NASA

Technical Memorandum 80546

A DIGITAL COMPUTER PROGRAM FOR THE DYNAMIC INTERACTION SIMULATION OF CONTROLS AND STRUCTURE (DISCOS)

VOLUME IV, SUPPLEMENTARY DOCUMENTATION

Harold P. Frisch

(NASA-TM-80546) A DIGITAL COMPUTER PROGRAM FOR THE DYNAMIC INTERACTION SIMULATION OF CONTROLS AND STRUCTURE (DISCOS). VOLUME 4: SUPPLEMENTARY DOCUMENTATION (NASA) 53 p HC A04/MF A01 CSCL 22B G3/18

N80-13150

Unclas 42000

AUGUST 1979

National Aeronautics and Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

A Digital Computer Program for the Dynamic Interaction Simulation of Controls and Structure (DISCOS)

Volume IV, Supplementary Documentation

Harold P. Frisch

August 1979

Previous Publications:

Volume I:	NASA Technical Paper No. 1219 Volume I, May 1978
Volume II:	NASA Technical Paper No. 1219 Volume II, May 1978
Volume III:	Demonstration Problems, distributed by COSMIC

GODDARD SPACE FLIGHT CENTER
Greenbelt, Maryland 20771

All measurement values are expressed in the International System of Units (SI) in accordance with NASA Policy Directive 2220.4, paragraph 4.

A DIGITAL COMPUTER PROGRAM FOR THE
DYNAMIC INTERACTION SIMULATION OF
CONTROLS AND STRUCTURE (DISCOS)
VOLUME IV, SUPPLEMENTARY DOCUMENTATION

Harold P. Frisch

ABSTRACT

Volumes I, II and III contain the complete documentation for the program DISCOS distributed through Computer Software Management and Information Center (COSMIC). Since the time of original release, several additions have been made to the program. These provide an added measure of user convenience and versatility, and enhance computational speed and accuracy. Furthermore, a complete set of twelve demonstration problems has been run with the updated version (referred to as DISCOS2), and the output has been generated in a form convenient for distribution to potential users. This volume briefly defines the additions made to the original version.

CONTENTS

ILLUSTRATIONS

PRECEDING PAGE BLANK NOT FILMED

CONTENTS (continued)

TABLES

A DIGITAL COMPUTER PROGRAM FOR THE DYNAMIC INTERACTION SIMULATION OF CONTROLS AND STRUCTURE (DISCOS)

INTRODUCTION

The original purpose of this volume was to effectively reproduce and add to the contents of Volume III which is distributed via COSMIC. Time constraints have made this impossible; consequently, this volume draws heavily from Volume III and provides only supplemental comments where applicable. Demonstration problem 12 (not in Volume III) is defined, and a few brief remarks are made pertaining to the new subroutines.

As with the work contained in Volumes I, II, and III, a preponderance of the theoretical work is a direct result of the research efforts of Carl S. Bodley of the Martin Marietta Corporation.

1. DEMONSTRATION PROBLEMS

1.1 DEFINITION AND FORMAT

All Demonstration (DEMO) problems, with the exception of DEMO problem No. 12, are defined in Volume III. These problems have been rerun with the new DISCOS2 program and crosschecked with the results obtained via the original DISCOS program.

For distribution purposes a DEMO problem output tape has been created. It has the following format:

- File 1:
  A listing of all default subroutines used for DEMO problems 1 through 12. In the JCL deck the object files for these subroutines are contained in DISKLIB DSN=NFHPF.DISCOS2. DEMO.DEFAULT. In each of the DEMO problems, source decks are read in which override one or more of these routines.

- File 2:
  A listing of the complete input deck used at NASA/GSFC to run DEMO problem No 1, followed directly by a complete listing of all line printer output.

- File 3 through 13:
  A listing of the complete input deck for DEMO problems 2 through 12, followed directly by a complete listing of all line printer output.

A partial listing of this DEMO tape can be found in the microfiche section of this volume. It contains the first thousand lines of data in each of the thirteen files. (If not attached, the microfiche is available through COSMIC, the University of Georgia, Athens, Georgia 30601.)

Many of these DEMO problems have exercised the DISCOS plot capability. A complete set of all generated plots can also be found in the microfiche section.

1.2 RERUNNING DEMO PROBLEMS

The following comments pertain to the rerunning of all DEMO problems:

1.2.1 DEMO Problems 1, 2, 3, 4, 5, and 7

DEMO problems 1, 2, 3, 4, 5, and 7 are in near-perfect agreement. Minor deviations in numerical magnitude of integrated quantities are explained by recognizing that an upgrade numerical integration subroutine is used in DISCOS2.

1.2.2 DEMO Problem No. 6

For DEMO problem No. 6, a one-for-one crosscheck with data in Volume III is not possible. The algorithm FINDU used to generate Volume III data has been changed. Now the algorithm is biased to choose a set of independent state variables which more closely correspond to the set that the analyst would normally choose based on physical interpretation considerations. However, the net results for this problem agree with those published in Volume III.

1.2.3 DEMO Problem No. 8

In the case of DEMO problem No. 8, some minor changing of output format for all frequency domain studies has been introduced into DYNS40 and associated subroutines. Results obtained agree perfectly with those obtained via DISCOS and provided in Volume III.

1.2.4 DEMO Problem No. 9

DEMO problem No. 9 has been, by far, the most difficult to numerically solve. Poles and zeros are very close and there is a several order of magnitude spread in system frequencies along with numerous system roots equal to zero. Determination of transfer function zeros has been the crux of the problem because the highest roots of the system are *numerically* near or beyond infinity.

To choose the best zero finding technique for DISCO2, the following methods have been coded and tested: those of Brockett (Reference 1), Sandberg and So (Reference 2), Kaufman (Reference 3), Davison (Reference 4), the original DISCOS routine (Reference 5), and a routine analogous to original DISCOS but with improved eiginanalysis and matrix inversion capability. For numerically simple problems, all methods gave identical results. For DEMO No. 9, they all failed in varying degrees.

In essence, each method is required to pick either a very large or a very small number. This chosen number is then used as a measure for determining the order of the numerator polynomial. When the zeros of the polynomial are a large magnitude, it becomes numerically difficult to distinguish between a *zero* and a root at infinity. For DFMO No. 9, the method now in DISCOS2 did the best job. The improved eiginanalysis and inversion routines seem to have added a measure of numerical error control which makes it superior to the original DISCOS method and far superior to the other methods tested.

Another problem with DEMO No. 9 was the fact that certain zeros and poles were approximately equal. For different choices in tolerance factors, these may or may not cancel.

While DEMO problem No. 9 has proven to be extremely difficult to handle numerically, it has proven to be an excellent problem for forcing the incorporation of first class numerical techniques.

1.2.5 DEMO Problem No. 10

In DEMO problem No. 10 (as in No. 6) FINDU picks a different set of independent state variables. However, plotted output data for particular variables agree.

1.2.6 DEMO Problem No. 11

As with DEMO No. 8, there is minor output data format change from Volume III for problem No. 11. However, the results agree.

1.2.7 DEMO No. 12, Appendage Deployment Example

The purpose of DEMO problem No. 12 is to show the methodology used to implement an appendage deployment problem from initial release through lockup. This is one of the more difficult problems to numerically simulate because the total number of system independent degrees of freedom can change each time an appendage hinge release or latch condition is computed.

The problem to be simulated is a simplification of one studied for the SCATHA satellite. It consists of a nonsymmetrical central rigid body with two unequal folded two-segment experiment booms. The spacecraft spins and it is important to determine system attitude dynamics during the entire boom deployment/latch sequence.

1.3 CONFIGURATION DESCRIPTION, DEMO No. 12

A simplified model of the SCATHA spacecraft with the two booms partially deployed is shown in figure 1.

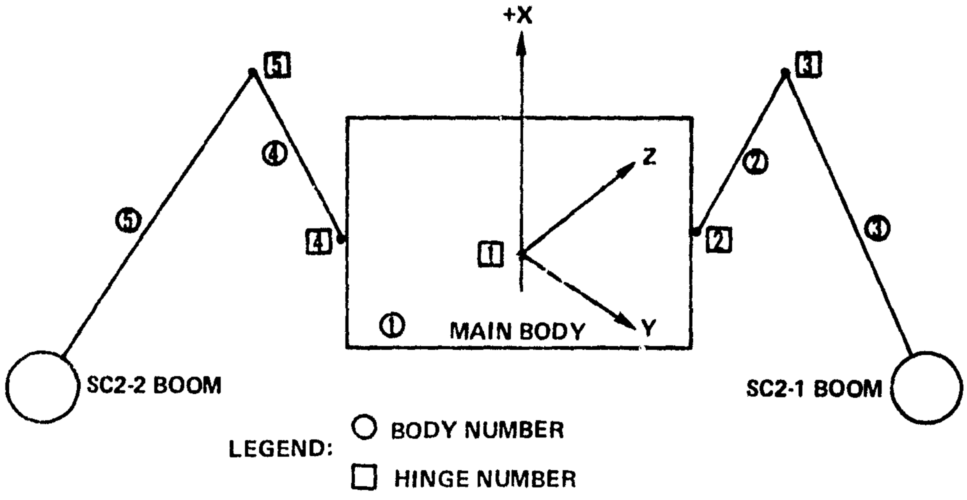

Figure 1. Partially Deployed SCATHA Model

The relative orientation of all body fixed reference frames in the fully deployed state is shown in figure 2.

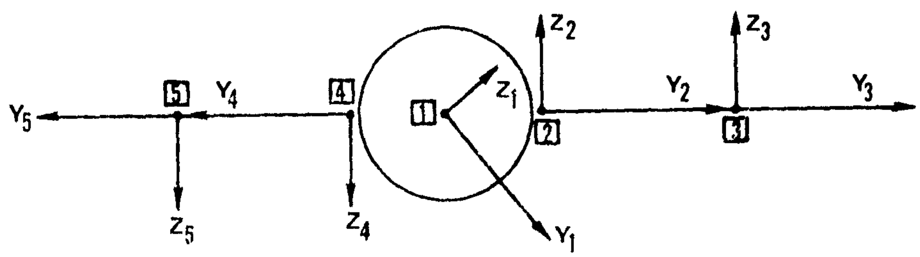

Figure 2. Fully Deployed SCATHA Model, Relative Orientation of Body Fixed Reference Frames

All body mass and inertia data is provided in Table 1 and all hinge point location data is provided in Table 2. Vector and tensor components are relative to their respective body fixed reference frame and are in the inch-pound-second system of units. The body 2 and body 4 fixed reference frames are rotated 57.0939° and 237.0939° respectively from the body 1 fixed reference about the X_1 axis. The body 3 and body 5 reference frames are parallel to the body 2 and body 4 frames respectively.

Table 1. Inertia Data

	Mass	J_{xx}	J_{yy}	J_{zz}	J_{xy}	J_{xz}	J_{xy}	S_x	S_y	S_z
Body 1	1.890	1390.0	1168.0	1216.0	−.40	1.210	−14.43			
Body 2	6.026E-3	7.353	0.0001	7.353					−.1549	
Body 3	2.538E-2	80.99	0.0001	80.99					−1.364	
Body 4	6.026E-3	7.353	0.0001	7.353					−.1549	
Body 5	2.460E-2	78.09	0.0001	78.09					−1.364	

<u>Note</u>

Units:
Mass = lb sec^2/in
Inertia = lb sec^2 in
Mass Moment = lb sec^2
Location = in

Table 2. Hinge Point Location Data

Hinge	Body	X	Y	Z
2	1	1.610	18.72	28.93
4	1	1.610	18.86	28.93
2	2	1.250		
3	2	1.250	56.25	
3	3	1.250		
4	4	1.250		
5	4	1.250	56.25	
5	5	1.250		

Note

Units:
Mass = lb sec^2/in
Inertia = lb sec^2 in
Mass Moment = lb sec^2
Location = in

1.4 INITIAL CONDITIONS, DEMO No. 12

The body 1 fixed reference is assumed to have six degrees of freedom relative to an inertial reference which is coincident with the body 1 reference frame, at time zero. Initially body 1 is spinning at 3.5 rpm about the $+X_1$ body axis.

One degree of rotational freedom is allowed at each shoulder hinge (2 and 4) and at each elbow hinge (3 and 5). At each hinge point, positive rotation is measured about a unit vector in the direction of the respective + Z body axis. Let

β_2 rotational angle at shoulder hinge 2 for SC2-1 boom

β_3 = rotation angle at elbow hinge 3 for SC2-1 boom

β_4 = rotation angle at shoulder hinge 4 for SC2-2 boom

β_5 = rotation angle at elbow hinge 5 for SC2-2 boom

at time zero

$$\beta_2(0) = -91.5^\circ$$
$$\beta_3(0) = +180^\circ$$
$$\beta_4(0) = -91.5^\circ$$
$$\beta_5(0) = +180^\circ$$

and all respective rates are zero.

1.5 CONSTRAINT CONDITIONS, SPRING AND DAMPING TORQUES, DEMO No. 12

The deployment/latch sequence at each shoulder hinge (2 and 4) is simulated by a rheonomic condition of constraint: specifically, a fixed constraint is in effect prior to shoulder release and, after the latching condition, is satisfied. During the period of deployment, the single axis angular motion at each shoulder hinge is free and subject only to spring and damping torques.

The deployment/latch sequence at each elbow hinge (1 and 3) is simulated by a nonlinear spring effect. The nonlinear spring attempts to model the spring powered deployment up to latching and then to model the dynamics of first mode boom flexibility after latching.

1.5.1 Nonlinear Spring Torque During Deployment:

During deployment, the net spring torque at each hinge is the result of two distinct effects: a power spring mechanism designed to ensure proper deployment, and the nonlinear elastic characteristics of the experiment wire bundle contained in each boom.

Let

$$\theta_i = \frac{180}{2\pi} \, |\beta_{i+1}| = \text{deployment angle at hinge point i+1 (degrees)}$$

Then the equations used to define the spring torque associated with the power springs are:

- Shoulder hinge power spring torque (i = 1, 3):

$$TPW_i = 3.0 * (.5860 + .01\ \theta_i)$$

- Elbow hinge power spring torque (i = 2, 4):

$$TPW_i = 3.0 * (2.9333 + .01\ \theta_i)$$

The spring torque associated with the wire bundle in each boom bent at the hinge points must be determined via experimentation. As a first approximation, the following equations are used:

- Shoulder hinge, wire bundle spring torque (i = 1, 3):

$$TBN_i = \frac{1.2 * 2.81}{90}\ \theta_i$$

- Elbow hinge, wire bundle spring torque (i = 2, 4):

$$TBN_i = \begin{cases} .8 * (\ 1.44 + \frac{1.68}{40}\theta_i) & 0 \leq \theta_i < 34.29^\circ \\ 1.2 * (\ 1.44 + \frac{1.68}{40}\theta_i) & 34.29^\circ \leq \theta_i < 40^\circ \\ 1.2 * [1.24 + \frac{.92}{40}(\theta_i - 40)] & 40^\circ \leq \theta_i < 80^\circ \\ 1.2 * [1.16 + \frac{2.97}{100}(\theta_i - 80)] & 80^\circ \leq \theta_i < 180^\circ \end{cases}$$

1.5.2 Nonlinear Spring Torque for Partially Deployed Case

If a hinge has latched, the elastic characteristics of the situation are changed. The problem no longer is a deployment problem but an elastic vibration problem, and the constants of the system must be adjusted accordingly:

- Shoulder hinge latched (i = 1, 3):

$$TLT_i = 1950. * \theta_i$$

- Elbow hinge latched (i = 2, 4) with associated shoulder hinge also latched:

$$TLT_i = 105. * \theta_i$$

- Elbow hinge latched (i = 2, 4) with associated shoulder hinge not latched:

$$TLT_i = 256 * \theta_i$$

- Rhenomic Constraint Conditions at Shoulder Hinge (i = 1, 3):

  Deployment design conditions must be simulated: these dictate that the shoulder hinge will be restrained until the respective elbow hinge deployment has proceeded through an angle of 22.4°.

  Shoulder hinge deployment can proceed after this point; however, there is a hard stop which prevents the shoulder angle from becoming less than −91.5° and physical considerations dictate that even though deployment is possible, it will not take place if the constraint torque is positive. Each of these conditions are checked in subroutine SET I. It should be noted that latching in the present context does not imply that motion is totally restrained: it implies a change in stiffness characteristics from a spring deployment to an elastic vibration situation.

- Damping torque during and after deployment:

  During deployment, damping torques will be assumed to be negligible so that a worse case situation can be studied. After latching viscous damping will be assumed and taken to be:

  (1) 12.5 in lbs sec/rad for shoulder hinges and

  (2) 5.5 in lbs sec/rad for elbow hinges.

  All other damping effects will be assumed negligible and ignored.

2. IMPROVED CAPABILITY

2.1 DEF6 SUBROUTINE

Continuing with the practice of putting the greater bulk of the documentation into the DISCOS source file, subroutine DEF6 has been written and included in DISCOS2. A copy of this routine is shown in Appendix 1. Theoretical background along with guides for user implementation are included in this subroutine.

2.2 RANDOM COMMENT FOR ANALYSTS

- Eigenvector computation, subroutine EIGAVV sets up the calling sequence for the computation of all eigenvalues and eigenvectors of a real nonsymmetric matrix. In general, the system matrix will be both defective and derogatory, i.e., there will be repeated roots and a full set of linearly independent eigenvectors will not exist. The EISPAC routines do not recognize this and hence blindly generate a full set of eigenvectors. The eigenvectors which contain extremely large numbers are the ones which are linearly dependent upon others. Since eigenvectors are only used herein as a data interpretation aid, no attempt is made to compute eigenvectors of grade greater than one. For further comments, see References 6 and 7.

- Don't expect a one-for-one exact crosscheck for DEMO No. 9. There are several sections of the output which are extremely sensitive to machine precision.

- The EISPAC eigenanalysis routines contain a machine precision number MACHEP = 2^{-52}. This should be changed if the machine is not IBM-360.

- Eigenvectors are used to provide a measure of how much each state variable contributes to the system response at the associated eigenfrequency.

- Please call H. P. Frisch NASA/GSFC if you find any programming bugs or if you are having implementation problems.

- Don't change the numerical integration algorithm unless you thoroughly understand logic used in implementing unilateral constraints.

- Subroutine LINEAR has two "small numbers," EPS1 and EPS2, defined via a DATA statement. These are used in the algorithm as a test for convergence. To date these values have worked satisfactorily in all applications; however, it would be better if these small numbers could be made to be functions of the machine precision number MACHEP used in the EISPAC routines.

- For large order time domain simulations, it is advisable to go through the linearization procedure and to obtain eigenvectors for the composite system. These results are useful as a time history data interpretation aid.

- LINPACK is a collection of FORTRAN subroutines which analyze and solve various systems of simultaneous linear algebraic equations. I have not been able to review these extensively; however they perhaps can be used to upgrade several of the DISCOS subroutines which do the same task.

REFERENCES

1. Brockett, R. W., "Poles, Zeros and Feedback: State Space Interpretation," *IEEE Trans Auto Control*, Vol. No. AC-10, pp. 129-135, April 1965.

2. Sandberg, I. W. and H. C. So, "Two-Sets-of-Eigenvalues Approach to the Computer Analysis of Linear Systems," *IEEE Trans Circuit Theory*, Vol. No. CT-16, pp. 509-517, November 1969.

3. Kaufman, I., "On Poles and Zeros of Linear Systems," *IEEE Trans Circuit Theory*, Vol. No. CT-20, No. 2., March 1973.

4. Davison, E. J., "A Computational Method for Finding the Zeros of a Multivariable Linear Time Invariant System," *Autermatica*, Vol. 6, pp. 481-484, 1920.

5. Bodly, C. S., A. A. Devers, A. C. Park and H. P. Frisch, "A Digital Computer Program for the Dynamic Interaction Simulation of Controls and Structure (DISCOS)," NASA Technical Paper 1219, Vol. I and II, May 1978.

6. Golub, G. H. and J. H. Wilkinson, "Ill-Conditioned Eigensystems and the Computation of of the Jordan Canonical Form," *SIAM Review*, Vol. 18, No. 4, pp. 578-619, October 1976.

7. Pease, M. C., *Methods of Matrix Algebra*, Academic Press, 1965.

8. Dongarra, J. J., J. R. Bunch, C. B. Moler, and G. W. Stewart, *LINPACK User's Guide*, SIAM, Phila., 1979.

APPENDIX

```
      SUBROUTINE DEF6
C                                                                  DEBUG # 127
C
C                        ******************************
C                        ***                        ***
C                        ***        DISCOS-2        ***
C                        ***     DOCUMENTATION      ***
C                        ***                        ***
C                        ******************************
C
C
C       DISCOS-2 IS THE FIRST MAJOR UPDATE OF THE PROGRAM 'DISCOS'
C                ALL UPDATES ARE MADE TO INCREASE COMPUTATION
C                SPEED, EFFICIENCY OF OPERATION AND MINIMIZATION
C                OF USER INTERFACE REQUIREMENTS
C
C       ALL DISCOS COMMENTS CONTAINED IN SUBROUTINES DEF1, DEF2, DEF3,
C                DEF4 AND DEF5  REMAIN VALID UNLESS AMMENDED BY COMMENTS
C                IN THIS SUBROUTINE
C
C       MAJOR CHANGES FROM ORIGINAL DISCOS PROGRAM:
C             1) COMPUTE ATTITUDE DYNAMICS RELATIVE TO ACCELERATING
C                FRAME OF REFERENCE, IF DESIRED (DEMBRT)
C             2) CORRECT DEFICIENCIES IN OLD GRAVITY GRADIENT
C                CAPABILITY (GRVGRD)
C             3) ALLOW FOR THE INPUT OF A CRUDE ESTIMATE OF FLEXIBLE
C                BODY DATA (MODALF)
C             4) IMPROVE ACCURACY OF TRANSFER FUNCTION ZERO FINDING
C                ALGORITHM (NIMS)
C             5) PROVISION FOR INCLUDING ORBIT EQUATIONS (ORBIT)
C             6) ADD IMPROVED VARIABLE STEP INTEGRATION WHICH YIELDS
C                ESTIMATE OF SYSTEM STATE AT T+H BEFORE THE LAST CYCLE OF
C                THE INTEGRATION STEP (RKADAM)
C             7) INCLUDE QUATERNION TECHNIQUE FOR FINDING HINGE AND
C                SENSOR POINT TRANSFORMATION MATRICES (ROTTRQ)
C             8) ALLOW FOR UNILATERAL CONSTRAINTS, THE NUMBER OF ACTUAL
C                DEGREES OF FREEDOM MAY BE TIME DEPENDENT, IMPORTANT FOR
C                DEPLOYMENT FROM RELEASE THROUGH LOCKUP (SETI)
C             9) IMPROVE CONTROL OF NUMERICAL ERROR IN FUNCTION
C                COMPUTATION FOR NON-LINEAR TIME DOMAIN ANALYSIS (YMOT)
C            10) INCORPORATE STATE OF THE ART EIGENANALYSIS ROUTINES
C                REFERENCE: 'MATRIX EIGENSYSTEM ROUTINES - EISPACK GUIDE'
C                           LECTURE NOTES IN COMPUTER SCIENCE
C                           VOLUMN 6  (SEE ALSO VOL 51)
C                           SPRINGER-VELAG, 1976
```

```
C                 BALANC, ORTHES, ORTRAN, HQR, HQR2, BALBAK
C
C                THE FOLLOWING SUBROUTINES AND COMMON BLOCKS IN DISCOS-2
C                     ARE EITHER NEW OR HAVE BEEN MODIFIED
C
C***********************************************************************
C
C
C      COMMON /COMPCS/
C         USED IN
C                     COMPIN
C                     YDOT
C                     PRNTOU
C         PARAMETERS
C         XAMU    = 6X6 INERTIA MATRIX FOR BODY N AT TIME T, AS USED
C                     IN EQUATION II-86
C         AJVAL   = SYSTEM PRINCIPAL MOMENTS OF INERTIA RELATIVE TO CM
C         AJVEC   = ROTATION TRANSFORMATION INERTIAL TRIAD TO PRINCIPAL
C                     INERTIA TRIAD
C         AJC     = SYSTEM INERTIA TENSOR REL TO CM, WRT INERTIAL FRAME
C         DOC     = SYSTEM CM LOCATION WRT INERTIAL FRAME
C
C***********************************************************************
C
C      COMMON /GRATIO/
C         USED IN
C                     (DELETED FROM DISCOS-2)
C
C***********************************************************************
C
C      COMMON /WRKTA/
C         USED IN
C                     (DELETED FROM DISCOS-2)
C
C***********************************************************************
C
C      COMMON /SAVLAM/
C         USED IN
C                     YDOT
C                     SETI
C         PARAMETERS
C         ALAMS    = ESTIMATE FOR VALUES OF THE LAGRANGE
C                     MULTIPLIERS AT THE END OF THE INTEGRATION STEP.
C                     COMPUTED IN YDOT, LOADED ON SECOND PASS THRU RKADAP
C
```

```
C***********************************************************************
C
C     SUBROUTINE ASQR                    DEBUG # 118
C        CHANGES
C                    (DELETED FROM DISCOS-2)
C        REASON
C                    EIGENANALYSIS NOW DONE VIA EISPAC ROUTINES
C
C***********************************************************************
C
C     SUBROUTINE ASIMLR                  DEBUG # 65
C        CHANGES
C                    REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C***********************************************************************
C
C     SUBROUTINE BAKSLV                  DEBUG # 52
C        CHANGES
C                    REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C***********************************************************************
C
C     SUBROUTINE BALANC                  DEBUG # 131
C                                              EIGENANALYSIS - EISPAC
C        CALLS
C                    NONE
C        CALLED BY
C                    EIGAVV
C                    EIGVAL
C        PURPOSE
C                    EISPAC ROUTINE TO BALANCE A REAL MATRIX AND
C                    ISOLATE EIGENVALUES BY ROW AND COLUMN PERMUTATIONS
C                    WHENEVER POSSIBLE
C
C***********************************************************************
C
C     SUBROUTINE BALBAK                  DEBUG # 136
C                                              EIGENANALYSIS - EISPAC
C        CALLS
C                    NONE
C        CALLED BY
C                    EIGAVV
C        PURPOSE
C                    EISPAC ROUTINE TO BACK TRANSFORM THE EIGENVECTORS OF
C                    THAT REAL MATRIX TRANSFORMED BY BALANC
```

```
C
C*************************************************************************
C
C      SUBROUTINE BOUTOP               DEBUG # 51
C          CHANGES
C                      REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C*************************************************************************
C
C      SUBROUTINE BBGENR               DEBUG # 45
C          CHANGES
C                      REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C*************************************************************************
C
C      SUBROUTINE BSGENR               DEBUG # 47
C          CHANGES
C                      REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C*************************************************************************
C
C      SUBROUTINE COMPIN               DEBUG # 124
C                                                MAIN LINE OPERATION
C          CALLS
C                      MULTAD     BTABA
C                      SKEWV3     EIGN1
C                      MULT3      WRITE
C          CALLED BY
C                      YDOT
C          PURPOSE
C                      TO COMPUTE TOTAL SYSTEM CENTER OF MASS LOCATION
C                      AND INERTIA TENSOR RELATIVE TO THE SYSTEM CM, WRT
C                      THE INERTIAL FRAME. ALSO TO FIND PRINCIPAL MOMENTS
C                      OF INERTIA AND ASSOCIATED ROTATION TRANSFORMATION.
C                      RESUTS STORED IN /COMPOS/ FOR EVENTUAL OUTPUT
C                      IN PRNTOU
C
C*************************************************************************
C
C      SUBROUTINE CRET3                DEBUG # 17
C          CHANGES
C                      REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C*************************************************************************
C
```

```
C      SUBROUTINE DFORMB               DEBUG * 80
C         CHANGES
C                    REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C***********************************************************************
C
C      SUBROUTINE DLMBRT               DEBUG * 120
C                                                USER DEFINED SUBROUTINE
C         CALLS
C                    (NONE UNLESS USER REQUESTED)
C         CALLED BY
C                    YDOT
C                    (ANY REQUIRED BY USER)
C         PURPOSE
C                    TO PROVIDE THE CAPABILITY TO COMPUTE ATTITUDE
C                    DYNAMICS RELATIVE TO AN ACCELERATING REFERENCE
C                    POINT. THIS CAPABILITY REQUIRED TO CIRCUMVENT
C                    NUMERICAL PROBLEMS.  DLMBRT ASSUMES THAT THE ORIGIN
C                    OF TH P-FRAME AT HINGE POINT 1 IS THE COMPOSITE
C                    SYSTEM CENTER OF MASS AND THAT ITS MOTION RELATIVE
C                    TO A TRUE INERTIAL REFERENCE IS USER DEFINABLE
C                    EITHER IN CLOSED FORM OR VIA INTEGRATION OF ITS
C                    EQUATIONS OF MOTION
C         REASON
C                    TO AVOID THE STANDARD NUMERICAL COMPUTATION PROBLEMS
C                    ASSOCIATED WITH ADDITION AND SUBTRACTION OF LARGE
C                    AND SMALL NUMBERS IT IS OFTEN DESIRABLE TO DEFINE
C                    SYSTEM ATTITUDE DYNAMICS RELATIVE TO AN ACCELERATING
C                    FRAME OF REFERENCE.
C
C                    THIS PROBLEM USUALLY OCCURS WHEN THE SYSTEM IS ACTED
C                    UPON BY A UNIDIRECTIONAL EXTERNAL FORCE FIELD AND
C                    THE PRIMARY GOAL OF THE SIMULATION IS THE ASSESSMENT
C                    OF ATTITUDE DYNAMICS, RATHER THAN TRANSLATIONAL
C                    DYNAMICS
C
C   ***
C   ****    USER CODEING RULES AND MODELLING CONSIDERATION
C   ***
C
C               THE DISCOS FORMALISM IS BASED UPON A LAGRANGE FORMULATION
C               OF THE EQUATIONS OF MOTION. INHERENT IN THE DEVELOPEMENT
C               IS THE DEFINITION THAT BOTH LINEAR AND ANGULAR VELOCITIES
C               ARE MEASURED RELATIVE TO AN INERTIALLY FIXED REFERENCE.
C
```

```
C               THE LOCATION OF THE COMPOSITE SYSTEM CENTER OF MASS
C               CANNOT ALWAYS BE TAKEN TO BE NEAR THE INERTIAL REFERENCE.
C               THE LOCATION IS DICTATED BY THE SIMULATION REQUIREMENTS.
C               IN SOME CASES THE INERTIAL ORIGIN MUST BE AT A GRAVITY
C               SOURCE, IN OTHER CASES LARGE RESULTANT EXTERNAL FORCES
C               PRODUCE LARGE RELATIVE DISPLACEMENTS WHICH MAY OR MAY NOT
C               BE IMPORTANT TO MONITOR.
C
C               IF THE INERTIAL ORIGIN IS OR BECOMES FAR REMOVED FROM THE
C               SYSTEM CENTER OF MASS NUMERICAL PROBLEMS CAN BE
C               ENCOUNTERED; THESE PROBLEMS ARE INHERENT IN DIGITAL
C               COMPUTATION WHENEVER VERY LARGE AND VERY SMALL NUMBERS
C               MUST BE ACCURATELY KNOWN DURING COMPUTATION.
C
C         PROCEDURE
C               1) COMPUTE THE RESULTANT EXTERNAL FORCE ACTING ON THE
C                  SYSTEM.
C
C               2) IF THE MOTION OF THE SYSTEM CENTER OF MASS RELATIVE
C                  TO THE INERTIAL REFERENCE IS IMPORTANT SET UP THESE
C                  EQUATIONS FOR INTEGRATION VIA SUBROUTINE CONTROL.
C                  FREQUENTLY THESE ARE THE ORBIT EQUATIONS
C
C               3) RESOLVE THE RESULTANT FORCE VECTOR INTO THE P-FRAME
C                  REFERENCE ASSOCIATED WITH HINGE POINT 1, DEFINE
C                    V(1) = X-COMPONENT OF FORCE VECTOR WRT P-FRAME
C                              AT HINGE POINT 1
C                    V(2) = Y-COMPONENT OF FORCE VECTOR WRT P-FRAME
C                              AT HINGE POINT 1
C                    V(3) = Z-COMPONENT OF FORCE VECTOR WRT P-FRAME
C                              AT HINGE POINT 1
C
C               4) RESOLVE FORCE VECTOR INTO BODY FIXED COORDINATES AND
C                  ADD APPARENT FORCE EFFECTS TO THE TORQUE ARRAY FOR
C                  EACH BODY AS FOLLOWS:
C
C
C           REAL*8 BV(3),V(3)
C     C           COMPUTE TOTAL MASS
C           AMT = 0.0
C           DO 1 N=1,NB
C         1 AMT = AMT + AMU(4,4,N)
C     C
C     C           TRANSFORM FORCE VECTOR TO BODY COORDINATES,
C     C           COMPUTE APPARENT FORCE EFFECTS,
```

```
C     C           ADD TO TORQUE ARRAY
C           DO 2 N=1,NB
C           BV(1) = ROL(1,1,N)*V(1) + ROL(2,1,N)*V(2) + ROL(3,1,N)*V(3)
C     .     BV(2) = ROL(1,2,N)*V(1) + ROL(2,2,N)*V(2) + ROL(3,2,N)*V(3)
C           BV(3) = ROL(1,3,N)*V(1) + ROL(2,3,N)*V(2) + ROL(3,3,N)*V(3)
C     C           GET STARTING LOCATION IN TORQUE ARRAY FOR BODY N
C           LOC = LOCU(N)
C     C           APPARENT TORQUE ON BODY N
C           G(LOC  ) = G(LOC  )
C          *                -(AMU(1,5,N)*BV(2) + AMU(1,6,N)*BV(3))/AMT
C           G(LOC+1) = G(LOC+1)
C          *                -(AMU(2,4,N)*BV(1) + AMU(2,6,N)*BV(3))/AMT
C           G(LOC+2) = G(LOC+2)
C          *                -(AMU(3,4,N)*BV(1) + AMU(3,5,N)*BV(2))/AMT
C     C           APPARENT FORCE ON BODY N
C           G(LOC+3) = G(LOC+3) - AMU(4,4,N)*BV(1)/AMT
C           G(LOC+4) = G(LOC+4) - AMU(4,4,N)*BV(2)/AMT
C           G(LOC+5) = G(LOC+5) - AMU(4,4,N)*BV(2)/AMT
C     C           APPARENT MODAL FORCE ON MODES OF BODY N
C           IF(IRGFLX(N).EQ.0) GO TO 2
C           DO 3 K=1,IRGFLX(N)
C           G(LOC+5+K) = G(LOC+5+K)
C          *                 -(AMU(4,6+K,N)*BV(1) +
C          *                   AMU(5,6+K,N)*BV(2) +
C          *                   AMU(6,6+K,N)*BV(3))/AMT
C         3 CONTINUE
C         2 CONTINUE
C           RETURN
C
C
C                5) NOTE THAT IT IS IMMATERIAL WHERE THE FORCE VECTOR
C                   (V(1),V(2),V(3)) IS COMPUTED. IT CAN BE COMPUTED
C                   HEREIN OR IN SUBROUTINE CONTRL, WHICH EVER IS MOST
C                   CONVIENENT FOR THE USER
C
C*****************************************************************************
C
C     SUBROUTINE DYNSIO              DEBUG # 2
C        CHANGES
C                     IF INPUT DATA CODE 'NYTPE' = 3  CALL MODALN(N)
C        REFER TO
C                     DEF3 DEFINITION OF DYNSIO INPUT DATA READ STATEMENTS
C
C     ***                                 ***
C     ****   DATA INPUT MODIFICATIONS   ****
```

```
C      ***                                        ***
C
C                 1) LAST READ STATEMENT READS INTEGER CODE NTYPE, NOW
C
C                           IF(NTYPE.EQ.3) CALL MODALN(N)
C
C                    ADDED TO DYNSIO
C
C                 2) CALL READ (TMDATA, 1, 3, 1, 3)
C                       NEW DEFINITION:
C
C                      IF(TMDATA(2).LT.0)  VARIABLE STEP RUNGE KUTTA
C                                          INTEGRATION WITH STEP SIZE
C                                           DELTAT(MAX) = ABS(TMDATA(2))
C
C                      IF(TMDATA(2).GT.0)  FIXED STEP RUNGE KUTTA
C                                          INTEGRATION WITH STEP SIZE
C                                           DELTAT = TMDATA(2)
C
C                 3) CALL READ (RV, 1, 4, 1, 5)
C
C                       VECTOR SIZE 1 BY 4 FOR GRAVITY GRADIENT DATA
C                       THIS DATA CAN BE UPDATED AND MADE TIME DEPENDENT
C                       VIA USE OF SUBROUTINE ORBIT
C
C                      IF( RV(1) = WV(2) = WV(3) = 0.0 )  DISCOS-2
C                      ASSUMES GRAVITY GRADIENT EFFECT NEGLIGIBLE
C                      AND SUBROUTINE ORBIT WILL NOT BE CALLED
C                      VIA GRVGRD
C         REASON
C                 1) AT TIMES IT IS NOT NECESSARY FOR ACCURATE SIMULATION
C                    TO INPUT A FULL FINITE ELEMENT FLEXIBLE BODY MODEL.
C                    MODALN WILL READ IN ONLY FIRST ORDER FLEXIBLE BODY
C                    MODEL. DATA REQUIRED USUALLY AVAILABLE WITHOUT
C                    TOO MUCH TROUBLE
C
C                 2) INTEGRATION PACKAGE IN DISCOS-2 HAS VARIABLE STEP
C                    CAPABILITY. ADAMS PREDICTOR/CORRECTOR CAPABILITY
C                    OF DISCOS REMOVED BECAUSE IT WAS NEVER USED AND
C                    A RUNGE KUTTA VARIABLE STEP ROUTINE WITH CERTIAN
C                    UNIQUE CHARACTERISTICS WAS NEEDED FOR UNILATERAL
C                    CONSTRAINT CAPABILITY
C
C                 3) PROCEDURE FOR COMPUTING GRAVITY GRADIENT EFFECTS
C                    UPON A CLUSTER OF COUPLED FLEXIBLE BODIES WAS
```

```
C                       LIMITED IN APPLICABILITY IN DISCOS
C
C                       NEW SUBROUTINE GRVGRD CALLS SUBROUTINE ORBIT FOR
C                       A DEFINITION OF ORBIT PARAMETERS ORIGINALLY SET
C                       BY ABOVE READ STATEMENT. NOW ELLIPTIC ORBITS AND
C                       CLUSTERS OF COUPLED FLEXIBLE BODIES CAN ROUTINELY
C                       BE ANALYZED WITH MINIMAL USER INTERFACE
C
C***********************************************************************
C
C        SUBROUTINE DYNS20              DEBUG # 37
C           CHANGE
C                       COMMON  /OPRKTA/ DELETED
C                       COMPUTATION TO LOAD ARRAYS IN /OPRKTA/ DELETED
C           REASON
C                       IN DISCOS-2 THE NEW VERSION OF SUBROUTINE RKADAM
C                       DOES NOT REQUIRE THE STORAGE AREA RESERVED BY
C                       /OPRKTA/
C
C***********************************************************************
C
C        SUBROUTINE DYNS40              DEBUG #64
C                                                     MAIN LINE OPERATION
C           CALLS (NEW)
C                       PUNCH
C                       EIGVAL
C                       EIGAVV
C           CHANGES
C                      1) CALL TO PUNCH ADDED SO THAT THE SYSTEM
C                         CHARACTERISTIC MATRIX ALONG WITH EACH TRANSFER
C                         FUNCTION CHARACTERISTIC MATRIX AND COLUMN OF INPUT
C                         SIGANL COEFFICIENTS CAN BE PUNCHED FOR FOLLOW-ON
C                         INDEPENDENT ANALYSIS
C
C            ***                                 ***
C            ****    DATA INPUT MODIFICATION    ****
C            ***                                 ***
C
C                 FIRST READ IN DYNS40 CHANGED TO
C
C-----    READ(NIT,FORMAT = 2A4)  LNAM,LPNCH
C                    LNAM = (NO CHANGE)
C                    LPNCH = 4HPNCH   IF SO, PUNCH SYSTEM AND TRANSFER
C                                     FUNCTION CHARACTERSITIC MATRICES
C                                     IF NOT, NO ACTION TAKEN
```

```
C
C                   2) SHIFT CONSTANT FOR ZERO FINDING ALGORITHM IN NUMS
C                      IS NOW SUBJECT TO INPUT CONTROL, IF DESIRED
C
C         ***                                   ***
C         ****    DATA INPUT MODIFICATION     ****
C         ***                                   ***
C
C                 THIRD READ IN DYNS40 CHANGED TO
C
C-----    CALL READIM(IRY,4,NCYC,4,KR)
C
C                       IRY(1,J) - (NO CHANGE)
C                       IRY(2,J) - (NO CHANGE)
C                       IRY(3,J) - (NO CHANGE)
C            (NEW)      IRY(4,J) = SHIFT CONSTANT FOR NUMS SET TO MINUS
C                                  SQUARE ROOT OF INPUT
C                    DEFAULT:
C                               CON = -DSQRT(3.0D0)
C                    OTHERWISE:
C                               CON = -DSQRT(1.0D0*IRY(4,J))
C
C
C
C
C                   3) REPLACE CALLS TO OLD EIGENANALYSIS ROUTINES
C
C                              QRDRVR      AND     EIGVEC
C
C                      WITH CALLS TO THE EISPAC EIGENANALYSIS ROUTINES
C
C          REASON
C                    EISPAC ROUTINES ARE CONSIDERED TO BE STATE OF THE
C                    ART, WHEREAS ORIGINAL DISCOS ROUTINES ARE ADEQUATE
C                    THEY DO NOT CONTAIN AS MUCH NUMERICAL ERROR
C                    CONTROL LOGIC IN THEIR CODING
C                                 ALSO
C                    EIGAVV COMPUTES ALL EIGENVECTORS WHEREAS EIGVEC
C                    COMPUTED ONLY SELECTED ONES AND BROKE DONE IF
C                    SYSTEM HAD REPEATED EIGENVALUES
C
C                   4) COMPUTE AND OUTPUT ALL EIGENVECTORS
C
C         ***                                   ***
C         ****    DATA INPUT MODIFICATION    ****
```

```
C          ***                                        ***
C
C                 READ RANGE FOR EIGENVECTOR
C                 COMPUTATION DELETED. IT IS
C                 COMPUTATIONALLY MORE EFFICIENT
C                 TO BLINDLY COMPUTE ALL OF THEM
C
C
C***********************************************************************
C
C      SUBROUTINE EIGAVV               DEBUG # 130
C                                                   EIGENANALYSIS - EISPAC
C         CALLS
C                       BALANC     HQR2
C                       ORTHES     WRITE
C                       ORTRAN     BALBAK
C         CALLED BY
C                       DYNS40
C         PURPOSE
C                       EISPAC DRIVER ROUTINE, TO SEQUENTIALLY CALL THE
C                       EISPAC ROUTINES REQUIRED TO COMPUTE ALL EIGENVALUES
C                       AND UNNORMALIZED EIGENVECTORS OF A REAL GENERAL
C                       MATRIX
C
C***********************************************************************
C
C      SUBROUTINE EIGN1                DEBUG # 125
C                                                 UTILITY,MATRIX OPERATION
C         CALLS
C                       WRITE
C         CALLED BY
C                       COMPIN
C         PURPOSE
C                       TO COMPUTE THE EIGENVALUES AND ASSOCIATED
C                       EIGENVECTORS OF ANY REAL SYMMETRIC MATRIX USING
C                       METHOD OF JACOBI, THRESHOLD VERSION WITH
C                       PIVOTING. COMPUTED EIGENVALUES AND EIGENVECTORS
C                       ARE REAL
C         REASON
C                       REQUIRED BY COMPIN TO COMPUTE SYSTEM PRINCIPAL
C                       MOMENTS OF INERTIA AND ASSOCIATED ROTATION
C                       TRANSFORMATION MATRIX
C
C***********************************************************************
C
```

```
C     SUBROUTINE EIGVAL              DEBUG # 129
C                                               EIGENANALYSIS - EISPAC
C        CALLS
C                     BALANC    HQR
C                     ORTHES    WRITE
C        CALLED BY
C                     DYNS40
C                     NUMS
C        PURPOSE
C                     EISPAC DRIVER ROUTINE, TO SEQUENTIALLY CALL THE
C                     EISPAC ROUTINES REQUIRED TO COMPUTE ALL EIGENVALUES
C                     OF A REAL GENERAL MATRIX
C
C**********************************************************************
C
C     SUBROUTINE EIGVEC              DEBUG # 117
C        CHANGES
C                     (DELETED FROM DISCOS-2)
C        REASON
C                     EIGENANALYSIS NOW DONE VIA EISPAC ROUTINES
C
C**********************************************************************
C
C     SUBROUTINE FINDT               DEBUG # 66
C        CHANGES
C                     REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C**********************************************************************
C
C     SUBROUTINE FINDU               DEBUG # 38
C        CHANGES
C                     REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C**********************************************************************
C
C     SUBROUTINE GRVGRD              DEBUG # 50
C                                             MAIN LINE OPERATION
C        CALLS
C                     WRITES    ORBIT
C                     MULT3
C                     MULTAD
C        CALLED BY
C                     YDOT
C        CHANGES
C                     CORRECTED TO PROPERLY DEFINE GRAVITY GRADIENT
```

```
C                  FORCE AND TORQUE ACTING UPON EACH BODY OF THE
C                  MULTI-BODY SYSTEM
C         THEORY
C                  DISCOS-2 ASSUMES THAT AN ORBITING REFERENCE FRAME
C                  WITH ORIGIN IN THE VICINITY OF THE SPACECRAFT MASS
C                  CENTER IS DEFINED. THIS REFERENCE FRAME IS FIXED
C                  INERTIALLY IN ROTATION BUT IT'S ORIGIN TRANSLATES.
C                  THE MOTION OF THE ORIGIN AND THE DIRECTION OF THE
C                  GRAVITY FIELD IS DEFINE VIA SUBROUTINE ORBIT
C              LET:
C
C               GR( ) = POSITION VECTOR FROM GRAVITY SOURCE TO THE
C                       ORIGIN OF THE ORBITING REFERENCE
C                         FROM SUBROUTINE ORBIT
C
C                         GR( ) = RCMAG*GAMGI( )
C
C              DOL(N) = POSITION VECTOR FROM ORBITING REFERENCE POINT
C                       TO THE BODY N REFERENCE POINT
C
C                  GM = EARTH'S GRAVITATIONAL CONSTANT
C
C              AMU(N) = MASS OF BODY N
C
C               GF(N) = GRAVITATIONAL FORCE VECTOR ACTING ON BODY N
C
C                GMAG = LOCAL GRAVITATIONAL ACCELERATION
C                         FROM SUBROUTINE ORBIT
C
C                         GMAG = GM/RCMAG**2
C
C              THEN THE GRAVITATIONAL FORCE VECTOR IS
C
C                                    (GR( )+DOL(N))
C               GF(N)  =  -AMU(N)*GM*--------------
C                                    |GR( )+DOL(N)|**3
C
C              EXPAND THE DENOMINATOR IN A BINOMIAL SERIES AND RETAIN
C               ONLY FIRST ORDER TERMS
C
C                |GR( )+DOL(N)|**(-3)
C
C                    =(((GR( )+DOL(N)).(GR( )+DOL(N)))**(-3.0/2.0)
C
C                    =RCMAG**(-3)*(1.0 - 3.0*GAMGI( ).DOL(N)/RCMAG)
```

```
C
C              SUBSTITUTE BACK INTO THE GRAVITATIONAL FORCE EQUATION
C               TO OBTAIN
C
C                GF(N) = -AMU(N)*GMAG*(GAMGI( ) + DOL(N)/RCMAG
C
C                          - 3.0*(GAMGI( ).DOL(N))*GAMGI( )/RCMAG
C
C                          + HIGHER ORDER TERMS )
C
C              IN THIS EQUATION THE ZERO ORDER TERM NAMELY
C
C                    -AMU(N)*GMAG*GAMGI( )
C
C              GOES INTO THE ORBIT EQUATION. THE GRAVITY GRADIENT FORCE
C              WHICH MUST BE UTILIZED IN SUBROUTINE GRVGRD IS
C
C                    -AMU(N)*GMAG*DOLL(N)
C
C              WHERE
C
C               DOLL(N) = (DOL(N) - 3*(GAMGI( ).DOL(N))*GAMGI( ))/RCMAG
C
C          FIXES
C                 THE THEORETICAL DEVELOPEMENT CAN BE FIXED BY
C                 CONSIDERING A CLUSTER OF DEFORMABLE BODIES IN
C                 FIGURE 7 PAGE 49 AND INTRODUCING THE ORBITING
C                 REFERENCE POINT
C
C                 IN VOL 1 SECTION F THE NET RESULTS ARE CORRECTED IF
C                 THE ELEMENTS OF GAMMA SUB G IN EQUATION II-120 ARE
C                 EQUATED TO THE ELEMENTS OF THE VECTOR DOLL(N)
C                 DEFINED ABOVE.
C
C                 THE SUBROUTINE GRVGRD WAS FIXED BY SIMPLY USING
C                 THE VECTOR DOLL(N) IN THE APPROPRIATE EQUATIONS
C
C**********************************************************************
C
C     SUBROUTINE LINEAR              DEBUG # 40
C         CHANGES
C                 REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C**********************************************************************
C
```

```
C      SUBROUTINE GAUSSI              DEBUG # 31
C                                                UTILITY,MATRIX OPERATION
C         CHANGE
C                    COMPLETELY REWRITTEN.
C         CALLS
C                    WRITES
C         CALLED BY
C                    MSMODC
C                    ASIMLR
C                    LTRESP
C         PURPOSE
C                    A MODIFIED ALGORITHM FOR MATRIX INVERSION VIA GAUSS
C                    ELIMINATION, ALGORTHM DOES A SEARCH FOR THE LARGEST
C                    POSSIBLE DIVISOR. THE INPUTTED N BY N MATRIX 'A' IS
C                    DESTROYED DURING COMPUTATION
C         REASON
C                    NEW METHOD FOR OBTAINING TRANSFER FUNCTION ZEROS
C                    DOES NOT REQUIRE GAUSSI FOR SUPPORTING COMPUTATION
C                    GAUSSI REWRITTEN TO REFLECT THIS AND IMPROVE
C                    COMPUTATION ALGORTHIM
C
C***********************************************************************
C
C      SUBROUTINE GETBMB              DEBUG # 48
C         CHANGES
C                    REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C***********************************************************************
C
C      SUBROUTINE HQR                 DEBUG # 134
C                                                 EIGENANALYSIS - EISPAC
C         CALLS
C                    NONE
C         CALLED BY
C                    EIGVAL
C         PURPOSE
C                    EISPAC ROUTINE TO DETERMINE THE EIGENVALUES OF A
C                    REAL UPPER HESSENBERG MATRIX
C
C***********************************************************************
C
C      SUBROUTINE HQR2                DEBUG # 135
C                                                 EIGENANALYSIS - EISPAC
C         CALLS
C                    NONE
```

```
C         CALLED BY
C                        EIGAVV
C         PURPOSE
C                        EISPAC ROUTINE TO DETERMINE ALL EIGENVALUES AND
C                        EIGENVECTORS OF A REAL UPPER HESSENBERG MATRIX
C
C************************************************************************
C
C      SUBROUTINE INVINP              DEBUG # 41
C
C         CHANGES
C                        REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C************************************************************************
C
C      SUBROUTINE LUDATF              DEBUG # 137
C                                                 UTILITY, MATRIX OPERATION
C         CALLS
C                        NONE
C         CALLED BY
C                        LUINVD
C         PURPOSE
C                        PERFORM THE L-U DECOMPOSITION OF A REAL GENERAL
C                        MATRIX WITH TEST FOR ALGORITMIC SINGULARITY USING
C                        THE CROUT ALGORITHM
C
C************************************************************************
C
C      SUBROUTINE LUINVD              DEBUG # 138
C                                                 UTILITY, MATRIX OPERATION
C         CALLS
C                        LUDATF
C         CALLED BY
C                        NUMS
C         PURPOSE
C                        TO COMPUTE FOR A REAL GENERAL MATRIX ITS DETERMINANT
C                        AND ITS INVERSE USING L-U DECOMPOSITION BY THE
C                        CROUT ALGORITHM
C
C************************************************************************
C
C      MAIN                           DEBUG # 1
C         CHANGES
C                        COMMON /DRATIO/ DELETED
C                        DISCOS SUBROUTINE LOCATION INDEX EXTENDED
```

```
C        REASON
C                    UPDATED VERSION OF SUBROUTINES GAUSSI AND NUMS
C                    NO LONGER NEED /DRATIO/ STORAGE AREA
C
C*********************************************************************
C
C     SUBROUTINE MGEN                  DEBUG # 49
C        CHANGES
C                    REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C*********************************************************************
C
C     SUBROUTINE MODALN(NBOD)          DEBUG # 126
C                                                  MAIN LINE OPERATION
C        CALLS
C                    UNITY      ZERO
C                    READ       ROTTR
C                    WRITE
C        CALLED BY
C                    DYNS10
C        PURPOSE
C                    TO SIMPLIFY THE DATA INPUT FOR FLEXIBLE BODIES WHEN
C                    THE COMBINED EFFECTS OF SPIN MAGNITUDE AND ELASTIC
C                    DEFORMATION ARE NEGLIGIBLE, THE USER INPUTS ONLY A
C                    BODY'S RIGID BODY CHARACTERISTICS, NATURAL
C                    FREQUENCIES, MODAL DAMPING, MODAL AMPLITUDE AND
C                    SLOPE AT HINGE AND SENSOR POINTS ON BODY
C
C    **                                                         ***
C    ***  DATA INPUT  (ALSO SEE COMMENT CARDS IN MODALN)       ****
C    **                                                         ***
C
C---------CALL READ (AMU, N, N, KMU, KMU)
C
C                  KMU = KMODE + 6
C                   NE = NUMBER OF ELASTIC MODES
C                    N = EITHER 6   OR   6+NE
C                  AMU = ARRAY CONTAINING ELEMENTS OF EQUATION II-39
C
C                        IF NO MODAL DATA OTHER THAN THE FREQUENCIES
C                        READ IN 6X6 RIGID BODY MASS MATRIX.
C
C                        IF MODAL DATA AVAILABLE READ IN ALL ELEMENTS
C                        OF EQUATION II-39 (6+NE) X (6+NE) MATRIX
C
```

```
C
C          _                                                       _
C         |JXX -JXY -JXZ    0 -SZ   SY DX(1)  . . .  DX(NE)   |
C         |      JYY -JYZ  SZ    0 -SX DY(1)  . . .  DY(NE)   |
C         |           JZZ -SY   SX    0 DZ(1)  . . .  DZ(NE)   |
C         |                  M    0    0 AX(1)  . . .  AX(NE)   |
C         |                       M    0 AY(1'  . . .  AY(NE)   |
C         |                            M AZ(1)  . . .  AZ(NE)   |
C   AMU = |   FULL SYMMETRIC             E(1,1) . . .  E(1,NE)  |
C         |   MUST BE READ IN                            .      |
C         |    EITHER 6X6                      .         .      |
C         |  OR (6+NE)X(6+NE)                     .      .      |
C         |                                            E(NE,NE)|
C         |_                                                  _|
C
C---------CALL READ (AMU, NE, NE, KMU, KMU)
C
C               AMU = FULL NE X NE MODAL STIFFNESS MATRIX
C                     USUALLY DIAGONAL MATRIX WITH
C                        OMEGA(N)**2, N=1,...,NE
C                     BEING DIAGONAL ELEMENTS
C
C
C---------CALL READ (AMU, NE, NE, KMU, KMU)
C
C               AMU = FULL NE X NE MODAL DAMPING MATRIX
C                     USUALLY DIAGONAL MATRIX WITH
C                       2.0*ZETA*OMEGA(N), N=1,...,NE
C                     BEING DIAGONAL ELEMENTS
C
C---------CALL READ (AMU,  1, NE, KMU, KMU)
C
C               AMU = 1 X NE MATRIX OF INITIAL MODAL DISPLACEMENT
C                     CONDITIONS
C
C---------CALL READ (AMU,  1, NE, KMU, KMU)
C
C               AMU = 1 X NE MATRIX OF INITIAL MODAL RATE
C                     CONDITIONS
C
C
C                      HINGE POINT LOOP
C
C                        NHB = NUMBER OF HINGES ON BODY NBOD
C                              (EXCLUDING HINGE POINT 1 OF BODY 1)
```

```
C
C        DO 10 II=1,NHB
C
C---------READ(NIT,FORMAT = 2I5)  NOH, ITYPE
C
C                 NOH    =  HINGE NUMBER
C                 ITYPE  =  EULER ROTATION TYPE TO ORIENT HINGE
C                           TRIAD WRT BODY TRIAD
C
C
C---------READ(NIT,FORMAT = 3E10.3)  (V(J),J=1,3)
C
C                 EULER ANGLES TO ORIENT HINGE TRIAD - PERMUTATION
C                 ORDER DEFINED BY ITYPE
C
C                 V(1) = THETA 1 (FIRST ROTATION)
C                 V(2) = THETA 2 (SECOND ROTATION)
C                 V(3) = THETA 3 (THIRD ROTATION)
C
C---------READ(NIT,FORMAT = 3E10.3) (V(J),J=1,3)
C
C                VECTOR TO POSITION HINGE TRIAD WRT BODY TRIAD
C
C                  V(1) = X (BODY REF POINT TO HINGE POINT, BODY TRIAD)
C                  V(2) = Y (BODY REF POINT TO HINGE POINT, BODY TRIAD)
C                  V(3) = Z (BODY REF POINT TO HINGE POINT, BODY TRIAD)
C
C---------CALL READ(AMU, 3, NE, KMU, KMU)
C
C                AMU = 3 X NE MATRIX OF MODAL AMPLITUDES AT HINGE POINT
C
C               AMU(1,N) = X-COMPONENT OF MODE N AT HINGE POINT
C               AMU(2,N) = Y-COMPONENT OF MODE N AT HINGE POINT
C               AMU(3,N) = Z-COMPONENT OF MODE N AT HINGE POINT
C
C---------CALL READ(AMU, 3, NE, KMU,KMU)
C
C                AMU = 3 X NE MATRIX OF MODAL SLOPES AT HINGE POINT
C
C               AMU(1,N) = THETA X ROTATION AT HINGE POINT FOR MODE N
C               AMU(2,N) = THETA Y ROTATION AT HINGE POINT FOR MODE N
C               AMU(3,N) = THETA Z ROTATION AT HINGE POINT FOR MODE N
C
C     10 CONTINUE
C
```

```
C
C                      SENSOR POINT LOOP
C
C                        NSB = NUMBER OF SENSOR POINTS ON BODY NBOD
C
C          DO 20 II=1,NSB
C
C---------READ(NIT,FORMAT = 2I5) NOS, ITYPE
C
C                   NOS    = SENSOR POINT NUMBER
C                   ITYPE = EULER ROTATION TYPE TO ORIENT SENSOR
C                           TRIAD WRT BODY TRIAD
C
C---------READ(NIT,FORMAT = 3E10.3)  (V(J),J=1,3)
C
C                   EULER ANGLES TO ORIENT SENSOR TRIAD - PERMUTATION
C                   ORDER DEFINED BY ITYPE
C
C                   V(1) = THETA 1 (FIRST ROTATION)
C                   V(2) = THETA 2 (SECOND ROTATION)
C                   V(3) = THETA 3  THIRD ROTATION)
C
C
C---------READ(NIT,FORMAT = 3E10.3) (V(J),J=1,3)
C
C                   VECTOR TO POSITION SENSOR TRIAD WRT BODY TRIAD
C
C                     V(1) = X (BODY REF POINT TO SENSOR POINT, BODY TRIAD)
C                     V(2) = Y (BODY REF POINT TO SENSOR POINT, BODY TRIAD)
C                     V(3) = Z (BODY REF POINT TO SENSOR POINT, BODY TRIAD)
C
C---------CALL READ(AMU, 3, NE, KMU, KMU)
C
C                  AMU = 3 X NE MATRIX OF MODAL AMPLITUDES AT SENSOR POINT
C
C                 AMU(1,N) = X-COMPONENT OF MODE N AT SENSOR POINT
C                 AMU(2,N) = Y-COMPONENT OF MODE N AT SENSOR POINT
C                 AMU(3,N) = Z-COMPONENT OF MODE N AT SENSOR POINT
C
C
C---------CALL READ(AMU, 3, NE, KMU, KMU)
C
C                  AMU = 3 X NE MATRIX OF MODAL SLOPES AT SENSOR POINT
C
C                 AMU(1,N) = THETA X ROTATION AT SENSOR POINT FOR MODE N
```

```
C              AMU(2,N) = THETA Y ROTATION AT SENSOR POINT FOR MODE N
C              AMU(3,N) = THETA Z ROTATION AT SENSOR POINT FOR MODE N
C
C
C       20 CONTINUE
C
C          RETURN
C          END
C
C**********************************************************************
C
C     SUBROUTINE NUMS                DEBUG #68
C
C        CHANGE
C                    UPGRADED TO MAKE USE OF STATE OF THE ART EISPAC
C                    EIGENANALYSIS AND MATRIX INVERSION ROUTINES
C        CALLS
C                    WRITE     SIFT
C                    EIGVAL
C                    LUINVD
C        CALLED BY
C                    DYNS40
C        REASON
C                    THE COMPUTATION OF TRANSFER FUNCTION ZEROS FOR LARGE
C                    ORDER SYSTEMS IS NUMERICALLY AN EXTREMELY DIFFICULT
C                    PROBLEM. EVERY METHOD WHICH CAN BE USED TO CONTROL
C                    NUMERICAL ERROR MUST BE UTILIZED. THIS IS BEING DONE
C
C**********************************************************************
C
C     SUBROUTINE ORBIT               DEBUG * 121
C                                               USER DEFINED SUBROUTINE
C        CALLS
C                    (NONE UNLESS USER REQUIRED)
C        CALLED BY
C                    GRVGRD
C                    (ANY REQUIRED BY USER)
C        PURPOSE
C                    TO DEFINE ORBITAL PARAMETERS REQUIRED TO LOCATE
C                    SPACECRAFT FROM A GRAVITATIONAL SOURCE WRT AN
C                    ORBITING REFERENCE, INERTIALLY FIXED IN ROTATION
C
C    **
C    ***  USER CODEING RULES AND MODELLING CONSIDERATIONS
C    ***
```

```
C     **
C
C
C              INORDER TO AVOID NUMERICAL COMPUTATION PROBLEMS AN
C              ORBITING REFERENCE FRAME IS USED. IT IS INERTIALLY
C              FIXED IN ROTATION AND FOR CONVIENENCE LOCATED IN THE
C              VICINITY OF THE SYSTEM CENTER OF MASS
C
C              SUBROUTINE ORBIT MUST DEFINE THE FOLLOWING PARAMETERS:
C
C                  RCMAG     = SCALAR DISTANCE FROM THE GRAVITATIONAL
C                              SOURCE TO THE ORIGIN OF THE ORBITING
C                              REFERENCE FRAME
C                  GMAG      = LOCAL GRAVITATIONAL ACCELERATION, FOR AN
C                              EARTH ORBIT
C
C                                EARTH'S GRAVITATIONAL CONSTANT
C                        GMAG = ------------------------------
C                                           RCMAG**2
C
C                  GAMGI(1) = X-COMPONENT OF THE UNIT VECTOR IN THE
C                                DIRECTION OF THE GRAVITY FIELD WRT
C                                THE ORBITING REFERENCE
C                  GAMGI(2) = Y-COMPONENT OF THE UNIT VECTOR IN THE
C                                DIRECTION OF THE GRAVITY FIELD WRT
C                                THE ORBITING REFERENCE
C                  GAMGI(3) = Z-COMPONENT OF THE UNIT VECTOR IN THE
C                                DIRECTION OF THE GRAVITY FIELD WRT
C                                THE ORBITING REFERENCE
C
C***********************************************************************
C
C      SUBROUTINE ORTHES               DEBUG # 142
C                                              EIGENANALYSIS- EISPAC
C        CALLS
C                   NONE
C        CALLED BY
C                   EIGAVV
C                   EIGVAL
C        PURPOSE
C                   EISPAC ROUTINE TO REDUCE A REAL GENERAL MATRIX TO
C                   UPPER HESSENBURG FORM USING ORTHOGONAL
C                   TRANSFORMATIONS
C
C***********************************************************************
```

```
C
C      SUBROUTINE ORTRAN              DEBUG # 133
C                                                   EIGENANALYSIS - EISPAC
C
C         CALLS
C                   NONE
C         CALLED BY
C                   EIGAVV
C         PURPOSE
C                   EISPAC ROUTINE TO ACCUMULATE THE TRANSFROMATIONS
C                   IN THE REDUCTION OF A REAL GENERAL MATRIX BY ORTHES
C
C**********************************************************************
C
C      SUBROUTINE PLOTWR              DEBUG # 61
C         CHANGE
C                   CALL POINT ADDED
C         REASON
C                   TO ACCOMODATE USER DESIRED VARIABLES TO BE PLOTTED
C                   MORE THAN JUST STANDARD SET OF TIME DEPENDENT
C                   VARIABLES CAN NOW BE PLOTTED
C
C**********************************************************************
C
C      SUBROUTINE POINT               DEBUG # 123
C                                                 USER DEFINED SUBROUTINE
C         CALLS
C                   (NONE UNLESS USER REQUESTED)
C         CALLED BY
C                   PLOTWR
C         PURPOSE
C                   TO DEFINE ANY AUXILIARY STATE VARIABLES
C                   TO BE PLOTTED AS A FUNCTION OF TIME
C
C**********************************************************************
C
C      SUBROUTINE PRNTOU              DEBUG # 60
C
C         CHANGE
C                  1) ADD COMMON BLOCK /COMPOS/
C                  2) CALLS TO INVINP DELETED. REQUIRED DATA IS NOW
C                     IN /COMPOS/
C                  3) PRINT OUT COMPOSITE SYSTEM CENTER OF MASS LOCATION
C                     AND INERTIA TENSOR DATA STORED IN /COMPOS/
C         REASON
```

```
C                       GOOD DATA GENERATED IN COMPIN, THIS IS PRINTED
C                       OUT IN THIS ROUTINE
C
C**********************************************************************
C
C       SUBROUTINE PUNCH                  DEBUG # 128
C                                                  UTILITY (I/O)
C           CALLS
C                       NONE
C           CALLED BY
C                       DYNS40
C           PURPOSE
C                       TO PUNCH A TWO DIMENSIONAL MATRIX OF REAL NUMBERS
C                       IN A FORMAT COMPATABLE WITH SUBROUTINE READ
C
C**********************************************************************
C
C       SUBROUTINE QRCON                  DEBUG # 85
C           CHANGES
C                       (DELETED FROM DISCOS-2)
C           REASON
C                       EIGENANALYSIS NOW DONE VIA EISPAC ROUTINES
C
C**********************************************************************
C
C       SUBROUTINE QRDRVR                 DEBUG # 84
C           CHANGES
C                       (DELETED FROM DISCOS-2)
C           REASON
C                       EIGENANALYSIS NOW DONE VIA EISPAC ROUTINES
C
C**********************************************************************
C
C       SUBROUTINE QR2                    DEBUG # 87
C           CHANGES
C                       (DELETED FROM DISCOS-2)
C           REASON
C                       EIGENANALYSIS NOW DONE VIA EISPAC ROUTINES
C
C**********************************************************************
C
C       SUBROUTINE RKADAM(NEQ)            DEBUG # 59
C                                                  UTILITY NON-MATRIX OPER.
C           CALLS
C                       YDOT
```

```
C          CALLED BY
C                     DYNS10
C
C          CHANGE
C                     COMPLETELY REWRITTEN TO
C                       1) ADD VARIABLE STEP INTEGRATION CAPABILITY.
C                          CALLED FOR BY INPUTTING A NEGATIVE MAXIMUM
C                          VALUE FOR STEP SIZE (TMDATA(2) IN DYNS10 INPUT)
C
C                       2) DELETE ADAMS PREDICTOR CORRECTOR CAPABILITY
C                          (TWO FIXED STEP INTEGRATION ROUTINES WERE NOT
C                           NEEDED)
C
C
C          THEORY
C                    1) IF INPUTTED INTEGRATION STEP SIZE POSITIVE
C                       STANDARD FOURTH ORDER FIXED STEP RUNGE KUTTA
C                       INTEGRATION USED
C                    2) IF NEGATIVE A SPECIALIALLY DEVELOPED VARIABLE
C                       STEP EXPLICIT, RUNGE-KUTTA ROUTINE IS USED
C
C             TO SOLVE
C
C                        YDT(T) = F(T,Y)
C
C             LET
C                        Y(T)    = STATE AT TIME T
C                        YO      = INITIAL STATE
C                        TO      = INITAL TIME
C                        F(T,Y) = OUTPUT OF SUBROUTINE YDOT FOR INPUT
C                                 TIME T AND STATE Y
C                           YDT = CALL YDOT(T,Y) = F(T,Y)
C
C             DO 5 LOOP
C                          P1 = YO
C                          P2 = F(TO,YO)
C
C         2 DO 20 LOOP; IFLAG = 2
C
C                JIL=1
CC                       Y(TO+.5*H) = YO + .5*H*P2
C                              YDT  = CALL YDOT(TO+.5H,YO+.5*H*P2)
C                JIL=2
C                          P3 = YDT = F(TO+.5*H,YO+.5*H*P2)
C                         Y(TO + H) = YO - H*P2 + 2*H*P3
```

```
C                           YDT = CALL YDOT(TO+H,YO-H*P2+2*H*P3)
C          EXIT DO 20 LOOP
C
C          ENTER SECTION TO OBTAIN MEASURE OF NUMERICAL ERROR
C            NOTE:
C                  A) SECOND ORDER ESTIMATE OF STATE AT (TO+H) IS
C
C                       Y(TO+H) = YO + H*P3 + O(H**2)
C
C                  B) THIRD ORDER ESTIMATE OF STATE AT (TO+H) IS
C
C                       Y(TO+H) = YO + H*(P2 + 4*P3 + YDT)/6 + O(H**3)
C
C            ERROR CRITERIA BASED UPON THE ABSOLUTE PERCENT DIFFERENCE
C            BETWEEN THE SECOND AND THIRD ORDER ESTIMATES OF AVERAGE
C            SLOPE OVER THE TIME INTERVAL H: THAT IS,
C
C                      P3  VS.  (P2 + 4*P3 + YDT)/6
C            LET:
C                  E = (ABSOLUTE PERCENT DIFFERENCE)/100
C            THEN
C                  H = H*DSQRT(0.07/E)
C            NOTE
C                  7 PERCENT IS THE SWITCH VALUE CHOOSEN, NUMERICAL
C                    EXPERIENCE HAS ESTABLISHED THAT THIS WORKS WELL
C            THEN
C                  WITH THE RESET VALUE OF H RETURN TO THE DO 20 LOOP
C                  WITH IFLAG=3 AND GO THROUGH A STANDARD FOURTH ORDER
C                  RUNGE KUTTA STEP, NO ERROR CHECK TO BE MADE AFTER
C                  THIS CYCLE.
C
C        2 DO 20 LOOP: IFLAG = 3
C
C             JIL=1
C                     YDT = CALL YDOT(TO+.5H,YO+.5H*P2)
C             JIL=2
C                      P3 = YDT
C                     YDT = CALL YDOT(TO+.5H,YO+.5H*P3)
C             JIL=3
C                      P4 = YDT
C                     YDT = CALL YDOT(TO+H,YO+H*P4)
C
C          END DO 20 LOOP, GO TO 45
C
C             JIL=4
```

```
C                   Y(TO+H) = YO + H*(P2 + 2*(P3+P4) + YDT)/6 + O(H**4)
C
C                       YDT = CALL YDOT(TO+H,Y(TO+H))
C
C
C         USER NOTE:
C                1) CARE WILL BE REQUIRED IN INSERTING YOUR STANDARD
C                   IN-HOUSE INTEGRATION PACKAGE. THE INTEGER 'JIL' IS
C                   USED IN THE CODE AS A LOGIC FLAG
C
C                2) QUOTE GEAR, NUMERICAL INITIAL VALUE PROBLEMS IN
C                   ORDINARY DIFFERENTIAL EQUATIONS, PAGE 40
C                     'IN GENERAL IT IS DIFFICULT TO JUSTIFY THE
C                      ADDITIONAL WORK OF THE IMPLICIT METHODS BY
C                      THE INCREASED ACCURACY ATTAINABLE, SO THEIR
C                      UTILITY IS RESTRICTED TO SOME SPECIAL PROBLEMS
C                      FOR WHICH THEY HAVE DESIRABLE STABILITY
C                      CHARACTERISTICS'
C
C                3) MULISTEP OR MULTIVALUE METHODS SHOULD BE USED WITH
C                   EXTREME CAUTION IN A GENERAL PURPOSE CONTROL SYSTEM
C                   SIMULATION PROGRAM: MORE OFTEN THAN NOT, NON-LINEAR
C                   CONTROL SYSTEM ELEMENTS WILL CAUSE SOME OF THE STATE
C                   VARIABLES TO BE DISCONTINUOUS FUNCTIONS OF TIME. FOR
C                   EXAMPLE LIMITERS DO THIS. SINCE THESE METHODS CAN
C                   BREAK DOWN ON SUCH PROBLEMS THEY SHOULD NOT BE USED.
C
C                4) EXPLICIT ONE STEP METHODS APPEAR TO BE THE ONLY
C                   LOGICAL CHOICE FOR INCLUSION IN THE GENERAL PURPOSE
C                   PROGRAM.
C
C                5) A METHOD APPLICABLE TO STIFF SYSTEMS WHICH UTILIZES
C                   PARTITIONING TO ISOLATE THE SUBSET OF STIFF STATE
C                   VARIABLE EQUATIONS WOULD BE DESIRABLE. IF AVAILABLE
C                   CALL H.P.FRISCH NASA/GSFC.
C
C************************************************************************
C
C      SUBROUTINE [illegible]                DEBUG # 44
C
C         CHANGE
C                   WHEN ACCOUNTING FOR FLEXIBILITY IN THE DETERMINATION
C                   OF ORIENTATION OF Q AND P FRAMES WRT THEIR BODY
C                   FIXED FRAMES USE SUBROUTINE ROTTRO
```

```
C         REASON
C                    ORIENTATION CHANGES ASSOCIATED WITH FLEXIBILITY ARE
C                    SMALL ANGLE, QUATERNION TECHNIQUE OF ROTTRO
C                    IS MORE EFFICIENT THAN EULER ANGLE TECHNIQUE OF
C                    ROTTR, ALSO EULER METHOD CAN LEAD TO ERRORS
C
C***********************************************************************
C
C       SUBROUTINE ROTDS                 DEBUG # 46
C
C         CHANGE
C                    WHEN ACCOUNTING FOR FLEXIBILITY IN THE DETERMINATION
C                    OF ORIENTATION OF SENSOR FRAMES WRT THE BODY FIXED
C                    FRAMES USE ROTTRO
C         REASON
C                    QUATERNION TECHNIQUE IS MORE EFFICIENT THAN EULER
C                    TECHNIQUE DUE TO SMALL ANGLE ELASTIC DEFORMATION
C                    ALSO EULER METHOD CAN LEAD TO ERRORS
C
C***********************************************************************
C
C       SUBROUTINE ROTTRO                DEBUG # 119
C                                                  MAIN LINE OPERATION
C         CALLS
C                    (NONE)
C         CALLED BY
C                    ROTDH
C                    ROTDS
C         PURPOSE
C                    TO USE EULER PARAMETER (QUATERNION) TECHNIQUE TO
C                    COMPUTE TRANSFORMATION MATRIX
C         REASON
C                    REFERENCE FRAME ORIENTATION CHANGES ASSOCIATED WITH
C                    BODY FLEXIBILITY ARE SMALL, QUATERNION TECHNIQUE
C                    IS COMPUTATIONALLY MORE EFFICIENT THAN EULER ANGLE
C                    TECHNIQUE USED ORIGINALLY IN DISCOS. ALSO EULER
C                    TECHNIQUE CAN LEAD TO ERRORS SINCE ORIENTATION
C                    CHANGE MEASURED IN DIRECTION COSINES NOT EULER
C                    ANGLES
C
C***********************************************************************
C
C       SUBROUTINE SETI                  DEBUG # 122
C                                                USER DEFINED SUBROUTINE
C         CALLS
```

```
C                   (NONE UNLESS USER REQUESTED)
C         CALLED BY
C                   YDOT
C         PURPOSE
C                   TO IMPLEMENT UNILATERAL CONSTRAINT SPECIFICATIONS
C         REASON
C                   THE EXISTANCE OF PARTICULAR CONSTRAINTS AT HINGE
C                   POINTS (DEFINED VIA IHDATA ARRAY) CANNOT ALWAYS BE
C                   TAKEN AS TIME INDEPENDENT
C
C                   FOR EXAMPLE: THE SIMULATION OF STICTION (SLIDING
C                   FRICTION), SPACECRAFT APPENDAGE DEPLOYMENT THROUGH
C                   LOCK-UP, SPRING-DASHPOT SYSTEMS WITH STOPS, ETC
C                   ALL REQUIRE THAT THE EXISTANCE OF A CONSTRAINT BE
C                   DETERMINED VIA A FUNCTION OF TIME AND SYSTEM STATE.
C
C   **
C   ***    USER CODEING RULES AND MODELLING CONSIDERATIONS
C   **
C
C              IN SUBROUTINE DYNS10 ALL CONSTRAINT CONDITIONS AT ALL
C              HINGE POINTS ARE USER DEFINED VIA INPUT DATA
C
C              SUBROUTINE SETI CAN BE USED TO EFFECTIVELY OVER-RIDE
C              THE DATA LOADED INTO THE IHDATA ARRAY
C
C              METHOD
C                   1) REFER TO SUBROUTINE DEF3, DYNS10 INPUT
C
C--------- MODIFY DEFINITION OF -------  IHDATA  ----  INTEGER ARRAY
C
C                      IHDATA(2-7,J) = 2   RHEONOMIC CONSTRAINT AT TIME
C                                             ZERO OR SOMTIME LATER
C                         ANY CONSTRAINT CONDITION SUBJECT TO CHANGE
C                         VIA SUBROUTINE SETI IS TYPE 2
C
C                   2) IN SUBROUTINE YDOT, FIRST PASS THROUGH ALL NLAM
C                      ELEMENTS OF IRH ARRAY SET TO 1
C
C                               DO 2 I=1,NLAM
C                             2 IRH(I) = 1
C
C                   3) RECALL ALL LAGRANGE MULTIPLIERS (FORCES AND
C                      TORQUES OF CONSTAINT) ARE CONSECUTIVELY
C                      NUMBERED:
```

```
C                     IV(I,L) = 0  IF DEGREE OF FREEDOM I,I=1,...,6
C                                  AT HINGE POINT L ALWAYS FREE OF
C                                  CONSTRAINT
C                             = NUMBER ASSIGNED TO LAGRANGE
C                                  MULTIPLIER ASSOCIATED WITH THE
C                                  FIXED OR RHENOMIC CONSTRAINT
C                                  WHICH IS OR WILL BE APPLIED TO
C                                  DEGREE OF FREEDOM I,I=1,....,6
C                                  AT HINGE POINT L
C                  IV STORED IN
C                   /IVCONS/
C
C              4) CONSIDER DEGREE OF FREEDOM I AT HINGE POINT L
C                 WHICH WILL AT SOME TIME BE SUBJECT TO EITHER
C                 A FIXED OR RHEONIMIC CONSTRAINT
C
C
C                 PROVIDE THE LOGIC TO DEFINE WHETHER OR NOT
C                 THE DEGREE OF FREEDOM IS FREE
C
C                 SET
C                      IRH(IV(I,L)) = 0  IF DEGREE OF FREEDOM I
C                                        AT HINGE POINT L IS FREE
C
C                      IRH(IV(I,L)) = 1  IF DEGREE OF FREEDOM I
C                                        AT HINGE POINT L IS
C                                        FIXED OR RHEONOMICALLY
C                                        CONSTRAINED
C
C                 THE LOGIC USED TO SET AND RESET THE ELEMENTS OF
C                 IRH MAY BE A FUNCTION OF TIME AND ANY SET OF
C                 SYSTEM STATE VARIABLES.
C
C              5) /SAVALM/ CONTAINS ALAMS, ARRAY OF
C                 ESTIMATES OF WHAT THE FORCES AND TORQUES OF
C                 CONSTRAINT WILL BE AT THE END OF THE
C                 INTEGRATION STEP. THESE CAN BE USE IN THE LOGIC
C                 TO DETERMINE CONSTAINT CONDITIONS FOR NEXT STEP
C
C              6) *** MAKE SURE THAT ALL CONSTRIANTS WHICH CAN BE
C                 AFFECTED BY SETI ARE OF TYPE 2 (RHEONOMIC)
C                 THIS IS REQUIRED BY COMPUTATION LOGIC. IF NOT
C                 THEN EITHER RELATIVE POSITION OR LAGRANGE
C                 MULTIPLIER WILL NOT BE CALCULATED
```

```
C***********************************************************************
C
C      SUBROUTINE SFREQ2            DEBUG # 73
C         CHANGES
C                    REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C***********************************************************************
C
C      SUBROUTINE START             DEBUG # 6
C         CHANGES
C                    REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C                    REDIMENSION CODE 486 ADDED TO FIX LIMIT ON DO LOOP
C                    READ STATEMENT FOR DEBUG FLAGS FIXED
C         REFER TO
C                    DEF3 DEFINITION OF START INPUT DATA READ STATEMENTS
C                    SECOND READ STATEMENT, NOW
C
C                       READ(NIT,FORMAT = 60L1) DEBUG
C
C         REASON
C                    SIZE OF DEBUG ARRAY HAD TO BE ENLARGED FOR DISCOS-2
C
C***********************************************************************
C
C      SUBROUTINE SUBDIA            DEBUG # 86
C         CHANGES
C                    (DELETED FROM DISCOS-2)
C         REASON
C                    EIGENANALYSIS NOW DONE VIA EISPAC ROUTINES
C
C***********************************************************************
C
C      SUBROUTINE TFTYPE            DEBUG # 67
C         CHANGES
C                    REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C***********************************************************************
C
C      SUBROUTINE TORQUE            DEBUG # 57
C         CHANGES
C                    REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C***********************************************************************
C
```

```
C       SUBROUTINE ITEF                   DEBUG # 83
C          CHANGES
C                       REDIMENSION CODE 485 ADDED FOR /LDEBUG/
C
C***********************************************************************
C
C       SUBROUTINE YDOT                   DEBUG # 39
C                                                   MAIN LINE OPERATION
C          CALLS
C                       WRITES     ROTDH      BHGENR     ROTDS
C                       BSGENR     MGEN       COMPIN     FINDU
C                       MULT3      GRVGRD     DLMBRT     INVINP
C                       GETBMB     SETI       DCOM2      ADT
C                       BAKSLV     TORQUE     BDOTOP
C          CALLED BY
C                       DYNS20
C                       RKADAM
C                       LINEAR
C          PURPOSE
C                       (SAME AS IN DISCOS) TO COMPUTE THE FIRST TIME
C                       DERIVATIVE OF THE STATE VECTOR Y
C          CHANGES
C                       MODIFIED TO USE IMPULSE/MOMENTUM CONDITIONS TO
C                       MAINTAIN THE PRESCRIBED CONSTRAINT CONDITIONS
C          REASON
C                       NUMERICAL ERRORS WHERE EXCEEDING TOLERABLE LIMITS
C                       FOR CERTIAN SIMULATIONS, A METHOD TO OVERCOME THIS
C                       DEFICIENCY HAD TO BE DEVELOPED
C          THEORY
C                       FOR SIMULATION PURPOSES NUMERICAL ERROR CAN BE
C                       VIEWED AS AN IMPLUSE, THE NET EFFECT OF THE ERROR
C                       INPLUSE IS TO CAUSE AN ASSOCIATED ERROR IN THE
C                       COMPUTATION OF THE ORDINARY MOMENTA AND SYSTEM STATE
C
C                       THIS ERROR CAN BE DETERMINED BY UTILIZATION OF THE
C                       CONSTRAINT EQUATIONS: ONCE KNOWN IT CAN BE
C                       REAPPLIED TO ARRIVE AT THE TRUE VALUE FOR CURRENT
C                       SYSTEM STATE AND ORDINARY MOMENTA
C
C                       LET
C                            P(-)      ORDINARY MOMENTA AS COMPUTED WITH
C                                      NUMERICAL ERROR
C                            U(-)      SYSTEM STATE AS COMPUTED WITH
C                                      NUMERICAL ERROR
C                            EF        ERROR IMPULSE MULTIPLIERS (TO BE FOUND)
```

```
C              P(+)     CORRECT VALUE OF ORDINARY MOMENTA
C              U(+)     CORRECT VALUE OF SYSTEM STATE
C              B        KINEMATIC COEFFICIENT MATRIX, SAME AS
C                       THAT USED IN EQUATION II-1
C              B**T     TRANSPOSE OF MATRIX B
C              B**T*EE ACTUAL ERROR IMPLUSE TO BE APPLIED TO
C                       COMPENSATE FOR NUMERICAL ERROR
C              M        MASS MATRIX AS USE IN EQUATION II-1
C
C        UTILIZING THESE DEFINITIONS THE CORRECT VALUE FOR
C        ORDINARY MOMENTA IS
C
C                P(+) = P(-) + B**T**EE                (1)
C
C        FROM EQUATION II-49
C
C                P(+OR-) = M*U(+OR-)                   (2)
C
C        THEREFORE BY DIRECT SUBSTITUTION OF (2) INTO (1)
C
C                U(+) = U(-) + M**(-1)*B**T*EE         (3)
C
C        UTILIZE THE CONSTRAINT EQUATION II-5
C
C              B*U(+) = ADT                            (4)
C
C        WHERE
C              ADT      PRESCRIBED VELOCITIES: EITHER ZERO FOR
C                       FIXED CONSTRAINTS OR USER DEFINED FOR
C                       RHEONOMIC CONSTRAINTS
C
C        SUBSTITUE (3) INTO (4) AND SOLVE FOR EE TO OBTAIN
C
C         EE = (B*M**(-1)*B**T)**(-1)*(ADT - B*U(-))         (5)
C
C        USE THIS ERROR IMPULSE MULTIPLIER (COMPUTED IN YDOT
C        BETWEEN DO 180  AND PROGRAM POINT 1305) TO COMPUTE
C        THE CORRECTED VALUES OF ORDINARY MOMENTA AND SYSTEM
C        STATE: THAT IS, IN THE DO 190 LOOP
C
C               P(+) = P(-) + B**T*EE                        (6)
C               U(+) = U(-) + B**T*M**(-1)*EE                (7)
C
C        IS EVALUATED AND CARRIED THRU ALL SUBSEQUENT
C        CALCULATIONS.
```

```
C
C                   THIS PROCEEDURE IS ONLY CARRIED OUT FOR NON-LINEAR
C                   TIME DOMAIN ANALYSIS, IF(IFLNER.EQ.0) SKIP IT
C
C                   FUTHERMORE
C
C                   YDOT NOW CALLS SUBROUTINE DUMBRT TO DEFINE FORCES OF
C                   INERTIA WHICH MUST BE INTRODUCED IF AN ACCELERATING
C                   FRAME OF REFERENCE IS TO BE USED
C
C                   CALL TO COMPIN ADDED, NOW COMPOSITE SYSTEM STATE
C                   COMPUTED AT END OF EACH INTEGRATION STEP
C
C                   CALL TO FINDU NOT MADE EACH INTEGRATION CYCLE NOW,
C                   FLIPPING FROM ONE TO ANOTHER SET OF INDEPENDENT
C                   DEGREES OF FREEDOM AVIODED, ACCURACY PROBLEM AND
C                   PHYSICAL INTERPRETATION OF RESULTS CAPABILITY
C                   INHANCED
C
C***********************************************************************
C
      RETURN
      END
```