)\Omega$$

where

$$\Omega = \begin{bmatrix} 0 & -\alpha & (\beta + \gamma) \\ \alpha & 0 & 0 \\ -(\beta + \gamma) & 0 & 0 \end{bmatrix}. \qquad \text{A.25}$$

The solution to A.25 is then given as

$$C^{i}_{a}(t) = [I + \frac{\Omega^2}{\omega^2}(1 - \cos \omega t) + \frac{\Omega}{\omega} \sin \omega t] \qquad \text{A.26}$$

where

$$\omega = (\alpha^2 + (\beta + \gamma)^2)^{\frac{1}{2}}. \qquad \text{A.27}$$

The value for $C^{i}_{b}(t)$ is now given as

$$C^{i}_{b}(t) = [I + \frac{\Omega^2}{\omega^2}(1 - \cos \omega t) + \frac{\Omega}{\omega}\sin \omega t]\, C^{a}_{b}(t). \qquad \text{A.28}$$

APPENDIX B.

Orthonormalization

One technique that has been suggested for orthonormalizing a computed direction cosine matrix is the following.

Letting $\hat{C}$ be the computed D.C.M. and

$$C^* = \hat{C}\ (\hat{C}^T\hat{C})^{-\frac{1}{2}} \qquad \text{B.1}$$

then C* will be the optimal orthogonal approximation which minimizes

$$\text{Trace}\ [(C^* - \hat{C})\ (C^* - \hat{C})^T]. \qquad \text{B.2}$$

Letting

$$\hat{C} = C + \delta C = (I + \delta C C^T)C \qquad \text{B.3}$$

where C is the true D.C.M., and expanding Equation B.1 to first order according to the binomial series yields the result

$$C^* = [I + \tfrac{1}{2}\,\delta C C^T - \tfrac{1}{2}\,C\delta C^T]C. \qquad \text{B.4}$$

Letting $P = \delta C C^T$ and substituting into B.4 gives

$$C^* = [I + \tfrac{1}{2}P - \tfrac{1}{2}P^T]C. \qquad \text{B.5}$$

When this first order approximation is substituted into the error equation of Chapter 4

$$E = \hat{C}\,C^T - I \qquad \text{B.6}$$

$$E = \frac{1}{2}P - \frac{1}{2}P^T. \qquad \text{B.7}$$

Note that the right hand side of equation B.7 is in skew symmetric form. This means that the error matrix would be of the following form.

$$E = \begin{pmatrix} 0 & \varepsilon_{12} & \varepsilon_{13} \\ -\varepsilon_{12} & 0 & \varepsilon_{23} \\ -\varepsilon_{13} & -\varepsilon_{23} & 0 \end{pmatrix} \qquad \text{B.8}$$

This error matrix would yield zero scale and skew errors independent of the magnitude of its elements.

Letting the difference matrix δC and the true D.C.M. be divided into row vectors

$$\delta C = \begin{pmatrix} \bar{a}_1 \\ \bar{a}_2 \\ \bar{a}_3 \end{pmatrix}; \quad \underline{C} = \begin{pmatrix} \bar{c}_1 \\ \bar{c}_2 \\ \bar{c}_3 \end{pmatrix} \qquad \text{B.9}$$

The error matrix becomes

$$E = \begin{bmatrix} 0 & \dfrac{\bar{a}_1 \cdot \bar{c}_2^T - \bar{a}_2 \cdot \bar{c}_1^T}{2} & \dfrac{\bar{a}_1 \cdot \bar{c}_3^T - \bar{a}_3 \cdot \bar{c}_1^T}{2} \\ \dfrac{a_2 \cdot c_1^T - a_1 \cdot c_2^T}{2} & 0 & \dfrac{\bar{a}_2 \cdot \bar{c}_3^T - \bar{a}_3 \cdot \bar{c}_2^T}{2} \\ \dfrac{\bar{a}_3 \cdot \bar{c}_1^T - \bar{a}_1 \cdot \bar{c}_3^T}{2} & \dfrac{\bar{a}_3 \cdot \bar{c}_2^T - \bar{a}_2 \cdot \bar{c}_3^T}{2} & 0 \end{bmatrix} \qquad \text{B.10}$$

and the three drift errors as defined in Chapter III are

$$\text{Drift}_1 = \frac{\bar{a}_2 \cdot \bar{c}_3^T - \bar{a}_3 \cdot \bar{c}_2^T}{2}$$

$$\text{Drift}_2 = \frac{\bar{a}_3 \cdot \bar{c}_1^T - \bar{a}_1 \cdot \bar{c}_3^T}{2} \qquad \text{B.11}$$

$$\text{Drift}_3 = \frac{\bar{a}_1 \cdot \bar{c}_2^T - \bar{a}_2 \cdot \bar{c}_1^T}{2}.$$

If the computed D.C.M. is not orthonormalized, the error matrix of Equation B.2 becomes

$$E = \begin{bmatrix} \bar{a}_1 \cdot \bar{c}_1^T & \bar{a}_1 \cdot \bar{c}_2^T & \bar{a}_1 \cdot \bar{c}_3^T \\ \bar{a}_2 \cdot \bar{c}_1^T & \bar{a}_2 \cdot \bar{c}_2^T & \bar{a}_2 \cdot \bar{c}_3^T \\ \bar{a}_3 \cdot \bar{c}_1^T & \bar{a}_3 \cdot \bar{c}_2^T & \bar{a}_3 \cdot \bar{c}_3^T \end{bmatrix} \qquad \text{B.12}$$

and the other errors as defined in Chapter III are as follows

$$\text{Scale}_1 = (\bar{a}_1 \cdot \bar{c}_1^T)^2 + (\bar{a}_2 \cdot \bar{c}_1^T)^2 + (\bar{a}_3 \cdot \bar{c}_1^T)^2$$

$$\text{Scale}_2 = (\bar{a}_1 \cdot \bar{c}_2^T)^2 + (\bar{a}_2 \cdot \bar{c}_2^T)^2 + (\bar{a}_3 \cdot \bar{c}_2^T)^2$$

$$\text{Scale}_3 = (\bar{a}_1 \cdot \bar{c}_3^T)^2 + (\bar{a}_2 \cdot \bar{c}_3^T)^2 + (\bar{a}_3 \cdot \bar{c}_3^T)^2$$

$$\text{Skew}_1 = \frac{\bar{a}_2 \cdot \bar{c}_3^T + \bar{a}_3 \cdot \bar{c}_2^T}{2} \qquad \text{B.13}$$

$$\text{Skew}_2 = \frac{\bar{a}_1 \cdot \bar{c}_3^T + \bar{a}_3 \cdot \bar{c}_1^T}{2}$$

$$\text{Skew}_3 = \frac{\bar{a}_1 \cdot \bar{c}_2^T + \bar{a}_2 \cdot \bar{c}_1^T}{2}$$

$$\text{Drift}_1 = \frac{\bar{a}_2 \cdot \bar{c}_3^T - \bar{a}_3 \cdot \bar{c}_2^T}{2}$$

$$\text{Drift}_2 \quad \frac{\bar{a}_3 \cdot \bar{c}_1^T - \bar{a}_1 \cdot \bar{c}_3^T}{2}$$

$$\text{Drift}_3 \quad \frac{\bar{a}_1 \cdot \bar{c}_2^T - \bar{a}_2 \cdot \bar{c}_1^T}{2}.$$

The drift errors of Equation B.13 are seen to be identical to those of Equation B.10. Therefore, an exact first order orthonormalization routine does not yield any benefit in reduction of drift errors. This is due to the fact that the error matrix associated with the orthonormalized D.C.M. is the skew symmetric form of the error matrix associated with the non-orthonormalized D.C.M., and drift errors are defined as the skew symmetric portions of the error matrix. Obviously, skew symmetritizing an already skew symmetric matrix does not change it.

Therefore, first order orthonormalization routines will effectively null scale and skew errors. However, they do not minimize B.2 as much as would a higher order expansion.

A second order expansion of Equation B.1 results in a C* of the form

$$C^* = [I - \tfrac{1}{2}P^T + \tfrac{1}{2}P - \tfrac{1}{8}P^TP - \tfrac{1}{8}PP^T + \tfrac{3}{8}P^TP^T - \tfrac{1}{2}P^2]\ C \qquad \text{B.14}$$

where again $P = \delta CC^T$.

The error matrix for this second order approximation is then

$$E = \tfrac{1}{2}P - \tfrac{1}{2}P^T - \tfrac{1}{8}P^TP - \tfrac{1}{8}PP^T + \tfrac{3}{8}P^TP^T - \tfrac{1}{2}P^2. \qquad \text{B.15}$$

This error matrix is not of skew symmetric form; and therefore, the scale and skew errors will not be zero. However, if the higher order terms of the binomial expansion are included, Equation B.2 should be more effectively minimized. This would imply that a lesser amount of change was made upon the computed direction cosine matrix to orthonormalize it.

The results of the preceeding analysis show scale and skew errors to be second order, while drift errors are first order with respect to the errors found in the computed direction cosine matrix.

```
C       MAIN PROGRAM                                                                    MAIN0001
        IMPLICIT REAL*8(A-G,O-Z)                                                        MAIN0002
        COMMON  TINT,NCNT                                                               MAIN0003
        COMMON /ABC/ PLOT,PPRINT                                                        MAIN0004
        COMMON /CONE/ ALPHA,BETA,GAMMA                                                  MAIN0005
        DIMENSION BC(3,3),BT(3,3),DB(3,3),DE(3,3),DTHETA(3),E(9)                        MAIN0006
        DIMENSION NT(2000),E1(2000),E2(2000),E3(2000)                                   MAIN0007
        DIMENSION E4(2000),E5(2000),E6(2000),E7(2000),E8(2000),E9(2000)                 MAIN0008
        DIMENSION HEAD(108)                                                             MAIN0009
        DATA  MARK/ 1H* /                                                               MAIN0010
        IREAD = 5                                                                       MAIN0011
        IWRITE = 6                                                                      MAIN0012
C------INITIALIZE PROGRAM PARAMETERS                                                   MAIN0013
        PI=3.141592653589732                                                            MAIN0014
        RDTODG = 180.0/PI                                                               MAIN0015
        ALPHA = 1.0                                                                     MAIN0016
        BETA=1.0                                                                        MAIN0017
        GAMMA = 1.0                                                                     MAIN0018
        DGTORD = PI/180.                                                                MAIN0019
        ALPHA=ALPHA*DGTORD                                                              MAIN0020
        BETA=BETA*DGTORD                                                                MAIN0021
        GAMMA=GAMMA*DGTORD                                                              MAIN0022
C------READ SIX JOB DESCRIPTION CARDS                                                  MAIN0023
        READ(5,3)  (HEAD(I),I=1,108)                                                    MAIN0024
    3 FORMAT(18A4/18A4/18A4/18A4/18A4/18A4)                                             MAIN0025
        WRITE  (IWRITE,2)                                                               MAIN0026
 2      FORMAT (1H1,40X,35HGSSA2 - ATTITUDE SIMULATION PROGRAM//                        MAIN0027
     1 1X,15HJOB DESCRIPTION//)                                                         MAIN0028
        WRITE(6,4) (HEAD(I),I=1,108)                                                    MAIN0029
    4 FORMAT(6(/2X,18A4))                                                               MAIN0030
        DO 10 I=1,3                                                                     MAIN0031
        DO 10 J=1,3                                                                     MAIN0032
        BT(I,J) =0.0                                                                    MAIN0033
 10     BC(I,J) =0.0                                                                    MAIN0034
        BT(1,1) = 1.0                                                                   MAIN0035
        BC(1,1) = 1.0                                                                   MAIN0036
```

```
      BT(2,2) = 1.0                                                    MAIN0037
      BC(2,2) = 1.0                                                    MAIN0038
      BT(3,3) = 1.0                                                    MAIN0039
      BC(3,3) = 1.0                                                    MAIN004C
C-----READ INPUTS                                                      MAIN0041
      READ (IREAD,402)   FREQ,TPLOT,TPRINT,TF                          MAIN0042
  402 FORMAT(4F10.0)                                                   MAIN0043
      WRITE  (IWRITE,403)                                              MAIN0044
  403 FORMAT  (////1X,10HINPUT DATA)                                   MAIN0045
      WRITE(IWRITE,404)  FREQ,TPLOT,TPRINT,TF                          MAIN0046
  404 FORMAT   (//1X,6HFREQ =D15.8,4X,7HTPLOT =D15.8,4X,8HTPRINT =     MAIN0047
     1    D15.8,4X,4HTF =D15.8//)                                      MAIN0048
      TINT = 1.0/FREQ                                                  MAIN0049
      T = 0.0                                                          MAIN005C
      NT(1) = T                                                        MAIN0051
      NPLT = 1                                                         MAIN0052
      PLOT = TPLOT                                                     MAIN0053
      PPRINT = TPRINT                                                  MAIN0054
      CALL ERROR  (BC,BT,DB,DE,E,IWRITE)                               MAIN0055
      DO 20 K=4,9                                                      MAIN0056
      F(K) = E(K)*ROTODG                                               MAIN0057
   20 CONTINUE                                                         MAIN0058
      E1(NPLT) = E(1)                                                  MAIN0059
      E2(NPLT) = E(2)                                                  MAIN006C
      E3(NPLT) = E(3)                                                  MAIN0061
      E4(NPLT) = E(4)                                                  MAIN0062
      E5(NPLT) = E(5)                                                  MAIN0063
      F6(NPLT) = E(6)                                                  MAIN0064
      E7(NPLT) = E(7)                                                  MAIN0065
      E8(NPLT) = E(8)                                                  MAIN0066
      F9(NPLT) = F(9)                                                  MAIN0067
      WRITE  (IWRITE,405)                                              MAIN0068
  405 FORMAT (////1X,11HOUTPUT DATA)                                   MAIN0069
      WRITE (IWRITE,406)  T                                            MAIN007C
  406 FORMAT (///1X,6HTIME =D15.8)                                     MAIN0071
      WRITE  (IWRITE,407)                                              MAIN0072
```

```
 407  FORMAT  (//1X,37HBC - COMPUTED DIRECTION COSINE MATRIX,22X,          MAIN007
     1  33HBT - TRUE DIRECTION COSINE MATRIX/)                            MAIN007
      DO 21 I=1,3                                                        MAIN007
      WRITE  (IWRITE,408)  (BC(I,J),J=1,3), (BT(I,J),J=1,3)              MAIN007
 21   CONTINUE                                                           MAIN007
 408  FORMAT  (3D16.8,11X,3D16.8)                                        MAIN007
      WRITE  (IWRITE,409)                                                MAIN007
 409  FORMAT (//1X,22HDB - DIFFERENCE MATRIX,37X,17HDE - ERROR MATRIX/)  MAIN008
      DO 22 I=1,3                                                        MAIN008
          WRITE  (IWRITE,408)  (DB(I,J),J=1,3),  (DE(I,J),J=1,3)         MAIN008
 22   CONTINUE                                                           MAIN008
      WRITE  (IWRITE,410)                                                MAIN008
 410  FORMAT  (//1X,8HE VECTOR//6X,21HSCALE (DIMENSIONLESS),29X,         MAIN008
     1 10HSKEW (DEG),42X,11HDRIFT (DEG)/)                                MAIN008
      DO 23 I=1,3                                                        MAIN008
      WRITE  (IWRITE,411)  E(I),E(I+3),E(I+6)                            MAIN008
 23   CONTINUE                                                           MAIN008
 411  FORMAT  (7X,D16.8,29X,D16.8,36X,D16.8)                             MAIN009
C-----SET UP DO LOOP LIMIT FOR MAIN LOOP                                 MAIN009
      XINT = TF/TINT                                                     MAIN009
      INT = XINT                                                         MAIN009
      I2 = 0                                                             MAIN009
      NCNT = 1                                                           MAIN009
C-----MAIN LOOP                                                          MAIN009
 100  I2 = I2 + 1                                                        MAIN009
      CALL PULSE  (T,TINT,DTHETA)                                        MAIN009
      T = T+TINT                                                         MAIN009
      CALL ALGOR (DTHETA,BC,IWRITE)                                      MAIN010
      IF (T .LT. PLOT)   GO TO 31                                        MAIN010
      NPLT =.NPLT+1                                                      MAIN010
      NT (NPLT) = PLOT                                                   MAIN010
      PLOT = PLOT+TPLOT                                                  MAIN010
      CALL TRUTH  (T,BT)                                                 MAIN010
      CALL ERROR  (BC,BT,DB,DE,E,IWRITE)                                 MAIN010
      DO 32 K=4,9                                                        MAIN010
      E(K) = E(K)*ROTODG                                                 MAIN010
```

```
 32   CONTINUE                                                          MAIN0109
      E1(NPLT) = E(1)                                                   MAIN0110
      E2(NPLT) = E(2)                                                   MAIN0111
      E3(NPLT) = E(3)                                                   MAIN0112
      E4(NPLT) = E(4)                                                   MAIN0113
      E5(NPLT) = E(5)                                                   MAIN0114
      E6(NPLT) = E(6)                                                   MAIN0115
      E7(NPLT) = E(7)                                                   MAIN0116
      E8(NPLT) = E(8)                                                   MAIN0117
      E9(NPLT) = E(9)                                                   MAIN0118
   31 IF(T.LT.PPRINT) GO TO 131                                         MAIN0119
      APLT=PLOT-TPLOT                                                   MAIN0120
      IF(T.GE.APLT) GO TO 132                                           MAIN0121
      CALL TRUTH(T,BT)                                                  MAIN0122
      CALL ERROR(BC,BT,DB,DE,E,IWRITE)                                  MAIN0123
  132 CONTINUE                                                          MAIN0124
      PPRINT = PPRINT+TPRINT                                            MAIN0125
C-----WRITE OUTPUT                                                      MAIN0126
      WRITE (IWRITE,406)  T                                             MAIN0127
      WRITE  (IWRITE,407)                                               MAIN0128
      DO 24 I=1,3                                                       MAIN0129
      WRITE  (IWRITE,408)  (BC(I,J),J=1,3), (BT(I,J),J=1,3)             MAIN0130
 24   CONTINUE                                                          MAIN0131
      WRITE  (IWRITE,409)                                               MAIN0132
      DO 25 I=1,3                                                       MAIN0133
      WRITE  (IWRITE,408)  (DB(I,J),J=1,3),  (DE(I,J),J=1,3)            MAIN0134
 25   CONTINUE                                                          MAIN0135
      WRITE  (IWRITE,410)                                               MAIN0136
      DO 26 I=1,3                                                       MAIN0137
      WRITE  (IWRITE,411)  E(I),E(I+3),E(I+6)                           MAIN0138
 26   CONTINUE                                                          MAIN0139
  131 CONTINUE                                                          MAIN0140
      IF (I2 .LT. INT)   GO TO 100                                      MAIN0141
C-----READ SIX PLOT DESCRIPTION CARDS                                   MAIN0142
      READ (IREAD,3)  (HEAD(I),I=1,72)                                  MAIN0143
C-----PRINT PLOT DESCRIPTION HEADING                                    MAIN0144
```

```
      WRITE  (IWRITE,5)                                                  MAIN0145
5     FORMAT (1H1,1X,16HPLOT DESCRIPTION//)                              MAIN0146
      WRITE(IWRITE,404)  FREQ,TPLOT,TPRINT,TF                            MAIN0147
      WRITE (IWRITE,4)  (HEAD(I),I=1,72)                                 MAIN0148
      WRITE (IWRITE,420)                                                 MAIN0149
      CALL PLOTER  (E1,NT,NPLT,MARK,IWRITE)                              MAIN0150
      WRITE (IWRITE,421)                                                 MAIN0151
      CALL PLOTER  (E2,NT,NPLT,MARK,IWRITE)                              MAIN0152
      WRITE (IWRITE,422)                                                 MAIN0153
      CALL PLOTER  (E3,NT,NPLT,MARK,IWRITE)                              MAIN0154
      WRITE (IWRITE,423)                                                 MAIN0155
      CALL PLOTER  (E4,NT,NPLT,MARK,IWRITE)                              MAIN0156
      WRITE (IWRITE,424)                                                 MAIN0157
      CALL PLOTER  (E5,NT,NPLT,MARK,IWRITE)                              MAIN0158
      WRITE (IWRITE,425)                                                 MAIN0159
      CALL PLOTER  (E6,NT,NPLT,MARK,IWRITE)                              MAIN0160
      WRITE (IWRITE,426)                                                 MAIN0161
      CALL PLOTER  (E7,NT,NPLT,MARK,IWRITE)                              MAIN0162
      WRITE (IWRITE,427)                                                 MAIN0163
      CALL PLOTER  (E8,NT,NPLT,MARK,IWRITE)                              MAIN0164
      WRITE (IWRITE,428)                                                 MAIN0165
      CALL PLOTER  (E9,NT,NPLT,MARK,IWRITE)                              MAIN0166
420   FORMAT  (1H1,45X,50HX-AXIS SCALE ERROR (DIMENSIONLESS) VS. TIME (S MAIN0167
     1EC.))                                                              MAIN0168
421   FORMAT  (1H1,45X,50HY-AXIS SCALE ERROR (DIMENSIONLESS) VS. TIME (S MAIN0169
     1EC.))                                                              MAIN0170
422   FORMAT  (1H1,45X,50HZ-AXIS SCALE ERROR (DIMENSIONLESS) VS. TIME (S MAIN0171
     1EC.))                                                              MAIN0172
423   FORMAT  (1H1,45X,40HX-Y SKEW ERROR (DEGREES) VS. TIME (SEC.))      MAIN0173
424   FORMAT  (1H1,45X,40HY-Z SKEW ERROR (DEGREES) VS. TIME (SEC.))      MAIN0174
 425  FORMAT  (1H1,45X,40HZ-X SKEW ERROR (DEGREES) VS. TIME (SEC.))      MAIN0175
426   FORMAT(1H1,45X,45HDRIFT ERROR ABOUT X (DEGREES) VS. TIME (SEC.))   MAIN0176
427   FORMAT(1H1,45X,45HDRIFT ERROR ABOUT Y (DEGREES) VS. TIME (SEC.))   MAIN0177
428   FORMAT(1H1,45X,45HDRIFT ERROR ABOUT Z (DEGREES) VS. TIME (SEC.))   MAIN0178
      STOP                                                               MAIN0179
      END                                                                MAIN0180
```

```
      SUBROUTINE ERROR  (BC,BT,DB,DE,E,IWRITE)                          ERR00001
      IMPLICIT REAL*8(A-G,O-Z)                                          ERR00002
      DIMENSION  BC(3,3),BT(3,3),E(9),DE(3,3),BTT(3,3)                  ERR00003
      DIMENSION  DB(3,3)                                                ERR00004
C-----COMPUTE ERROR MATRIX                                              ERR00005
      CALL MATXT  (BT,BTT)                                              ERR00006
      CALL MATMLT  (BC,3,3,BTT,3,3,DE,IWRITE)                           ERR00007
      DO 10 I=1,3                                                       ERR00008
 10   DE(I,I) = DE(I,I) - 1.0                                           ERR00009
      DO 20  I=1,3                                                      ERR00010
      DO 20  J=1,3                                                      ERR00011
      DB(I,J) = BC(I,J) - BT(I,J)                                       ERR00012
 20   CONTINUE                                                          ERR00013
      E(1) = DE(1,1)                                                    ERR00014
      E(2) = DE(2,2)                                                    ERR00015
      E(3) = DE(3,3)                                                    ERR00016
      E(4) = (DE(2,1)+DE(1,2))/2.0                                      ERR00017
      E(5) = (DE(2,3)+DE(3,2))/2.0                                      ERR00018
      E(6) =    (DE(3,1)+DE(1,3))/2.0                                   ERR00019
      E(7) = (DE(2,3)-DE(3,2))/2.0                                      ERR00020
      E(8) = (DE(3,1)-DE(1,3))/2.0                                      ERR00021
      E(9) = (DE(1,2)-DE(2,1))/2.0                                      ERR00022
      RETURN                                                            ERR00023
      END                                                               ERR00024
```

```
      SUBROUTINE  PLOTER (PLOT,NT,NUMBR,MARK,IWRITE)                    PLOT0001
      IMPLICIT REAL*8(A-G,O-Z)                                          PLOT0002
      INTEGER BLANK,DOT,PLUS                                            PLOT0003
      DATA BLANK / 1H  /                                                PLOT0004
      DATA DOT / 1H. /                                                  PLOT0005
      DATA PLUS / 1H+ /                                                 PLOT0006
      DIMENSION PLOT(NUMBR),NT(NUMBR),LINE(103)                         PLOT0007
      ABS(X)=DABS(X)                                                    PLOT0008
      DMAX=ABS(PLOT(1))                                                 PLOT0009
      DO 5 I=1,NUMBR                                                    PLOT0010
      XDATA=ABS(PLOT(I))                                                PLOT0011
      IF(DMAX-XDATA)4,5,5                                               PLOT0012
4     DMAX=XDATA                                                        PLOT0013
5     CONTINUE                                                          PLOT0014
      FACTR = DMAX                                                      PLOT0015
      IF(FACTR.EQ.0.0) GO TO 13                                         PLOT0016
      SPACE = 50.0 / FACTR                                              PLOT0017
      HDG = FACTR                                                       PLOT0018
      WRITE  (IWRITE,102)  HDG                                          PLOT0019
102   FORMAT  (/50X,17HSCALE (0 TO 10) = 1PD10.1//                      PLOT0020
     1 36X,5HMINUS,55X,4HPLUS/)                                         PLOT0021
      DO 1 I=1,103                                                      PLOT0022
1     LINE(I)=DOT                                                       PLOT0023
      LINE (1)  = 1                                                     PLOT0024
      LINE (2)  = 0                                                     PLOT0025
      LINE (7)  = 9                                                     PLOT0026
      LINE (12) = 8                                                     PLOT0027
      LINE (17) = 7                                                     PLOT0028
      LINE (22) = 6                                                     PLOT0029
      LINE (27) = 5                                                     PLOT0030
      LINE (32) = 4                                                     PLOT0031
      LINE (37) = 3                                                     PLOT0032
      LINE (42) = 2                                                     PLOT0033
      LINE (47) = 1                                                     PLOT0034
      LINE (52) = 0                                                     PLOT0035
      LINE (57) = 1                                                     PLOT0036
```

```
      LINE (62) = 2                                                          PLOT0037
      LINE (67) = 3                                                          PLOT0038
      LINE (72) = 4                                                          PLOT0039
      LINE (77) = 5                                                          PLOT0040
      LINE (82) = 6                                                          PLOT0041
      LINE (87) = 7                                                          PLOT0042
      LINE (92) = 8                                                          PLOT0043
      LINE (97) = 9                                                          PLOT0044
      LINE (102)= 1                                                          PLOT0045
      LINE (103)= 0                                                          PLOT0046
      WRITE (IWRITE,100)  LINE                                               PLOT0047
  100 FORMAT  (20X,2I1,4A1,I1,4A1,I1,4A1,I1,4A1,I1,4A1,I1,4A1,I1,4A1,        PLOT0048
     1  I1,4A1,I1,4A1,I1,4A1,I1,4A1,I1,4A1,I1,4A1,I1,4A1,I1,4A1,             PLOT0049
     2I1,4A1,I1,4A1,I1,4A1,I1,4A1,2I1)                                       PLOT0050
      DO 3 I=1,103                                                           PLOT0051
    3 LINE(I)=BLANK                                                          PLOT0052
      LINE(1)=DOT                                                            PLOT0053
      LINE(52)=DOT                                                           PLOT0054
      LINE(103)=DOT                                                          PLOT0055
      DO 13 I=1,NUMBR                                                        PLOT0056
      J=SPACE*PLOT(I)+52.5                                                   PLOT0057
      LINE(J)=MARK                                                           PLOT0058
      WRITE (IWRITE,101)  NT(I),LINE                                         PLOT0059
      LINE(J)=BLANK                                                          PLOT0060
      LINE(52)=DOT                                                           PLOT0061
   13 CONTINUE                                                               PLOT0062
  101 FORMAT(9X,I10,1X,103A1)                                                PLOT0063
      RETURN                                                                 PLOT0064
      END                                                                    PLOT0065
```

```
      SUBROUTINE MATMLT  (A,N1,N2,B,M1,M2,C,IWRITE)                          MATM0001
      IMPLICIT REAL*8(A-G,O-Z)                                               MATM0002
      DIMENSION  A(3,3), B(3,3), C(3,3)                                      MATM0003
C                                                                            MATM0004
      IF (N2- M1) 10,20,10                                                   MATM0005
   10 WRITE (IWRITE,1)              N1, N2, M1, M2,                          MATM0006
     1  ((A(I,J),J=1,N2), I=1,N1) , ((B(I,J),J=1,M2),I=1,M1)                 MATM0007
    1  FORMAT (1H1  26H MATRIX MULTIPLY ROUTINE          //                 MATM0008
     1   25H INCOMPATIBILITY ERROR         3H A(, I5, 1H, I5, 1H),  3X ,     MATM0009
     2   3H B(, I5, 1H, I5, 1H)   //  18H A MATRIX BY ROWS   /               MATM0010
     3   18H B MATRIX BY ROWS   //  (6D20.8)    )                           MATM0011
C                                                                            MATM0012
   20 DO 30 I=1,N1                                                           MATM0013
      DO 30 J=1,M2                                                           MATM0014
      C(I,J) = 0.0                                                           MATM0015
      DO 40 K=1,N2                                                           MATM0016
   40 C(I,J)=C(I,J) + A(I,K)*B(K,J)                                          MATM0017
   30 CONTINUE                                                               MATM0018
      RETURN                                                                 MATM0019
      END                                                                    MATM0020
```

```
      SUBROUTINE MATXT(A,AT)                                            MATX0001
      IMPLICIT REAL*8(A-G,O-Z)                                          MATX0002
      DIMENSION A(3,3) , AT(3,3)                                        MATX0003
      DO 1 I=1,3                                                        MATX0004
      DO 1 J=1,3                                                        MATX0005
      AT(I,J)=A(J,I)                                                    MATX0006
1     CONTINUE                                                          MATX0007
      RETURN                                                            MATX0008
      END                                                               MATX0009
```

```
      SUBROUTINE VECMLT(A,N1,N2,B,C)                                    VECM0000
      IMPLICIT REAL*8(A-G,O-Z)                                          VECM0000
      DIMENSION A(4,4),B(4),C(4)                                        VECM0000
      DO 10 I=1,N1                                                      VECM0000
      C(I)=0.0                                                          VECM0000
      DO 10 J=1,N2                                                      VECM0000
   10 C(I)=C(I) + A(I,J)*B(J)                                           VECM0000
      RETURN                                                            VECM0000
      END                                                               VECM0000
```

```
      SUBROUTINE ORTHO(B)                                                ORTH0001
      IMPLICIT REAL*8(A-G,O-Z)                                           ORTH0002
      DIMENSION B(3,3),C(3,3)                                            ORTH0003
      SQRT(X)=DSQRT(X)                                                   ORTH0004
      C(1,1)=B(2,2)*B(3,3)-B(3,2)*B(2,3)                                 ORTH0005
      C(1,2)=B(3,1)*B(2,3)-B(2,1)*B(3,3)                                 ORTH0006
      C(1,3)=B(2,1)*B(3,2)-B(2,2)*B(3,1)                                 ORTH0007
      C(3,1)=B(1,2)*B(2,3)-B(2,2)*B(1,3)                                 ORTH0008
      C(3,2)=B(2,1)*B(1,3)-B(1,1)*B(2,3)                                 ORTH0009
      C(3,3)=B(1,1)*B(2,2)-B(1,2)*B(2,1)                                 ORTH0010
      C(2,1)=B(3,2)*B(1,3)-B(1,2)*B(3,3)                                 ORTH0011
      C(2,2)=B(1,1)*B(3,3)-B(3,1)*B(1,3)                                 ORTH0012
      C(2,3)=B(3,1)*B(1,2)-B(3,2)*B(1,1)                                 ORTH0013
      DO 1 I=1,3                                                         ORTH0014
      DO 1 J=1,3                                                         ORTH0015
    1 B(I,J)=B(I,J)+C(I,J)                                               ORTH0016
      DO 2 I=1,3                                                         ORTH0017
      D=SQRT(B(I,1)**2+B(I,2)**2 + B(I,3)**2)                            ORTH0018
      DO 2 J=1,3                                                         ORTH0019
      B(I,J)=B(I,J)/D                                                    ORTH0020
    2 CONTINUE                                                           ORTH0021
      RETURN                                                             ORTH0022
      END                                                                ORTH0023
```

```
      SUBROUTINE TRUTH  (T,BT)                                          TRUT0001
C        SINGLE AXIS CONSTANT-RATE                                      TRUT0002
      IMPLICIT REAL*8(A-G,O-Z)                                          TRUT0003
      DIMENSION BT(3,3), A(5), TM(6)                                    TRUT0004
      SIN(X)=DSIN(X)                                                    TRUT0005
      COS(X)=DCOS(X)                                                    TRUT0006
      SQRT(X)=DSQRT(X)                                                  TRUT0007
      PI=3.141592653589732                                              TRUT0008
      DGTORD = PI/180.0                                                 TRUT0009
      THETAN = 0.0                                                      TRUT0010
      A(1)=10.0                                                         TRUT0011
      A(2) = 0.0                                                        TRUT0012
      A(3) = 0.0                                                        TRUT0013
      A(4) = 0.0                                                        TRUT0014
      A(5) = 0.0                                                        TRUT0015
      TM(1) = T                                                         TRUT0016
      DO 7 I=1,5                                                        TRUT0017
      XI = I                                                            TRUT0018
      THETAN =  (A(I)/XI) * TM(I)  + THETAN                             TRUT0019
      TM(I+1) = TM(I)*TM(1)                                             TRUT0020
 7    CONTINUE                                                          TRUT0021
      DO 10 I=1,3                                                       TRUT0022
      DO 10 J=1,3                                                       TRUT0023
      BT(I,J) = 0.0                                                     TRUT0024
 10   CONTINUE                                                          TRUT0025
      THETAN = THETAN*DGTORD                                            TRUT0026
      ST = SIN(THETAN)                                                  TRUT0027
      CT = COS(THETAN)                                                  TRUT0028
      BT(2,2) = 1.0                                                     TRUT0029
      BT(1,1) = CT                                                      TRUT0030
      BT(3,1) = -ST                                                     TRUT0031
      BT(3,3) = BT(1,1)                                                 TRUT0032
      BT(1,3) = -BT(3,1)                                                TRUT0033
      RETURN                                                            TRUT0034
      END                                                               TRUT0035
```

```
      SUBROUTINE  PULSE  (T,TINT,DTHETA)                                    PULS0001
      IMPLICIT REAL*8(A-G,O-Z)                                               PULS0002
C-----GENERATES PULSES FOR THE POLYNOMIAL INPUT CASE -- SINGLE AXIS          PULS0003
      DIMENSION  DTHETA(3),A(4)                                              PULS0004
      PI=3.141592653589732                                                   PULS0005
      DGTORD = PI/180.0                                                      PULS0006
      A0=10.0                                                                PULS0007
      A(1) = 0.0                                                             PULS0008
      A(2) = 0.0                                                             PULS0009
      A(3) = 0.0                                                             PULS0010
      A(4) = 0.0                                                             PULS0011
      DTHETA(1) = 0.0                                                        PULS0012
      DTHETA(3) = 0.0                                                        PULS0013
      DT1 = TINT                                                             PULS0014
      DT2 = TINT**2                                                          PULS0015
      DT3 = TINT**3                                                          PULS0016
      DT4 = TINT**4                                                          PULS0017
      T1 = T                                                                 PULS0018
      T2 = T**2                                                              PULS0019
      T3 = T**3                                                              PULS0020
      T4 = T**4                                                              PULS0021
      PART1 = A0+A(1)*(T1+0.5*DT1)                                           PULS0022
      PART2 = A(2)*(T2+T1*DT1+DT2/3.0)                                       PULS0023
      PART3 = A(3)*(T3+1.5*T2*DT1+T1*DT2+.25*DT3)                            PULS0024
      PART4 = A(4)*(T4+2.0*T3*DT1+2.0*T2*DT2+T1*DT3+.2*DT4)                  PULS0025
      DTHETA(2) = (PART1+PART2+PART3+PART4)*DT1*DGTORD                       PULS0026
      RETURN                                                                 PULS0027
      END                                                                    PULS0028
```

```
      SUBROUTINE  ALGOR  (DTHETA,BC,IWRITE)                                  ALG00001
C  DIRECTION COSINE MATRIX, FOURTH ORDER                                     ALG00002
      IMPLICIT REAL*8(A-G,O-Z)                                               ALG00003
      COMMON  TINT,NCNT                                                      ALG00004
      COMMON /ABC/ PLOT,PPRINT                                               ALG00005
      DIMENSION  DTHETA(3),BC(3,3),UNITY(3,3),DTHET1(3)                      ALG00006
      DIMENSION DTHET2(3),THMAT1(3,3),THMAT2(3,3),TEMP1(3,3),OMG2(3,3)       ALG00007
      DIMENSION OMG0(3,3),TEMP2(3,3)                                         ALG00008
      DIMENSION OMG3(3,3),BN(3,3),CN(3,3),DN(3,3)                            ALG00009
      SIN(X)=DSIN(X)                                                         ALG00010
      SQRT(X)=DSQRT(X)                                                       ALG00011
      COS(X)=DCOS(X)                                                         ALG00012
      DT = TINT/2.0                                                          ALG00013
      GO TO (100,200),  NCNT                                                 ALG00014
100   T= 0.0                                                                 ALG00015
      NCNT = 2                                                               ALG00016
 200  CALL PULSE (T,DT,DTHET1)                                               ALG00017
      T = T + TINT                                                           ALG00018
      DO 300 I=1,3                                                           ALG00019
 300  DTHET2(I) = DTHETA(I) - DTHET1(I)                                      ALG00020
      DO 305 I=1,3                                                           ALG00021
      THMAT1(I,I) = 0.0                                                      ALG00022
      THMAT2(I,I) = 0.0                                                      ALG00023
 305  CONTINUE                                                               ALG00024
      THMAT1(1,2) = -DTHET1(3)                                               ALG00025
      THMAT1(2,3) = -DTHET1(1)                                               ALG00026
      THMAT1(3,1) = -DTHET1(2)                                               ALG00027
      THMAT1(1,3) = DTHET1(2)                                                ALG00028
      THMAT1(2,1) = DTHET1(3)                                                ALG00029
      THMAT1(3,2) = DTHET1(1)                                                ALG00030
      THMAT2(1,2) = -DTHET2(3)                                               ALG00031
      THMAT2(2,3) = -DTHET2(1)                                               ALG00032
      THMAT2(3,1) = -DTHET2(2)                                               ALG00033
      THMAT2(1,3) = DTHET2(2)                                                ALG00034
      THMAT2(2,1) = DTHET2(3)                                                ALG00035
      THMAT2(3,2) = DTHET2(1)                                                ALG00036
```

```
      DO 310  I=1,3                                                          ALG00037
      DO 310  J=1,3                                                          ALG00038
      OMG0(I,J) =  (3.0*THMAT1(I,J)-THMAT2(I,J))/TINT                        ALG00039
      OMG2(I,J) = (3.0*THMAT2(I,J)-THMAT1(I,J))/TINT                         ALG00040
      OMG3(I,J)=(THMAT1(I,J)+THMAT2(I,J))/TINT                               ALG00041
  310 CONTINUE                                                               ALG00042
      CALL MATMLT(BC,3,3,OMG0,3,3,TEMP1,IWRITE)                              ALG00043
      DO 500 I=1,3                                                           ALG00044
      DO 500 J=1,3                                                           ALG00045
      TEMP1(I,J)=TEMP1(I,J)*TINT                                             ALG00046
  500 TEMP2(I,J)=TEMP1(I,J)/2.0 + BC(I,J)                                    ALG00047
      CALL MATMLT(TEMP2,3,3,OMG3,3,3,BN,IWRITE)                              ALG00048
      DO 501 I=1,3                                                           ALG00049
      DO 501 J=1,3                                                           ALG00050
      BN(I,J)=BN(I,J)*TINT                                                   ALG00051
  501 TEMP2(I,J)=BN(I,J)/2.0 + BC(I,J)                                       ALG00052
      CALL MATMLT(TEMP2,3,3,OMG3,3,3,CN,IWRITE)                              ALG00053
      DO 502 I=1,3                                                           ALG00054
      DO 502 J=1,3                                                           ALG00055
      CN(I,J)=CN(I,J)*TINT                                                   ALG00056
  502 TEMP2(I,J)=CN(I,J)      + BC(I,J)                                      ALG00057
      CALL MATMLT(TEMP2,3,3,OMG2,3,3,DN,IWRITE)                              ALG00058
      DO 503 I=1,3                                                           ALG00059
      DO 503 J=1,3                                                           ALG00060
      DN(I,J)=DN(I,J)*TINT                                                   ALG00061
  503 BC(I,J)=BC(I,J) + TEMP1(I,J)/6.0 + BN(I,J)/3.0 + CN(I,J)/3.0 +         ALG00062
     1 DN(I,J)/6.0                                                           ALG00063
      IF(T.GE.PLOT) GO TO 888                                                ALG00064
      IF(T.GE.PPRINT) GO TO 888                                              ALG00065
      RETURN                                                                 ALG00066
  888 CALL ORTHO(BC)                                                         ALG00067
      RETURN                                                                 ALG00068
      END                                                                    ALG00069
```