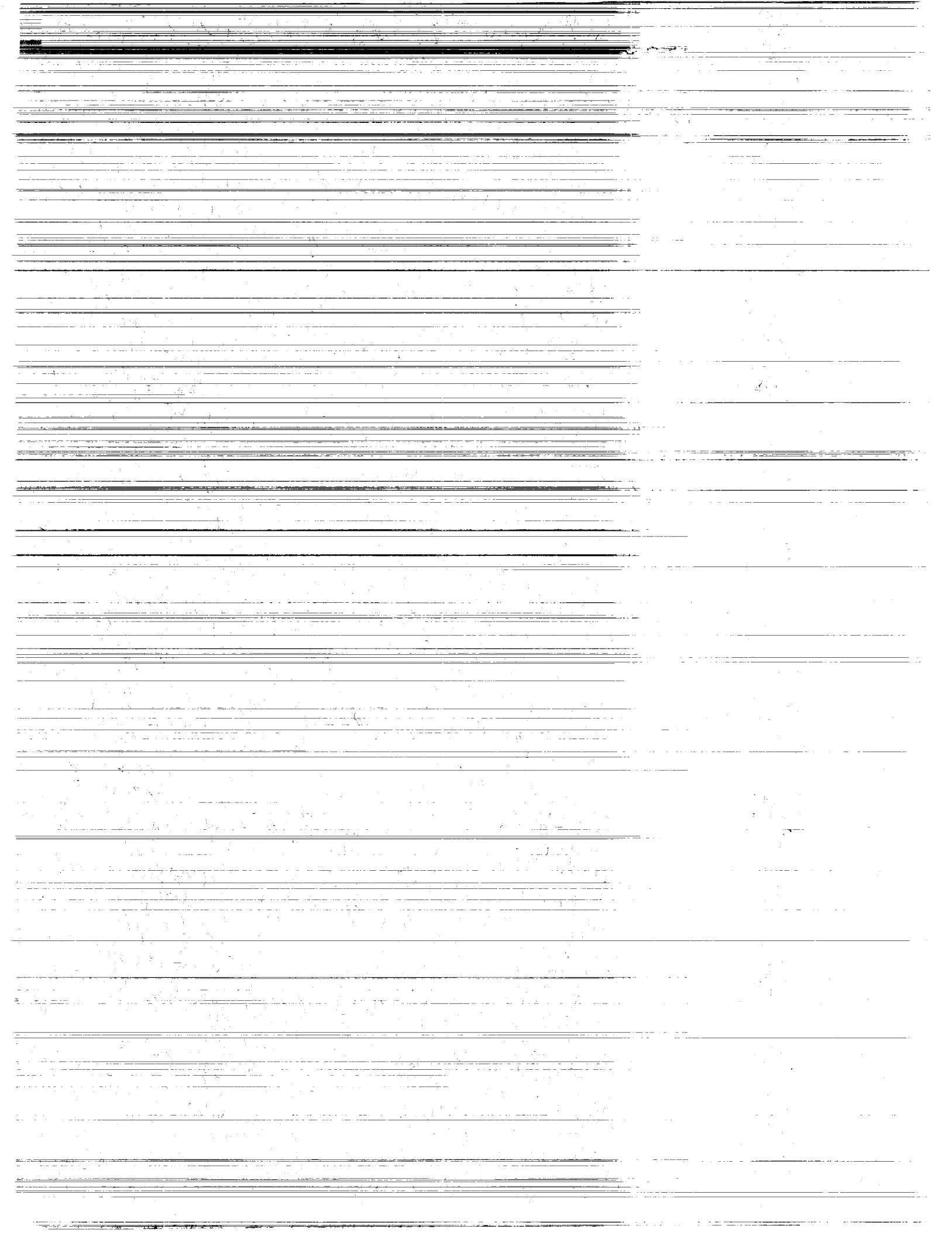
NASA Technical Paper 3220

April 1992

Technique To Eliminate Computational Instability in Multibody Simulations Employing the Lagrange Multiplier

G. Watts

NASA



NASA Technical Paper 3220

1992

Technique To Eliminate Computational Instability in Multibody Simulations Employing the Lagrange Multiplier

G. Watts
George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama

NASA
National Aeronautics and Space Administration
Office of Management
Scientific and Technical Information Program

TABLE OF CONTENTS

TECHNICAL PAPER

TECHNIQUE TO ELIMINATE COMPUTATIONAL INSTABILITY IN MULTIBODY SIMULATIONS EMPLOYING THE LAGRANGE MULTIPLIER

I. INTRODUCTION

The Lagrange multiplier method has been used for many years by the author and other engineers to develop the equations of motion in multibody problems. An important application for this method is in the simulation of a space shuttle solid rocket booster (SRB) which is being decelerated and stabilized for water impact by a drogue parachute or three main parachutes. Reference 1 describes one of the first digital computer programs at Marshall Space Flight Center (MSFC) that employed the Lagrange multiplier method to simulate SRB-parachute dynamics.

Use of the Lagrange multiplier has an advantage over some other formulations in that it preserves the original simple form of the equations of motion for each body. However, in typical time-varying dynamics simulations that use the Lagrange multiplier, computational instability can occur because integration error causes the two (or more) attached bodies to drift apart and violate the constraints. The computational instability usually happens suddenly, and the computed values "blow up" within a few time steps. Computational instability occurred often in SRB-parachute dynamics work at MSFC in the 1970's resulting in much frustration for those involved.

This paper describes a technique to eliminate computational instability caused by drifting apart of the attached bodies when the Lagrange multiplier is used. The equations of motion are not changed; instead, the programming of the equations on a digital computer is changed. A computer program listing is provided in the appendix to aid in the description of the programming technique.

II. EQUATIONS OF MOTION WITH LAGRANGE MULTIPLIER

For the sake of completeness, the multibody equations of motion employing the Lagrange multiplier will first be presented. A simple dynamics problem has been selected as the example so that the basic principles can be clearly illustrated. The example consists of two rigid bodies connected by a frictionless swivel. Each body would have six degrees of freedom (DOF) if not connected to the other body.

The development of the equations of motion with the Lagrange multiplier begins with the general Newtonian equations that will be written for each body at its center of mass (CM) in a body-fixed frame. The equations for either body have the familiar vector form:

Translation

$$\text{MASS}*\overrightarrow{VDOT} = \overrightarrow{\text{FORCE}} - \vec{W}\times\vec{V}*\text{MASS} \ , \tag{1A}$$

Rotation

$$\overrightarrow{HDOT} = \overrightarrow{\text{MOMENT}} - \vec{W}\times\vec{H} \ , \tag{1B}$$

where $\vec{V}$ is the velocity of the CM; $\vec{W}$ and $\vec{H}$ are the angular velocity and momentum, respectively; $\overrightarrow{VDOT}$ and $\overrightarrow{HDOT}$ are time derivatives taken in the body frame; and MASS is a scalar quantity. The $\overrightarrow{\text{FORCE}}$ and $\overrightarrow{\text{MOMENT}}$ vectors include both the external and attach point constraint components.

The Newtonian equations for each body must be converted to matrix form by using inertia matrices and tilde matrices. The equations for both bodies can then be combined into the following matrix equation:

$$[M][D\Omega] = [FE] + [FA] + [FW] \ , \tag{2}$$

where $[M]$ is a 12×12 quasi-diagonal matrix containing the mass and inertia matrices for both bodies which will be described in detail later; $[D\Omega]$ is a 12×1 matrix containing the body-fixed components of the $\overrightarrow{VDOT}$ and $\overrightarrow{WDOT}$ accelerations for each body; $[FE]$ is a 12×1 matrix containing the external forces and moments for each body; $[FA]$ is a 12×1 matrix containing the attach point constraint forces and moments expressed at the CM of each body; and $[FW]$ is a 12×1 matrix equivalent of the $(-\vec{W}\times\vec{V}*\text{MASS})$ and $(-\vec{W}\times\vec{H})$ vectors for each body. The calculation of $[FW]$ is described in the computer program listing in the appendix.

At this point, the subscripts B and C are chosen to represent the two bodies, and the following sequence is specified for the cells in the $[M]$ matrix and the 12×1 matrices in equation (2):

First cell—translation of body B
Second cell—rotation of body B
Third cell—translation of body C
Fourth cell—rotation of body C

$[M]$, the 12×12 quasi-diagonal matrix for masses and inertias in equation (2), thus has the form:

$$[M] = \begin{bmatrix} [\text{MASS}_B] & & & \\ & [IM_B] & & 0 \\ & & [\text{MASS}_C] & \\ & 0 & & [IM_C] \end{bmatrix}$$

where $[\text{MASS}_B]$ and $[\text{MASS}_C]$ are 3×3 diagonal matrices containing the mass of each body, and $[IM_B]$ and $[IM_C]$ are the 3×3 inertia matrices in body-fixed frames.

$[D\Omega]$ in equation (2) is a 12×1 matrix with the following four 3×1 cells containing the body-fixed components of translational and rotational acceleration for both bodies:

$$[D\Omega] = \begin{bmatrix} [VDOT_B] \\ [WDOT_B] \\ [VDOT_C] \\ [WDOT_C] \end{bmatrix} .$$

A 12×1 velocity matrix, $[\Omega]$, will be defined similarly to $[D\Omega]$. Its four 3×1 cells are:

$$[\Omega] = \begin{bmatrix} [V_B] \\ [W_B] \\ [V_C] \\ [W_C] \end{bmatrix} ,$$

where $[V_B]$ and $[V_C]$ contain the body-fixed components of the velocity of the CM for each body, and $[W_B]$ and $[W_C]$ contain the body-fixed angular velocity components for each body.

Continuing with the Newtonian equations, equation (2) will be rearranged using the inverse of $[M]$ as follows:

$$[D\Omega] = [M]^{-1}[[F]+[FA]] \ , \tag{3}$$

where $[F]$ is simply the sum of $[FE]$ and $[FW]$. If the attach point constraint forces and moments, $[FA]$, were known, equation (3) could be used to determine the translational and rotational accelerations of both bodies as is done in a typical dynamics problem.

To calculate $[FA]$, the Lagrange multiplier method will be used. To prepare for the incorporation of the Lagrange multiplier, the attach point constraint equation, which states the velocities of the attach points of both bodies are equal, will be presented. In vector form, the attach point constraint equation for the chosen example is:

$$\vec{V}_B+\vec{W}_B\times\vec{L}_B \ = \ \vec{V}_C+\vec{W}_C\times\vec{L}_C \ , \tag{4}$$

where the subscripts B and C represent the two bodies (as is the case throughout this paper); $\vec{V}$ and $\vec{W}$ are the velocity of the CM and angular velocity previously defined; and $\vec{L}$ is the attach point vector which defines the distance from the CM to the attach point for each body.

Equation (4) must be converted to a matrix form that will allow the 12×1 $[\Omega]$ matrix to be factored out. To do this, a vector, $\vec{U}$, will be substituted for the $(\vec{W}\times\vec{L})$ cross-product of each side of equation (4):

$$\vec{V}_B+\vec{U}_B \ = \ \vec{V}_C+\vec{U}_C \ . \tag{5}$$

Equation (5) can easily be converted to a matrix equation in the C body frame by using a transformation matrix, $[ACB]$, to transform from the B body frame to the C body frame. The resulting equation is:

$$[ACB][V_B]+[ACB][U_B] \ = \ [V_C]+[U_C] \ . \tag{6}$$

Returning to the $\vec{U}$ vectors in equation (5) they represent the following cross products:

$$\vec{U}_B = \vec{W}_B \times \vec{L}_B \quad \text{and} \quad \vec{U}_C = \vec{W}_C \times \vec{L}_C \ . \tag{7}$$

These two cross products can be rearranged by reversing their order and inserting a minus sign, producing:

$$\vec{U}_B = -\vec{L}_B \times \vec{W}_B \quad \text{and} \quad \vec{U}_C = -\vec{L}_C \times \vec{W}_C \ . \tag{8}$$

By using the tilde matrix for each attach point vector, $\vec{L}$, the two parts of equation (8) can be converted to matrix form as follows:

$$[U_B] = -[LT_B][W_B] \quad \text{and} \quad [U_C] = -[LT_C][W_C] \ , \tag{9}$$

where $[LT_B]$ and $[LT_C]$ are tilde matrices for the attach point vectors. The tilde matrices and angular velocity matrices are all expressed in their original frames.

By substituting both parts of equation (9) back into equation (6), the following equation is obtained:

$$[ACB][V_B] - [ACB][LT_B][W_B] = [V_C] - [LT_C][W_C] \ , \tag{10}$$

where every term is a 3×1 matrix.

After gathering all the terms in equation (10) to the left side, we have:

$$[ACB][V_B] - [ACB][LT_B][W_B] - [V_C] + [LT_C][W_C] = 0 \ , \tag{11}$$

Equation (11), the constraint equation, is now in a form that will allow the 12×1 [Ω] matrix to be factored out. After factoring out [Ω], the reconfigured constraint equation is:

$$[A][\Omega] = 0 \ , \tag{12}$$

where [A] is a 3×12 matrix that will be called the "constraint matrix." By inspection of the terms in equation (11), [A] can be expressed as four 3×3 cells as follows:

$$[A] = \left[[ACB] \ \vdots \ -[ACB][LT_B] \ \vdots \ -[\text{IDENT}] \ \vdots \ [LT_C] \right] \ ,$$

where [IDENT] is a 3×3 identity matrix.

Continuing with the derivation, equation (12) will be differentiated with respect to time, which gives the following:

$$[A][D\Omega] + [ADOT][\Omega] = 0 \ . \tag{13}$$

Substituting the expression for $[D\Omega]$ from equation (3) into equation (13) yields:

$$[A][M]^{-1}[[F]+[FA]]+[ADOT][\Omega] = 0 \ . \tag{14}$$

Equation (14) will be rearranged as follows:

$$[A][M]^{-1}[FA] = -[A][M]^{-1}[F]-[ADOT][\Omega] \ . \tag{15}$$

$[FA]$, the attach point constraint forces and moments, must be determined to finish the derivation of the equations of motion. Unfortunately, no inverse exists for the 3×12 matrix term $[A][M]^{-1}$ in equation (15), preventing a direct solution for $[FA]$.

To obtain $[FA]$, the Lagrange multiplier method which uses the following derivable relationship for the attach point constraint forces and moments will be introduced:

$$[FA] = [A]^T[\lambda] \ , \tag{16}$$

where $[A]^T$ is the transpose of the "constraint matrix," and $[\lambda]$ is the 3×1 Lagrange multiplier matrix.

The derivation of the Lagrange multiplier relationship in equation (16) will not be presented in this paper. The reader is urged to study reference 2 for an excellent explanation of the Lagrange multiplier and related subjects.

To continue the calculation of $[FA]$, note that $[\lambda]$ can be determined by first combining equations (15) and (16) to eliminate $[FA]$, which produces the following:

$$[A][M]^{-1}[A]^T[\lambda] = -[A][M]^{-1}[F]-[ADOT][\Omega] \ . \tag{17}$$

The matrix, $[A][M]^{-1}[A]^T$, is a 3×3 which has an inverse. Therefore, equation (17) can be used to solve directly for $[\lambda]$ as follows:

$$[\lambda] = \left[[A][M]^{-1}[A]^T\right]^{-1}\left[-[A][M]^{-1}[F]-[ADOT][\Omega]\right] \ . \tag{18}$$

To calculate $[FA]$, the expression for $[\lambda]$ in equation (18) is substituted back into equation (16) to produce the following:

$$[FA] = [A]^T\left[[A][M]^{-1}[A]^T\right]^{-1}\left[-[A][M]^{-1}[F]-[ADOT][\Omega]\right] \ . \tag{19}$$

The derivation of $[FA]$, the attach point constraint forces and moments, is now complete. Equation (19) can be used to calculate $[FA]$ as a function of the known system parameters such as mass, geometry, external forces, and velocities. The components of $[FA]$ are expressed at the CM of each body, not at the attach point.

III. GENERAL PROGRAMMING COMMENTS

After $[FA]$ has been calculated in the computer program, it is inserted in equation (3) to calculate $[D\Omega]$ which contains the 12 acceleration components. For convenience, equation (3) is again presented:

$$[D\Omega] = [M]^{-1}[[F]+[FA]] \ . \tag{3}$$

In all previous simulations which used the Lagrange multiplier and which are known to the author, the 12 accelerations in equation (3) were integrated to obtain velocity and angular velocity components. After the integration had been performed over a period of time, integration error caused the two bodies to drift apart and violate the constraints, which eventually led to computational instability.

IV. TECHNIQUE TO ELIMINATE COMPUTATIONAL INSTABILITY

The technique that eliminates computational instability caused by drifting of the attached bodies will now be presented. Instead of integrating the 12 acceleration coordinates in equation (3), the 2 bodies in the chosen example will be examined to determine the independent coordinates. One can see that there are only nine independent coordinates in the example: three rotational coordinates for each body (total of six coordinates) and three translational coordinates for one of the bodies. Body B will be chosen as the body whose translational coordinates are independent, meaning that the translational coordinates of body C are dependent upon the other nine coordinates. Only the nine independent acceleration coordinates will be integrated to get the three translational velocity components of body B and the six angular velocity components for both bodies. The three translational velocity components of body C must somehow be calculated. To do this, one needs only to take equation (11), the constraint equation, and solve it for the velocity of body C:

$$[V_C] = [ACB][V_B]-[ACB][LT_B][W_B]+[LT_C][W_C] \ , \tag{20}$$

where all matrices have been previously defined. By using equation (20) to calculate the velocity of body C at each integration time step, drifting of the bodies is eliminated, as is the associated computational instability.

It would be sufficient to stop at this point because the stated intention of preventing computational instability has been accomplished. However, one more step is added to the process. The translational position of body C will be determined in the same manner as was its velocity. Specifically, a position constraint equation will be used, instead of integration, to define the translation of the CM of body C at each integration time step. The two bodies will thus not only maintain the proper relationship of their velocities but their positions as well. Details of the position constraint are presented in the computer program listing in the appendix.

REFERENCES

1. Murphree, H.I.: "Computer Program Development and User's Manual for Program PARACH." NASA TM-78238, Marshall Space Flight Center, AL, September 1979.

2. Rheinfurth, M.H., and Wilson, H.B.: "Methods of Applied Dynamics." NASA Reference Publication 1262, Marshall Space Flight Center, AL, May 1991.

APPENDIX

This appendix contains a digital computer program listing of the two-body simulation which uses the programming technique that eliminates computational instability associated with the Lagrange multiplier method. The two bodies are connected by a frictionless swivel. The aerodynamics and certain other aspects of the simulation are somewhat simplified to allow clearer illustration of the basic principles used. No subroutines are listed because they all perform relatively simple calculations such as matrix multiplication, trigonometric functions with zero denominators, or integration. The program listing, in FORTRAN, begins on the next page.

```
C                                                                      JAN 15, 1992
C                                                                      GAINES WATTS
C                                                                      MSFC - ED13
C*********************************************************************
C
C    TTTTTTT  W   W   W   OOOOO   BBBBBB    OOOOO   DDDDDD   Y     Y
C       T     W   W   W  O     O  B     B  O     O  D     D   Y   Y
C       T     W   W   W  O     O  BBBBBB   O     O  D     D    Y
C       T     W   W   W  O     O  B     B  O     O  D     D    Y
C       T      WW WW    OOOOO   BBBBBB    OOOOO   DDDDDD     Y
C
C*********************************************************************
C
C     A SIMPLE TWO-BODY LAGRANGE MULTIPLIER SIMULATION TO ILLUSTRATE
C  THE PROGRAMMING TECHNIQUE THAT ELIMINATES COMPUTATIONAL INSTABILITY
C  CAUSED BY DRIFTING APART OF THE BODIES.
C
C*********************************************************************
C
C
C     PROGRAM INTRODUCTORY COMMENTS:
C
C     1.   THE TWO BODIES IN THIS PROGRAM ARE AN SRB AND A PARACHUTE,
C          DESIGNATED BY SUBSCRIPTS "B" AND "C", RESPECTIVELY.  A
C          FRICTIONLESS SWIVEL IS USED TO CONNECT THEM.
C
C     2.   BOTH BODIES ARE ASSUMED TO BE RIGID WITH NO CHANGES IN
C          MASS PROPERTIES DURING A SIMULATION RUN.  NO ADDED MASS,
C          APPARENT MASS, OR AERODYNAMIC DAMPING ARE CONSIDERED.
C
C     3.   A FLAT EARTH WITH CONSTANT GRAVITATIONAL ACCELERATION IS
C          ASSUMED.  ATMOSPHERIC DENSITY VARIES WITH ALTITUDE, BUT NO
C          WINDS ARE USED.
C
C     4.   THE INERTIAL REFERENCE IS ASSUMED TO BE THE 'I' FRAME WHICH IS
C          LOCATED AT THE EARTH'S SURFACE DIRECTLY BENEATH THE INITIAL
C          POSITION OF THE SRB.  THE ORIENTATION OF THE 'I' FRAME
C          IS:  NORTH - EAST- DOWN.
C
C     5.   THE 'B' AND 'C' FRAMES ARE FIXED AT THE C.M. OF THE SRB AND
C          PARACHUTE, RESPECTIVELY.  EACH FRAME USES THE X-AXIS AS THE
C          ROLL AXIS.
C
C     6.   TWO SETS OF 3-2-1 EULER ANGLES RELATE THE 'B' AND 'C'
C          FRAMES TO THE 'I' FRAME.
C
C     7.   AERO COEFFICIENT TABLES FOR THE SRB AND THE PARACHUTE ARE
C          EXPRESSED IN "MISSILE" AXES AS A FUNCTION OF THE TOTAL
C          ANGLE OF ATTACK.  A POSITIVE ROTATION ABOUT THE ROLL AXIS
C          IS USED TO TRANSFORM FROM "MISSILE" AXES TO THE BODY-FIXED
C          FRAME FOR THE SRB OR PARACHUTE.
C
C     8.   THE MOMENT REFERENCE POINT (M.R.P.) FOR EITHER BODY IS AT
C          THE C.M.
C
```

```
C     9.    A 'V' FRAME IS USED FOR THE SRB, HAVING ITS X-AXIS ALONG THE
C           SRB VELOCITY VECTOR AND ITS Y-AXIS HORIZONTAL.
C
C    10.    A FLAG IS AVAILABLE TO ZERO OUT THE EXTERNAL FORCES AND
C           MOMENTS, AND PERMIT THE ATTACH POINT CONSTRAINT FORCES AND
C           MOMENTS TO BE CHECKED.  (SET IFE = 0 TO USE THIS OPTION).
C
C
C*****************************************************************************
C
      PARAMETER  (NAB  =  9)
      PARAMETER  (NAC  =  5)
      PARAMETER  (NRHO =  8)
      PARAMETER  (NRNK = 30)
C
      REAL LABB(3),LACC(3),LABI(3),LACI(3),LREFB,LREFC
      REAL MASSB,MASSC,IXXB,IXXC,IYYB,IYYC,IZZB,IZZC,IXYB,IXYC,
     2IXZB,IXZC,IYZB,IYZC
      REAL IMB(3,3),IMC(3,3),IMBINV(3,3),IMCINV(3,3)
      REAL M(12,12),MINV(12,12),MIDENT(12,12)
C
      DIMENSION ABI(3,3),ACI(3,3),AIB(3,3),AIC(3,3),ACB(3,3)
      DIMENSION ABV(3,3),AVI(3,3),AIV(3,3)
      DIMENSION A(3,12),AT(12,3),ADOT(3,12),AMINV(3,12),AMINVA(3,3),
     2TERMA1(3,3),AMINVF(3),ADOTOM(3),TERMV1(3),TERMV2(3),ADOMEG(3)
C
      DIMENSION OMEGA(12),DOMEGA(12),DOM9(9)
      DIMENSION VIBB(3),VICC(3),VIBI(3),VICI(3),VOBB(3),VOCC(3),
     2VABB(3),VABC(3),RIBI(3),RABI(3),RICI(3),OMB(3),OMC(3)
      DIMENSION FL(12),FG(12),FE(12),FW(12),F(12),FA(12),FTOTAL(12)
      DIMENSION GI(3),GB(3),GC(3),H(12)
C
      DIMENSION ALBTL(NAB),CAMBTL(NAB),CNMBTL(NAB),CMMBTL(NAB)
      DIMENSION ALCTL(NAC),CAMCTL(NAC),CNMCTL(NAC),CMMCTL(NAC)
      DIMENSION CARD(20),TITLE(16),RHOALT(NRHO),RHOTL(NRHO)
      DIMENSION XXD(NRNK),XX(NRNK)
C
      EQUIVALENCE (XXD(01),B11D),(XXD(02),B12D),(XXD(03),B13D)
      EQUIVALENCE (XXD(04),B21D),(XXD(05),B22D),(XXD(06),B23D)
      EQUIVALENCE (XXD(07),B31D),(XXD(08),B32D),(XXD(09),B33D)
      EQUIVALENCE (XXD(10),C11D),(XXD(11),C12D),(XXD(12),C13D)
      EQUIVALENCE (XXD(13),C21D),(XXD(14),C22D),(XXD(15),C23D)
      EQUIVALENCE (XXD(16),C31D),(XXD(17),C32D),(XXD(18),C33D)
      EQUIVALENCE (XXD(19),DOM9(1))
      EQUIVALENCE (XXD(28),VIBI(1))
C
      EQUIVALENCE (XX(01),B11),(XX(02),B12),(XX(03),B13)
      EQUIVALENCE (XX(04),B21),(XX(05),B22),(XX(06),B23)
      EQUIVALENCE (XX(07),B31),(XX(08),B32),(XX(09),B33)
      EQUIVALENCE (XX(10),C11),(XX(11),C12),(XX(12),C13)
      EQUIVALENCE (XX(13),C21),(XX(14),C22),(XX(15),C23)
      EQUIVALENCE (XX(16),C31),(XX(17),C32),(XX(18),C33)
      EQUIVALENCE (XX(19),VIBB(1)),(XX(22),OMB(1)),(XX(25),OMC(1))
      EQUIVALENCE (XX(28),RIBI(1))
C
```

```
C * * * * * * * * * * *
C      AERO COEFFICIENT TABLES IN MISSILE AXES FOR THE SRB AND THE
C         PARACHUTE AS A FUNCTION OF TOTAL ANGLE OF ATTACK.
C
C      CA - - AXIAL FORCE
C      CN - - NORMAL FORCE
C      CM - - PITCH MOMENT
C      CY - - SIDE FORCE
C      CW - - YAW MOMENT
C      CR - - ROLL MOMENT
C
       DATA ALBTL/0.,20.,40.,60.,90.,120.,147.,160.,180./
       DATA CAMBTL/0.0,0.0,0.0,+0.50,+1.40,-1.35,-3.65,-3.00,-2.20/
       DATA CNMBTL/0.0,0.0,0.0,+9.00,+9.00,+9.00,+4.10,+1.80,+0.00/
       DATA CMMBTL/0.0,0.0,0.0,+15.5,+5.00,-3.30,-1.90,-1.10,+0.00/
       CYMB   = 0.0
       CWMB   = 0.0
       CRMB   = 0.0
C
       DATA ALCTL/0.,5.,10.,15.,25./
       DATA CAMCTL/+0.62,+0.62,+0.62,+0.61,+0.56/
       DATA CNMCTL/.000,+.0100,+.032,+.090,+.170/
       DATA CMMCTL/.000,-.0133,-.037,-.091,-.175/
       CYMC   = 0.0
       CWMC   = 0.0
       CRMC   = 0.0
C * *
C      REFERENCE AREAS AND LENGTHS FOR THE SRB AND PARACHUTE.
       SREFB = 116.26
       LREFB = 12.1667
       SREFC = 2289.1
       LREFC = 54.00
C
C * * * * * * * * * * *
C      ATMOS. DENSITY TABLE - - ENGLISH SYSTEM.
C
       DATA RHOALT/00.,5000.,8224.,8803.,14000.,20000.,30000.,40000./
       DATA RHOTL/0.0022964,0.0019909,0.0018053,0.0017735,
      B0.0015086,0.0012483,0.0009024,0.0006306/
C
C * * * * * * * * * * *
C      CONSTANTS.
       CNV    = 57.2957795
       GZERO = 32.1740485
C * * * * * * * * * * *
C      READ AND PRINT DESIRED PORTION OF FORTRAN SOURCE DATASET.
C
       READ (5,805) NPMIN,NPMAX
       DO 100 I=1,NPMAX
       READ (9,820, END=105) CARD
       IF(I .LT. NPMIN) GO TO 100
       WRITE(6,822) CARD
  100  CONTINUE
  105  WRITE (6,800)
C * * * * * * * * * * *
```

```
C      READ AND WRITE THE INPUT DATA.
C
       READ (5,801) TITLE,IDIAG,IFE
       WRITE(6,802) TITLE,IDIAG,IFE
       READ (5,807) TSTART,TMAX,DELT,DTPRT,TNP1,TNP2
       WRITE(6,808) TSTART,TMAX,DELT,DTPRT,TNP1,TNP2
C
       READ (5,807) VIBS,AZIIBS,GAMIBS,ALTBS
       WRITE(6,808) VIBS,AZIIBS,GAMIBS,ALTBS
       READ (5,807) PDEGBS,QDEGBS,RDEGBS,ALPHBS,PHIABS,BANKBS
       WRITE(6,808) PDEGBS,QDEGBS,RDEGBS,ALPHBS,PHIABS,BANKBS
       READ (5,807) PDEGCS,QDEGCS,RDEGCS,PSICS,THETCS,PHICS
       WRITE(6,808) PDEGCS,QDEGCS,RDEGCS,PSICS,THETCS,PHICS
C
       READ (5,807) WGHTB,IXXB,IYYB,IZZB,IXYB,IXZB,IYZB
       WRITE(6,808) WGHTB,IXXB,IYYB,IZZB,IXYB,IXZB,IYZB
       READ (5,807) WGHTC,IXXC,IYYC,IZZC,IXYC,IXZC,IYZC
       WRITE(6,808) WGHTC,IXXC,IYYC,IZZC,IXYC,IXZC,IYZC
       READ (5,807) XATTB,YATTB,ZATTB,XATTC,YATTC,ZATTC
       WRITE(6,808) XATTB,YATTB,ZATTB,XATTC,YATTC,ZATTC
C
       WRITE(6,802)
C * * * * * * * * * *
       MASSB = WGHTB/GZERO
       MASSC = WGHTC/GZERO
C
C**********************************
C      SET UP THE [M] MATRIX AND ITS INVERSE.  [M] IS A 12X12 WHICH CAN
C         BE TREATED AS A QUASI-DIAGONAL MATRIX WITH FOUR 3X3 CELLS.
C         THE INVERSE OF [M] CAN BE FOUND BY SIMPLY TAKING THE INVERSE
C         OF EACH CELL.
C
C                  ---------  0   0   0   0   0   0   0   0   0
C                  |       |
C                  | MASSB |  0   0   0   0   0   0   0   0   0
C                  |       |
C                  ---------  0   0   0   0   0   0   0   0   0
C
C                   0   0   0 ---------  0   0   0   0   0   0
C                             |       |
C                   0   0   0 |  IMB  |  0   0   0   0   0   0
C                             |       |
C          [M]  =   0   0   0 ---------  0   0   0   0   0   0
C
C                   0   0   0   0   0   0 ---------  0   0   0
C                                         |       |
C                   0   0   0   0   0   0 | MASSC |  0   0   0
C                                         |       |
C                   0   0   0   0   0   0 ---------  0   0   0
C
C                   0   0   0   0   0   0   0   0   0 ---------
C                                                     |       |
C                   0   0   0   0   0   0   0   0   0 |  IMC  |
C                                                     |       |
C                   0   0   0   0   0   0   0   0   0 ---------
```

```
C
C * *
C     FIRST, ZERO ALL THE ELEMENTS OF [M] AND [MINV].
      DO 121 I=1,12
      DO 120 J=1,12
      M(I,J)    = 0.0
  120 MINV(I,J) = 0.0
  121 CONTINUE
C
C     NOW, SET UP THE MASS CELLS IN [M] AND [MINV].  EACH MASS CELL IS
C        A DIAGONAL MATRIX.
      DO 130 I=1,3
      M(I,I)     = MASSB
      M(I+6,I+6) = MASSC
      MINV(I,I)     = 1./MASSB
  130 MINV(I+6,I+6) = 1./MASSC
C
C     NEXT, SET UP EACH INERTIA MATRIX AND ITS INVERSE.
      CALL IMATRX (IXXB,IYYB,IZZB,IXYB,IXZB,IYZB,IMB)
      CALL IMATRX (IXXC,IYYC,IZZC,IXYC,IXZC,IYZC,IMC)
      CALL MAINV3 (IMB,IMBINV)
      CALL MAINV3 (IMC,IMCINV)
C
C     FINALLY, SET UP THE INERTIA CELLS IN [M] AND [MINV].
      DO 133 I=1,3
      DO 132 J=1,3
      M(I+3,J+3) = IMB(I,J)
      M(I+9,J+9) = IMC(I,J)
      MINV(I+3,J+3) = IMBINV(I,J)
  132 MINV(I+9,J+9) = IMCINV(I,J)
  133 CONTINUE
C * *
C     CHECK THE INVERSE OF [M] BY PRINTING [M] AND [M][MINV].
C * *
      DO 141 I=1,12
  141 WRITE (6,812) (M(I,J),J=1,12)
      CALL MMUL (M,MINV,MIDENT,12,12,12)
      WRITE (6,802)
      DO 142 I=1,12
  142 WRITE (6,812) (MIDENT(I,J),J=1,12)
C
C**********************************
C     SET UP VECTORS FOR C.M.-TO-ATTACH-POINT DISTANCES.  BOTH VECTORS
C        ARE DEFINED AS POSITIVE FROM C.M. TO ATTACH POINT.
      LABB(1) = XATTB
      LABB(2) = YATTB
      LABB(3) = ZATTB
      LACC(1) = XATTC
      LACC(2) = YATTC
      LACC(3) = ZATTC
C
C * * * * * * * * * * * * * * * * * * *
C     SET UP THE 3RD AND 4TH CELLS OF [A], THE 3X12 CONSTRAINT
C        MATRIX WHICH HAS THE GENERAL FORM:
C
```

```
C                    |          |          |          |          |
C          [A]  =    |  +ACB    |-ACB*LTB| -IDENT   |  +LTC    |
C                    |          |          |          |          |
C
C          WHERE LTB AND LTC ARE TILDE MATRICES FOR THE LABB AND
C             LACC VECTORS.
C
C     FIRST, ZERO ALL THE ELEMENTS OF [A].  ZERO [ADOT] ALSO.
C
      DO 173 I=1,3
      DO 172 J=1,12
      A(I,J)    = 0.0
  172 ADOT(I,J)= 0.0
  173 CONTINUE
C
C     NOW, FILL IN NON-ZERO ELEMENTS IN THE 3RD AND 4TH CELLS OF [A].
      A(1,7) = -1.0
      A(2,8) = -1.0
      A(3,9) = -1.0
      A(1,11) = -LACC(3)
      A(1,12) = +LACC(2)
      A(2,10) = +LACC(3)
      A(2,12) = -LACC(1)
      A(3,10) = -LACC(2)
      A(3,11) = +LACC(1)
C
C * * * * * * * * * * * * *
C     BEGIN INITIALIZATION SECTION IN WHICH THE VALUES OF ALL NEEDED
C        PARAMETERS ARE CALCULATED AT TIME = TSTART.
C * * * * *
      TIME = TSTART
C * * *
C     ESTABLISH THE SRB PARAMETERS AT TIME = TSTART BY SETTING THEM
C        EQUAL TO THE VALUES ALREADY READ IN.
      VIB   = VIBS
      AZIIB = AZIIBS
      GAMIB = GAMIBS
      ALTB  = ALTBS
      PDEGB = PDEGBS
      QDEGB = QDEGBS
      RDEGB = RDEGBS
      ALPHAB= ALPHBS
      PHIAB = PHIABS
      BANKB = BANKBS
C * * *
C     DETERMINE 2 MATRICES TO INITIALIZE [ABI], THE 'I' FRAME TO 'B'
C        FRAME TRANSFORMATION.  - -    [ABI] = [ABV][AVI]
C *
C     SET UP [ABV] USING A 1-2-1 EULER ANGLE SEQUENCE (BANKB,ALPHAB,
C        PHIAB).
C
      SBKB = SIN(BANKB/CNV)
      CBKB = COS(BANKB/CNV)
      SALB = SIN(ALPHAB/CNV)
      CALB = COS(ALPHAB/CNV)
```

```
      SPAB = SIN(PHIAB/CNV)
      CPAB = COS(PHIAB/CNV)
      ABV(1,1) = CALB
      ABV(1,2) = SBKB*SALB
      ABV(1,3) =-CBKB*SALB
      ABV(2,1) = SPAB*SALB
      ABV(2,2) = CPAB*CBKB - SPAB*SBKB*CALB
      ABV(2,3) = CPAB*SBKB + SPAB*CBKB*CALB
      ABV(3,1) = CPAB*SALB
      ABV(3,2) =-SPAB*CBKB - CPAB*SBKB*CALB
      ABV(3,3) =-SPAB*SBKB + CPAB*CBKB*CALB
C * *
C     SET UP [AVI].
      SAZB = SIN(AZIIB/CNV)
      CAZB = COS(AZIIB/CNV)
      SGMB = SIN(GAMIB/CNV)
      CGMB = COS(GAMIB/CNV)
      AVI(1,1) = CAZB*CGMB
      AVI(1,2) = SAZB*CGMB
      AVI(1,3) =-SGMB
      AVI(2,1) =-SAZB
      AVI(2,2) = CAZB
      AVI(2,3) = 0.0
      AVI(3,1) = CAZB*SGMB
      AVI(3,2) = SAZB*SGMB
      AVI(3,3) = CGMB
C * *
C     CALC. [ABI] AND SET UP ITS ELEMENTS AT TIME = TSTART.
      CALL MMUL (ABV,AVI,ABI,3,3,3)
      B11=ABI(1,1)
      B12=ABI(1,2)
      B13=ABI(1,3)
      B21=ABI(2,1)
      B22=ABI(2,2)
      B23=ABI(2,3)
      B31=ABI(3,1)
      B32=ABI(3,2)
      B33=ABI(3,3)
C * *
C     CALC. THE SRB INERTIAL POSITION COMPONENTS IN THE
C        'I' FRAME AT TIME = TSTART.
      RIBI(1) = 0.0
      RIBI(2) = 0.0
      RIBI(3) = -ALTB
C * *
C     CALC. THE SRB INERTIAL VELOCITY COMPONENTS IN THE 'I' FRAME
C        AND TRANSFORM TO THE 'B' FRAME AT TIME = TSTART.
      VIBI(1) =  VIB*CGMB*CAZB
      VIBI(2) =  VIB*CGMB*SAZB
      VIBI(3) = -VIB*SGMB
      CALL MATVEC (ABI,VIBI,VIBB,3,3)
C
C     SET UP THE SRB ANGULAR VELOCITY COMPONENTS AT TIME = TSTART.
      OMB(1) = PDEGB/CNV
      OMB(2) = QDEGB/CNV
```

```
      OMB(3) = RDEGB/CNV
C
C * * * * * *
C     ESTABLISH THE PARACHUTE PARAMETERS AT TIME = TSTART BY SETTING
C        THEM EQUAL TO THE VALUES ALREADY READ IN.
      PSIC  = PSICS
      THETC = THETCS
      PHIC  = PHICS
      PDEGC = PDEGCS
      QDEGC = QDEGCS
      RDEGC = RDEGCS
C * *
C     SET UP THE PARACHUTE ANGULAR VELOCITY COMPONENTS AT TIME = TSTART.
      OMC(1) = PDEGC/CNV
      OMC(2) = QDEGC/CNV
      OMC(3) = RDEGC/CNV
C * *
C     SET UP [ACI] USING 3-2-1 EULER ANGLE SEQUENCE (PSIC,THETC,PHIC).
      ST3C = SIN(PSIC/CNV)
      CT3C = COS(PSIC/CNV)
      ST2C = SIN(THETC/CNV)
      CT2C = COS(THETC/CNV)
      ST1C = SIN(PHIC/CNV)
      CT1C = COS(PHIC/CNV)
      ACI(1,1) = CT2C*CT3C
      ACI(1,2) = CT2C*ST3C
      ACI(1,3) =-ST2C
      ACI(2,1) = ST1C*ST2C*CT3C - CT1C*ST3C
      ACI(2,2) = ST1C*ST2C*ST3C + CT1C*CT3C
      ACI(2,3) = ST1C*CT2C
      ACI(3,1) = CT1C*ST2C*CT3C + ST1C*ST3C
      ACI(3,2) = CT1C*ST2C*ST3C - ST1C*CT3C
      ACI(3,3) = CT1C*CT2C
C * *
C     SET UP THE ELEMENTS OF [ACI] AT TIME = TSTART.
      C11=ACI(1,1)
      C12=ACI(1,2)
      C13=ACI(1,3)
      C21=ACI(2,1)
      C22=ACI(2,2)
      C23=ACI(2,3)
      C31=ACI(3,1)
      C32=ACI(3,2)
      C33=ACI(3,3)
C * *
C     SET INITIAL VALUES OF PRINT PARAMETERS AND OTHER PARAMETERS.
      TPRINT = TSTART
      IPRFLG = 0
      ICOFLG = 0
      KUTTA = 4
C
C     INITIALIZATION COMPLETED; PRINT HEADER INFORMATION.
C * * * * * * * * * * * * * * * * *
      WRITE (6,800)
      WRITE (6,802) TITLE
```

```
      WRITE (6,808) VIBB(1),VIBB(2),VIBB(3),VIBI(3),XXD(30),XX(30)
      WRITE (6,802)
      GO TO 407
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     BEGIN INTEGRATION LOOP (INTEGRATION SUBR NOT CALLED FIRST PASS).
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
  400 KUTTA = KUTTA + 1
      CALL RUNGF (NRNK,DELT,TIME,XXD,XX,KUTTA)
  407 CONTINUE
C * *
C     CALCULATE ELEMENTS OF [ABIDOT] AND [ACIDOT].
      B11D = B21*OMB(3) - B31*OMB(2)
      B12D = B22*OMB(3) - B32*OMB(2)
      B13D = B23*OMB(3) - B33*OMB(2)
      B21D = B31*OMB(1) - B11*OMB(3)
      B22D = B32*OMB(1) - B12*OMB(3)
      B23D = B33*OMB(1) - B13*OMB(3)
      B31D = B11*OMB(2) - B21*OMB(1)
      B32D = B12*OMB(2) - B22*OMB(1)
      B33D = B13*OMB(2) - B23*OMB(1)
C
      C11D = C21*OMC(3) - C31*OMC(2)
      C12D = C22*OMC(3) - C32*OMC(2)
      C13D = C23*OMC(3) - C33*OMC(2)
      C21D = C31*OMC(1) - C11*OMC(3)
      C22D = C32*OMC(1) - C12*OMC(3)
      C23D = C33*OMC(1) - C13*OMC(3)
      C31D = C11*OMC(2) - C21*OMC(1)
      C32D = C12*OMC(2) - C22*OMC(1)
      C33D = C13*OMC(2) - C23*OMC(1)
C * *
C     SET UP THE [ABI] AND [ACI] MATRICES.
      ABI(1,1)=B11
      ABI(1,2)=B12
      ABI(1,3)=B13
      ABI(2,1)=B21
      ABI(2,2)=B22
      ABI(2,3)=B23
      ABI(3,1)=B31
      ABI(3,2)=B32
      ABI(3,3)=B33
C
      ACI(1,1)=C11
      ACI(1,2)=C12
      ACI(1,3)=C13
      ACI(2,1)=C21
      ACI(2,2)=C22
      ACI(2,3)=C23
      ACI(3,1)=C31
      ACI(3,2)=C32
      ACI(3,3)=C33
C * *
C     CALCULATE [AIB],[AIC], AND [ACB].
```

```
      CALL TRANSP (ABI,3,3,AIB)
      CALL TRANSP (ACI,3,3,AIC)
      CALL MMUL (ACI,AIB,ACB,3,3,3)
C
C * * * * * * * * *
C
C     USE CONSTRAINT EQUATIONS TO DETERMINE THE INERTIAL POSITION AND
C        VELOCITY OF THE PARACHUTE, ELIMINATING DRIFT CAUSED BY
C        INTEGRATION ERROR.
C
C     THE CONSTRAINT EQUATIONS WILL BE CALCULATED USING A VECTOR
C        CROSS-PRODUCT SUBROUTINE INSTEAD OF TILDE MATRICES.  ALLOWING
C        A LITTLE MIXING OF MATRIX AND VECTOR NOTATION, THE CONSTRAINT
C        EQUATIONS ARE:
C
C              RICI = RIBI + [AIB]*LABB - [AIC]*LACC
C              VICC = [ACB]*(VIBB + OMB X LABB) - OMC X LACC
C * * * *
C     BEGIN WITH POSITION CONSTRAINT.
C * * * *
C     TRANSFORM SRB C.M.-TO-ATTACH POINT DISTANCE INTO 'I' FRAME.
      CALL MATVEC (AIB,LABB,LABI,3,3)
C
C     CALC. INERTIAL POSITION OF THE ATTACH POINT IN THE 'I' FRAME.
      CALL VADD (RIBI,LABI,RABI,1)
C
C     TRANSFORM PARACHUTE C.M.-TO-ATTACH POINT DISTANCE INTO 'I' FRAME.
      CALL MATVEC (AIC,LACC,LACI,3,3)
C
C     CALC. INERTIAL POSITION OF PARACHUTE C.M. IN THE 'I' FRAME.
      CALL VADD (RABI,LACI,RICI,-1)
C
C * * * *
C     NOW CALCULATE THE VELOCITY CONSTRAINT.
C * * * *
C     DETERMINE ATTACH POINT VELOCITY RELATIVE TO THE SRB C.M. IN THE
C        'B' FRAME.
      CALL VCROSS (OMB,LABB,VOBB)
C
C     CALC. INERTIAL VELOCITY OF THE ATTACH POINT IN THE 'B'
C        FRAME AND TRANSFORM INTO 'C' FRAME.
      CALL VADD (VIBB,VOBB,VABB,1)
      CALL MATVEC (ACB,VABB,VABC,3,3)
C
C     DETERMINE ATTACH POINT VELOCITY RELATIVE TO THE PARACHUTE C.M. IN
C        THE 'C' FRAME.
      CALL VCROSS (OMC,LACC,VOCC)
C
C     CALC. INERTIAL VELOCITY OF THE PARACHUTE C.M. IN THE 'C' FRAME.
C        (THIS VELOCITY WILL BE PLACED IN THE [OMEGA] MATRIX).
      CALL VADD (VABC,VOCC,VICC,-1)
C
C * * * * * *
C     TRANSFORM INERTIAL VELOCITY OF THE SRB INTO 'I' FRAME TO PROVIDE
C        A DERIVATIVE OF INERTIAL POSITION.  DO THE SAME FOR PARACHUTE.
```

```
      CALL MATVEC (AIB,VIBB,VIBI,3,3)
      CALL MATVEC (AIC,VICC,VICI,3,3)
C
C * * * * * * * * * * * *
C
C     BEGIN AERO SECTION.
C
C * * * *
C     LOOK UP ATMOSPHERIC DENSITY USING SRB ALTITUDE.
      ALTB = -RIBI(3)
      ALTC = -RICI(3)
      CALL TBLXY (NRHO,RHOALT,RHOTL,ALTB,RHO)
C * *
C     CALCULATE DYNAMIC PRESSURES FOR SRB AND PARACHUTE, ALONG WITH
C        ASSOCIATED TERMS FOR AERO CALCULATIONS.
C
      CALL VMAG (VIBB,VIB2,VIB)
      CALL VMAG (VICC,VIC2,VIC)
      QBRB = 0.5*RHO*VIB2
      QBRC = 0.5*RHO*VIC2
      QBRSB = QBRB*SREFB
      QBRSC = QBRC*SREFC
      QBRSLB = QBRSB*LREFB
      QBRSLC = QBRSC*LREFC
C * *
C     CALCULATE ALPHA-TOTAL AND AERO ROLL ANGLE FOR THE SRB
C        AND PARACHUTE.  ALPHA-TOTAL HAS RANGE: 0 TO 180 DEGREES.
C        AERO ROLL ANGLE HAS RANGE: -180 TO +180 DEGREES.
C
      ALPHAB =  ACOS(VIBB(1)/VIB)*CNV
      ALPHAC =  ACOS(VICC(1)/VIC)*CNV
      PHIAB  = ZTAN2(VIBB(2),VIBB(3))*CNV
      PHIAC  = ZTAN2(VICC(2),VICC(3))*CNV
C * *
C     LOOK UP SRB AND PARACHUTE AERO COEFFICIENTS IN MISSILE AXES.
      CALL TBLXY (NAB,ALBTL,CAMBTL,ALPHAB,CAMB)
      CALL TBLXY (NAB,ALBTL,CNMBTL,ALPHAB,CNMB)
      CALL TBLXY (NAB,ALBTL,CMMBTL,ALPHAB,CMMB)
C
      CALL TBLXY (NAC,ALCTL,CAMCTL,ALPHAC,CAMC)
      CALL TBLXY (NAC,ALCTL,CNMCTL,ALPHAC,CNMC)
      CALL TBLXY (NAC,ALCTL,CMMCTL,ALPHAC,CMMC)
C * *
C     TRANSFORM AERO COEFFICIENTS FOR THE SRB AND PARACHUTE INTO THE
C        'B' AND 'C' FRAMES.
C
      SPAB = SIN(PHIAB/CNV)
      CPAB = COS(PHIAB/CNV)
C
      CAB  = +CAMB
      CYB  = +CYMB*CPAB - CNMB*SPAB
      CNB  = +CNMB*CPAB + CYMB*SPAB
      CRB  = +CRMB
      CMB  = +CMMB*CPAB + CWMB*SPAB
      CWB  = +CWMB*CPAB - CMMB*SPAB
```

```
C
      SPAC = SIN(PHIAC/CNV)
      CPAC = COS(PHIAC/CNV)
C
      CAC  = +CAMC
      CYC  = +CYMC*CPAC - CNMC*SPAC
      CNC  = +CNMC*CPAC + CYMC*SPAC
      CRC  = +CRMC
      CMC  = +CMMC*CPAC + CWMC*SPAC
      CWC  = +CWMC*CPAC - CMMC*SPAC
C * *
C     CALC. AERO FORCES AND MOMENTS AT EACH C.M., TAKING INTO ACCOUNT
C        THE SIGN CHANGE REQUIRED FOR CA AND CN.
C
      FL(1)  = -QBRSB*CAB
      FL(2)  = +QBRSB*CYB
      FL(3)  = -QBRSB*CNB
      FL(4)  =  QBRSLB*CRB
      FL(5)  =  QBRSLB*CMB
      FL(6)  =  QBRSLB*CWB
C
      FL(7)  = -QBRSC*CAC
      FL(8)  = +QBRSC*CYC
      FL(9)  = -QBRSC*CNC
      FL(10) =  QBRSLC*CRC
      FL(11) =  QBRSLC*CMC
      FL(12) =  QBRSLC*CWC
C
C     END OF AERO SECTION.
C * * * * * * * * * * *
C
C     SET UP GRAV. ACCELERATION COMPONENTS IN THE 'I' FRAME.
C
      GI(1) =  0.0
      GI(2) =  0.0
      GI(3) =  GZERO
C * *
C     TRANSFORM THE GRAV. ACCELERATION TO THE 'B' AND 'C' FRAMES,
C        AND CALC. THE GRAV. FORCES AT EACH C.M. - - [FG] MATRIX.
C
      CALL MATVEC (ABI,GI,GB,3,3)
      CALL MATVEC (ACI,GI,GC,3,3)
      DO 460 I=1,3
      FG(I)   = MASSB*GB(I)
      FG(I+3) = 0.0
      FG(I+6) = MASSC*GC(I)
  460 FG(I+9) = 0.0
C
C * * * * *
C     IF 'IFE' EQ 0, ZERO OUT THE TERMS THAT COMPRISE THE [FE] MATRIX.
C
      IF(IFE .NE. 0) GO TO 481
      DO 480 I=1,12
      FL(I) = 0.0
  480 FG(I) = 0.0
```

```
  481 CONTINUE
C
C * * * * * * * * * *
C     ALL THE EXTERNAL FORCES AND MOMENTS HAVE NOW BEEN CALCULATED AT
C        THE SRB AND PARACHUTE C.M.'S.  SET UP THE [FE] MATRIX.
      DO 490 I=1,12
  490 FE(I) = FL(I) + FG(I)
C
C * * * * * * * * * * * * * *
C     SET UP THE [OMEGA] MATRIX USING THE PARACHUTE INERTIAL VELOCITY
C        FROM THE CONSTRAINT EQUATION.
      DO 512 I=1,3
      OMEGA(I)   = VIBB(I)
      OMEGA(I+3) = OMB(I)
      OMEGA(I+6) = VICC(I)
  512 OMEGA(I+9) = OMC(I)
C
C * * * * * *
C     CALCULATE THE MOMENTUM MATRIX [H] USING THE 12X12 MASS MATRIX.
      CALL MATVEC (M,OMEGA,H,12,12)
C
C * * *
C     SET UP THE [FW] MATRIX BY CALCULATING THE 'MINUS W-CROSS' TERMS
C        FOR THE SRB AND THE PARACHUTE.
C
      FW(1)  =  -OMB(2)*H(3)  + OMB(3)*H(2)
      FW(2)  =  -OMB(3)*H(1)  + OMB(1)*H(3)
      FW(3)  =  -OMB(1)*H(2)  + OMB(2)*H(1)
C
      FW(4)  =  -OMB(2)*H(6)  + OMB(3)*H(5)
      FW(5)  =  -OMB(3)*H(4)  + OMB(1)*H(6)
      FW(6)  =  -OMB(1)*H(5)  + OMB(2)*H(4)
C
      FW(7)  =  -OMC(2)*H(9)  + OMC(3)*H(8)
      FW(8)  =  -OMC(3)*H(7)  + OMC(1)*H(9)
      FW(9)  =  -OMC(1)*H(8)  + OMC(2)*H(7)
C
      FW(10) =  -OMC(2)*H(12) + OMC(3)*H(11)
      FW(11) =  -OMC(3)*H(10) + OMC(1)*H(12)
      FW(12) =  -OMC(1)*H(11) + OMC(2)*H(10)
C
C * * * * * * * * *
C     ADD [FW] TO THE EXTERNAL FORCES AND MOMENTS TO GET
C        THE [F] MATRIX.
      DO 524 I=1,12
  524 F(I) = FE(I) + FW(I)
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     BEGIN CALCULATION OF ATTACH POINT CONSTRAINT FORCES AND
C        MOMENTS AT THE C.M.'S - - - [FA] MATRIX.
C * * *
C     SET UP THE 1ST AND 2ND CELLS OF [A], THE CONSTRAINT MATRIX.  THE
C        1ST CELL IS [ACB].  THE 2ND CELL, -[ACB][LTB], IS OBTAINED
C        BY USING THE 1ST CELL.
C
```

```
      DO 533 I=1,3
      DO 532 J=1,3
  532 A(I,J) = ACB(I,J)
  533 CONTINUE
      A(1,4) = -A(1,2)*LABB(3) + A(1,3)*LABB(2)
      A(1,5) = +A(1,1)*LABB(3) - A(1,3)*LABB(1)
      A(1,6) = -A(1,1)*LABB(2) + A(1,2)*LABB(1)
      A(2,4) = -A(2,2)*LABB(3) + A(2,3)*LABB(2)
      A(2,5) = +A(2,1)*LABB(3) - A(2,3)*LABB(1)
      A(2,6) = -A(2,1)*LABB(2) + A(2,2)*LABB(1)
      A(3,4) = -A(3,2)*LABB(3) + A(3,3)*LABB(2)
      A(3,5) = +A(3,1)*LABB(3) - A(3,3)*LABB(1)
      A(3,6) = -A(3,1)*LABB(2) + A(3,2)*LABB(1)
C *
C     RECALLING THAT THE 3RD AND 4TH CELLS OF [A] WERE CALCULATED AT THE
C        START OF THE PROGRAM, [A] HAS NOW BEEN DETERMINED.  TAKE THE
C        TRANSPOSE OF [A] FOR LATER USE.
      CALL TRANSP (A,3,12,AT)
C
C * * * *
C     SET UP THE 1ST AND 2ND CELLS OF [ADOT].  THE 3RD AND 4TH CELLS
C        ARE ZERO.  THE 1ST CELL IS [ACBDOT] WHICH IS OBTAINED FROM
C        THE FOLLOWING EQUATION:
C
C             [ACBDOT] = -[OMTC][ACB] + [ACB][OMTB]
C
C            WHERE [OMTC] AND [OMTB] ARE TILDE MATRICES FOR THE
C            PARACHUTE AND SRB ANGULAR VELOCITIES.
C
      ADOT(1,1)=+OMC(3)*A(2,1)-OMC(2)*A(3,1)+OMB(3)*A(1,2)-OMB(2)*A(1,3)
      ADOT(1,2)=+OMC(3)*A(2,2)-OMC(2)*A(3,2)-OMB(3)*A(1,1)+OMB(1)*A(1,3)
      ADOT(1,3)=+OMC(3)*A(2,3)-OMC(2)*A(3,3)+OMB(2)*A(1,1)-OMB(1)*A(1,2)
      ADOT(2,1)=-OMC(3)*A(1,1)+OMC(1)*A(3,1)+OMB(3)*A(2,2)-OMB(2)*A(2,3)
      ADOT(2,2)=-OMC(3)*A(1,2)+OMC(1)*A(3,2)-OMB(3)*A(2,1)+OMB(1)*A(2,3)
      ADOT(2,3)=-OMC(3)*A(1,3)+OMC(1)*A(3,3)+OMB(2)*A(2,1)-OMB(1)*A(2,2)
      ADOT(3,1)=+OMC(2)*A(1,1)-OMC(1)*A(2,1)+OMB(3)*A(3,2)-OMB(2)*A(3,3)
      ADOT(3,2)=+OMC(2)*A(1,2)-OMC(1)*A(2,2)-OMB(3)*A(3,1)+OMB(1)*A(3,3)
      ADOT(3,3)=+OMC(2)*A(1,3)-OMC(1)*A(2,3)+OMB(2)*A(3,1)-OMB(1)*A(3,2)
C
C     THE 2ND CELL, -[ACBDOT][LTB], IS OBTAINED BY USING THE 1ST CELL.
C
      ADOT(1,4) = -ADOT(1,2)*LABB(3) + ADOT(1,3)*LABB(2)
      ADOT(1,5) = +ADOT(1,1)*LABB(3) - ADOT(1,3)*LABB(1)
      ADOT(1,6) = -ADOT(1,1)*LABB(2) + ADOT(1,2)*LABB(1)
      ADOT(2,4) = -ADOT(2,2)*LABB(3) + ADOT(2,3)*LABB(2)
      ADOT(2,5) = +ADOT(2,1)*LABB(3) - ADOT(2,3)*LABB(1)
      ADOT(2,6) = -ADOT(2,1)*LABB(2) + ADOT(2,2)*LABB(1)
      ADOT(3,4) = -ADOT(3,2)*LABB(3) + ADOT(3,3)*LABB(2)
      ADOT(3,5) = +ADOT(3,1)*LABB(3) - ADOT(3,3)*LABB(1)
      ADOT(3,6) = -ADOT(3,1)*LABB(2) + ADOT(3,2)*LABB(1)
C
C * * * * *
C     ALL MATRICES NEEDED TO CALCULATE [FA] HAVE NOW BEEN
C        SET UP.  PERFORM THE REQUIRED OPERATIONS.
C
```

```
      CALL MMUL (A,MINV,AMINV,3,12,12)
      CALL MMUL (AMINV,AT,AMINVA,3,12,3)
      CALL MAINV3 (AMINVA,TERMA1)
C
      CALL MATVEC (AMINV,F,AMINVF,3,12)
      CALL MATVEC (ADOT,OMEGA,ADOTOM,3,12)
C
C     INSERT THE MINUS SIGN WHILE COMBINING [AMINVF] AND [ADOTOM].
      DO 563 I=1,3
  563 TERMV1(I) = -AMINVF(I) - ADOTOM(I)
C
      CALL MATVEC (TERMA1,TERMV1,TERMV2,3,3)
C
C     FINALLY, PERFORM THE STEP IN WHICH [FA], A 12X1 MATRIX, IS
C        CALCULATED.
      CALL MATVEC (AT,TERMV2,FA,12,3)
C
C * * * * * * * * * *
C     THE CALCULATION OF [FA] COMPLETES THE DETERMINATION OF ALL THE
C        FORCES AND MOMENTS AT THE C.M.'S OF THE SRB AND PARACHUTE.
C        ADD [F] AND [FA] TO GET [FTOTAL].
      DO 570 I=1,12
  570 FTOTAL(I) = F(I) + FA(I)
C
C * * * * * * * * * * *
C     CALCULATE ALL ACCELERATIONS IN THE 'B' AND 'C' FRAMES.
C             [DOMEGA]  =  [MINV][FTOTAL]
C
      CALL MATVEC (MINV,FTOTAL,DOMEGA,12,12)
C * *
C     SET UP THE 9X1 ACCELERATION MATRIX THAT IS TO BE INTEGRATED.
      DO 590 I=1,3
      DOM9(I)   = DOMEGA(I)
      DOM9(I+3) = DOMEGA(I+3)
  590 DOM9(I+6) = DOMEGA(I+9)
C
C     END OF INTEGRATION LOOP.
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

      IF(KUTTA .LT. 4) GO TO 400
      KUTTA=0
C
C * * * * * * *
C     SET FLAGS TO CONTROL OUTPUT SECTION OF PROGRAM AND CUT-OFF.
C
      IF(TIME .GE. TPRINT) IPRFLG=1
      IF(TIME .GE. TMAX)   ICOFLG=1
      IF(IPRFLG .EQ. 1) GO TO 701
      IF(ICOFLG .EQ. 1) GO TO 701
C     IF ALL FLAGS = 0, GO TO START OF INTEGRATION LOOP.
      GO TO 400
  701 CONTINUE
C
C * * * * * * *
```

```
C
C       CALCULATE OUTPUT PARAMETERS FOR PLOTTING.
C
C * * * *
C       CALC. THE 3-2-1 EULER ANGLES FOR THE ORIENTATIONS OF THE SRB AND
C          PARACHUTE.
        ST2B   =  -ABI(1,3)
        THETB = ASIN(ST2B)*CNV
        PSIB =ZTAN2(ABI(1,2),ABI(1,1))*CNV
        PHIB =ZTAN2(ABI(2,3),ABI(3,3))*CNV
C
        ST2C   =  -ACI(1,3)
        THETC = ASIN(ST2C)*CNV
        PSIC =ZTAN2(ACI(1,2),ACI(1,1))*CNV
        PHIC =ZTAN2(ACI(2,3),ACI(3,3))*CNV
C * * *
C       CALC. FLIGHT PATH ANGLE AND AZIMUTH OF THE SRB AND PARACHUTE.
C
        GAMIB = ZSIN(-VIBI(3),VIB)*CNV
        AZIIB = ZTAN2(VIBI(2),VIBI(1))*CNV
        GAMIC = ZSIN(-VICI(3),VIC)*CNV
        AZIIC = ZTAN2(VICI(2),VICI(1))*CNV
C * * *
C       SET UP [AIV] TO CALCULATE THE SRB BANK ANGLE.
C
        SAZB = SIN(AZIIB/CNV)
        CAZB = COS(AZIIB/CNV)
        SGMB = SIN(GAMIB/CNV)
        CGMB = COS(GAMIB/CNV)
        AIV(1,1) = CAZB*CGMB
        AIV(2,1) = SAZB*CGMB
        AIV(3,1) =-SGMB
        AIV(1,2) =-SAZB
        AIV(2,2) = CAZB
        AIV(3,2) = 0.0
        AIV(1,3) = CAZB*SGMB
        AIV(2,3) = SAZB*SGMB
        AIV(3,3) = CGMB
        CALL MMUL (ABI,AIV,ABV,3,3,3)
        BANKB = ZTAN2(ABV(1,2),-ABV(1,3))*CNV
C * * *
C       SET UP ERROR INDICATORS.
C
        CALL ORTH (ABI,ORTABI)
        CALL ORTH (ACI,ORTACI)
        CALL MATVEC (A,DOMEGA,ADOMEG,3,12)
C * * *
C       CALC. VECTOR MAGNITUDES; CONVERT ANG. VELOCITIES TO DEG/SEC.
C
        CALL VMAG (VABB,DUMM,VAB)
        CALL VMAG (RIBI,DUMM,RIB)
        CALL VMAG (RABI,DUMM,RAB)
        CALL VMAG (RICI,DUMM,RIC)
        CALL VMAG (GB,DUMM,GMAG)
        PDEGB = OMB(1)*CNV
```

```
      QDEGB = OMB(2)*CNV
      RDEGB = OMB(3)*CNV
      PDEGC = OMC(1)*CNV
      QDEGC = OMC(2)*CNV
      RDEGC = OMC(3)*CNV
C * * *
C     SORT OUT THE PRINT CONTROL TIME LOGIC.
C
      IF(IPRFLG .EQ. 0) GO TO 740
      IPRFLG = 0
      TPRINT = TPRINT + DTPRT
      IF(TIME .LT. TNP1) GO TO 720
      IF(TIME .LT. TNP2) GO TO 740
  720 CONTINUE
C * * PRINT THE STANDARD SET OF PARAMETERS.
      WRITE (6,858)
      WRITE (6,861) TIME,ALTB,QBRB,VIB,ALPHAB,PHIAB,PDEGB,QDEGB,RDEGB,
     2GAMIB,AZIIB,RIBI(1),RIBI(2),PSIB,THETB,PHIB
      WRITE (6,862)       ALTC,QBRC,VIC,ALPHAC,PHIAC,PDEGC,QDEGC,RDEGC,
     2GAMIC,AZIIC,RICI(1),RICI(2),PSIC,THETC,PHIC
      WRITE (6,871)
      DO 730 I=1,12
  730 WRITE (6,808) FL(I),FG(I),FW(I),FA(I),FTOTAL(I),DOMEGA(I),OMEGA(I)
      WRITE (6,879) ORTABI,ORTACI
C
C * * BEGIN DIAGNOSTIC PRINT, IF DESIRED.
      IF(IDIAG .LT. 1) GO TO 739
      WRITE (6,889)
      WRITE (6,809) VIB,VAB,VIC,RIB,RAB,RIC
      WRITE (6,809) RHO,QBRSB,QBRSC,CAMB,CNMB,CMMB,CAMC,CNMC,CMMC
      WRITE (6,808) GMAG,MASSB,MASSC,BANKB
      DO 733 I=1,3
  733 WRITE (6,812) (A(I,J),J=1,12)
      DO 734 I=1,3
  734 WRITE (6,812) (ADOT(I,J),J=1,12)
      WRITE (6,808) ADOMEG(1),ADOTOM(1),ADOMEG(2),ADOTOM(2),
     2ADOMEG(3),ADOTOM(3)
  739 WRITE (6,802)
  740 CONTINUE
C
C     END OF OUTPUT SECTION.
C * * * * * * *
      IF(ICOFLG .EQ. 0) GO TO 400
C     END OF RUN.
      STOP
  800 FORMAT (1H1)
  801 FORMAT (16A4,I1,I2)
  802 FORMAT (1H0,16A4,I16,2I5)
  805 FORMAT (I15,I5,2X,4E11.8)
  807 FORMAT (7E11.8)
  808 FORMAT (1X,F18.9,5F19.9,F18.9)
  809 FORMAT (4X,9E14.7)
  812 FORMAT (1X,11E11.4,E10.3)
  820 FORMAT (20A4)
  822 FORMAT (2X,20A4)
```

```
858 FORMAT      (3X,4HTIME,6X,3HALT,3X,4HQBAR,3X,2HVI,5X,5HALPHA,4X,
   B4HPHIA,5X,1HP,6X,1HQ,6X,1HR,5X,5HGAMMA,3X,7HAZIMUTH,5X,2HXI,7X,
   C2HYI,8X,18H3-2-1 EULER ANGLES)
861 FORMAT (F8.3,F9.1,F6.1,F7.2,3F8.2,2F7.2,2F9.3,2F9.1,F9.2,2F8.2)
862 FORMAT    (8X,F9.1,F6.1,F7.2,3F8.2,2F7.2,2F9.3,2F9.1,F9.2,2F8.2)
871 FORMAT (10X,2HFL,17X,2HFG,17X,2HFW,17X,2HFA,17X,6HFTOTAL,13X,
   B6HDOMEGA,12X,5HOMEGA)
879 FORMAT   (85X,16HERROR INDICATORS,2F15.10)
889 FORMAT   (6X,11HDIAGNOSTICS)
    END
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1992	Technical Paper

4. TITLE AND SUBTITLE

Technique To Eliminate Computational Instability in Multibody Simulations Employing the Lagrange Multiplier

5. FUNDING NUMBERS

6. AUTHOR(S)

G. Watts

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812

8. PERFORMING ORGANIZATION REPORT NUMBER

M-687

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TP-3220

11. SUPPLEMENTARY NOTES

Prepared by Structures and Dynamics Laboratory, Science and Engineering Directorate.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified — Unlimited
Subject Category: 61

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This paper presents a programming technique to eliminate computational instability in multibody simulations that use the Lagrange multiplier. The computational instability occurs when the attached bodies drift apart and violate the constraints. The programming technique uses the constraint equation, instead of integration, to determine the coordinates that are not independent. Although the equations of motion are unchanged, a complete derivation of the incorporation of the Lagrange multiplier into the equation of motion for two bodies is presented. A listing of a digital computer program which uses the programming technique to eliminate computational instability is also presented. The computer program simulates a solid rocket booster and parachute connected by a frictionless swivel.

14. SUBJECT TERMS

Computational Instability, Multibody Simulation, Lagrange Multiplier, Computer Programming Techniques, Parachute Dynamics

15. NUMBER OF PAGES

32

16. PRICE CODE

A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited