

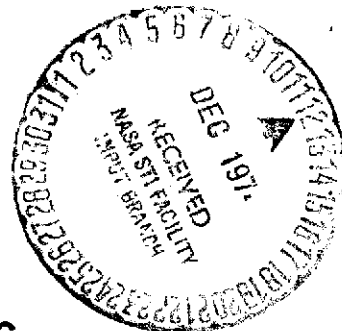
NASA CR-139156

**USERS' GUIDE**  
**TO**  
**FOUR-BODY AND THREE-BODY TRAJECTORY**  
**OPTIMIZATION PROGRAMS**  
  
 by  
  
**C. L. Pu and T. N. Edelbaum**

(NASA-CR-139156) USER'S GUIDE TO N75-12020  
 FOUR-BODY AND THREE-BODY TRAJECTORY  
 OPTIMIZATION PROGRAMS (Massachusetts Inst.  
 of Tech.) 227 p HC \$7.50 CSCL 22A Unclas  
G3/12 03892



**The Charles Stark Draper Laboratory, Inc.**  
 Cambridge, Massachusetts 02139



USERS' GUIDE

to

Four-Body and Three-Body Trajectory  
Optimization Programs

by

C. L. Pu and T. N. Edelbaum

Prepared under NASA Contract NAS 5-11984

October 1974

## Table of Contents

	Page
Abstract	iv
Introduction	1
Program Organization	2
Coordinates Systems	5
Variable Step Integration	10
Correction of Errors by Quadrature Formula	13
Input Parameters	14
Program Descriptions	
Program TRAJ (TRAJ3)	17
Program EXLAM (EXLAM3)	18
Program ETP2I (ETP2I3)	19
Program PTP3I (PTP3I3)	22
Input Data Decks	25
Program Printout Description	33
Subroutine Descriptions	
FOURBY	42
THRBDY	46
TWOBDY	49
CSTEP	50
CSTEP3	53
DELRV	54
DELRV3	56
COMIC	57
COMIC3	59
COMFG	60
COMFG3	64
COMAUG	66
CTAR	67
CTAR3	69
COMDX	71
RVEMV	72
UPX	73
LAMB	74
LAMB3	76
COMF	78
COMF3	81
COMG	84
PVEC	86
DISP	88

REPRODUCIBILITY OF THE  
ORIGINAL PAGE IS POOR

DISP3	91
PTRAJ	94
PTRAJ3	95
FDATA	96
FDATA3	97
Trouble Shooting	98
Examples	100
Program and Subroutine Listings	134
Reference	222

## ABSTRACT

This users' guide documents a collection of computer programs and subroutines written in FORTRAN to calculate 4-body (sun-earth-moon space) and 3-body (earth-moon space) optimal trajectories. The programs incorporate a new variable step integration technique and a new quadrature formula to correct single step errors. These new features have resulted in significant improvement in efficiency and accuracy.

The programs provide capability to solve initial value problem, two point boundary value problem of a transfer from a given initial position to a given final position in fixed time, optimal 2-impulse transfer from an earth parking orbit of given inclination to a given final position and velocity in fixed time and optimal 3-impulse transfer from a given position to a given final position and velocity in fixed time.

## INTRODUCTION

This users' guide documents a collection of computer programs and sub-routines written in FORTRAN to calculate 4-body optimal trajectories under the combined influence of the sun, earth, and moon. Additional programs and sub-routines have been provided so that 3-body optimal trajectories can be computed without the influence of the sun.

The method used in the extrapolation of the state vector and the solution of boundary value problems is essentially the same as presented in Ref. (1) and (2). However, the present programs incorporate a new variable step integration technique and a new quadrature formula to correct single step errors. These changes have resulted in significant improvement in efficiency and accuracy. The 4-body as well as the 3-body state extrapolation subroutine automatically determines the step size that will yield a fixed estimate of the single step position and velocity errors. The quadrature formula developed by T. N. Fdelbaum is based on the 4th derivatives of the position error estimates at the beginning of a step and the second derivatives of the Stumpff-Weiss position errors at the end of the step.

In addition, two subroutines have been provided to compute target position and velocity in a Halo orbit about the  $L_1$  libration point on the sun-earth line or one of the libration points on the earth-moon line. State vectors are now available also in rotating frames similar to the familiar barycentric frame commonly used in the restricted 3-body problem. These new features will be described in some detail.

## PROGRAM ORGANIZATION

The 4-body and 3-body programs consist of 4 options. The name of a 3-body program is given in parenthesis following the name of a corresponding 4-body program.

- Option 1: Program TRAJ (TRAJ3)  
This option is a trajectory integration program to solve initial value problems.
- Option 2: Program EXLAM (EXLAM3)  
This option is a program to solve two point boundary value problem of a transfer from a given initial position to a given final position in fixed time using Newton-Raphson method.
- Option 3: Program ETP2I (ETP2I3)  
This option determines the optimal 2-impulse transfer from an earth parking orbit of given inclination to a given final position and velocity in fixed time. It employs an accelerated gradient projection method using Davidon's Variance algorithm.
- Option 4: Program PTP3I (PTP3I3)  
This option computes the optimal 3-impulse transfer from a given initial position to a given final position and velocity in fixed time. It uses an accelerated gradient method (Davidon's Variance algorithm) in the outer-loop and Newton-Raphson method in the second leg of the inner loop.

These 4 main programs call one or more of the following subroutines.

<u>Subroutine</u>	<u>Function</u>
FOURBY (THREBY)	State extrapolation.
TWOBY	2-body state extrapolation.
CSTEP (CSTEP3)	Computes single step size and error estimate at beginning of step.
DELRV (DELRV3)	Computes corrections by quadrature formula.
LAMB (LAMB3)	Solves Lambert problem.
CTAR (CTAR3)	Computes target position and velocity on a Halo orbit about a libration point. Called by ETP2I (ETP2I3).
COMIC (COMIC3)	Transforms input parameters with respect to earth into states with respect to sun (4-body) or earth (3-body).
COMPG (COMPG3)	Computes function and gradient in ETP2I (ETP2I3).

COMAUG	Forms augmented function and gradient in ETP21 (ETP213)
COMDX	Computes changes in independent variables in ETP21 (ETP213).
COMF (COMF3)	Computes function in PTP31 (PTP313).
COMG	Computes gradient in PTP31 (PTP313).
PVEC	Advances primer vectors and monitor its magnitude in FOURBY (THRBDY).
RVEMV	Computes state vectors in FOURBY. Not used in 3-body.
UPX	Updates variables and functions in ETP21 (ETP213).
DISP (DISP3)	Computes states in rotating coordinates for display.
PTRAJ (PTRAJ3)	Prints trajectory.
FDATA (FDATA3)	Files data for further processing outside of the program.

The main programs and the above mentioned subroutines will be described in some detail in the sections to follow. In addition, there are a number of service subroutines, which performs standard mathematical operations.

#### Service Subroutines

#### Function

MXV	Multiplication of a vector by a matrix, $\bar{a} = M \bar{b}$ .
VVT	Outer product of a vector, $M = \bar{a} \bar{a}^T$ .
DOT	Dot product of 2 vectors, $\bar{c} = \bar{a}^T \bar{b}$ .
VMAG	Magnitude of a vector.
INVERT	Inversion of a matrix
UNITV	Unit vector of a vector
VXV	Cross product of 2 vectors, $\bar{a} = \bar{b} \times \bar{c}$ .
MTRANS	Transpose a matrix, $N = M^T$ .
MXM	Multiplication of 2 matrices, $C = MN$

These service subroutines will be listed but not discussed further.

A tabulation of the subroutines called by the main programs is given in Table 1.



Table 1 PROGRAMS AND SUBROUTINES

Program Sub-routine	TRAJ (TRAJ3)	EXLAM (EXLAM3)	ETP2I (ETP2I3)	PTP3I (PTP3I3)
FOURBY(THRBDY)	X	X	X	X
TWOBDY	X	X	X	X
CSTEP (CSTEP3)	X	X	X	X
DELRV (DELRV3)	X	X	X	X
COMIC (COMIC3)	X	X	X	
COMFG (COMFG3)			X	
COMF (COMF3)				X
COMG				X
LAMB (LAMB3)		X		X
COMAUG			X	
COMDX			X	
CTAR (CTAR3)			X	
UPX			X	
(1) RVEMV	X	X	X	X
PVEC	X	X	X	X
DISP (DISP3)	X	X	X	X
PTRAJ (PTRAJ3)	X	X	X	X
FDATA (FDATA3)	X	X	X	X
MXV	X	X	X	X
VVT	X	X	X	X
DOT	X	X	X	X
VMAG	X	X	X	X
INVERT		X	X	X
UNITV	X	X	X	X
VXV	X	X	X	X
MTRANS			X	X
MXM	X	X	X	X

Note: 1 Not used in 3-body programs.

## COORDINATES SYSTEMS

The position or velocity vector of a body with respect to another body may be expressed in different coordinates systems at different stages of computation. These coordinates systems are described below.

### Coordinate Systems

### Definition

S-frame (Basic computation frame in 4-body programs)	An inertial frame centered at the sun in which state extrapolation is performed. Since it is likely that input ephemerides will be obtained from JPL tapes, the $X_S$ - $Y_S$ plane will be the Ecliptic-Equinox plane of 1950.0, with the $X_S$ -axis pointing in the Equinox direction.
E-frame (4-body)	An inertial frame centered at the earth parallel to S-frame.
M-frame (4-body)	An inertial frame centered at the moon parallel to S-frame.
e-frame (Basic computation frame in 3-body programs)	An inertial frame centered at the earth. The $X_e$ - $Y_e$ plane is the Equator-Equinox plane of 1950.0 with $X_e$ pointing in the Equinox direction.
m-frame (3-body)	An inertial frame centered at the moon parallel to e-frame.
L-frame (4-body)	A rotating frame centered at the $L_1$ libration point used to define a target on a Halo orbit about $L_1$ . The $X_L$ -axis is along the line from the sun to the earth-moon barycenter. The $Z_L$ -axis is along the total angular momentum vector of the earth and moon about the sun.
$l$ -frame (3-body)	A rotating frame centered at either $L_1$ or $L_2$ libration point to define a target on a Halo orbit. The $X_l$ -axis is along the line from the earth to the moon. The $Z_l$ -axis is along the angular momentum vector of the moon about the earth.

O-frame	An inertial frame centered at the earth. The $X_O$ - $Y_O$ plane is the parking orbit plane of an inclination $i$ . The $X_O$ -axis is along the line of ascending node.
D-frame (4-body)	A rotating frame centered at the earth used for display purpose. This frame is the equivalent of the rotating barycentric frame in a restricted 3-body problem. The $X_D$ -axis is along the line from the sun to the earth. The $Z_D$ -axis is along the angular momentum vector of the earth about the sun.
d-frame (3-body)	A rotating frame centered at the moon used for display purpose. The $X_d$ -axis is along the line from the earth to the moon. The $Z_d$ -axis is along the angular momentum vector of the moon about the earth.

These frames are shown in Figure 1.

In general, when there is no ambiguity, a vector of a body A with respect to another body B will be written with subscripts BA. For example,  $\bar{R}_{SV}$  will be a position vector of the vehicle with respect to the sun in the S-frame. Sometimes, it is necessary to add a superscript to indicate the frame in which the components are resolved. Thus,  $\bar{\omega}_{SL}^S$  is the angular velocity of the L-frame with respect to the S-frame with components in the S-frame.

A transformation matrix will be denoted by a subscript and a superscript. It will transform a vector from the frame indicated by the subscript to the frame indicated by the superscript. For example  $\bar{R}^L = C_S^L \bar{R}^S$  transforms  $\bar{R}$  in S-frame to  $\bar{R}$  in L-frame.

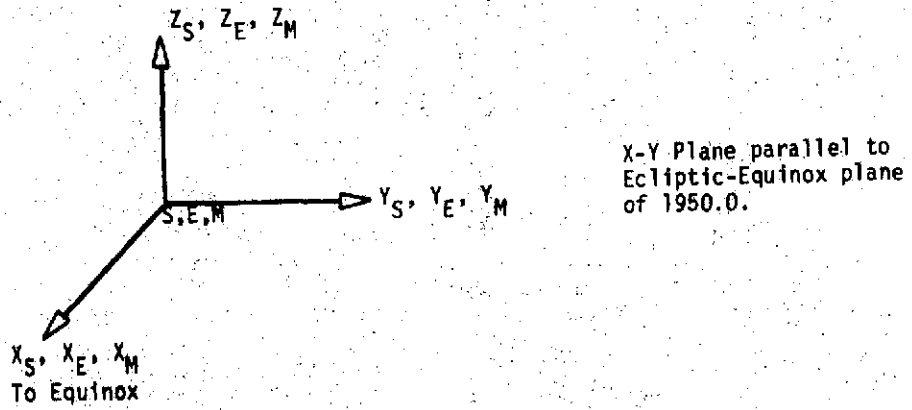


Figure 1a Inertial S, E, and M Frames

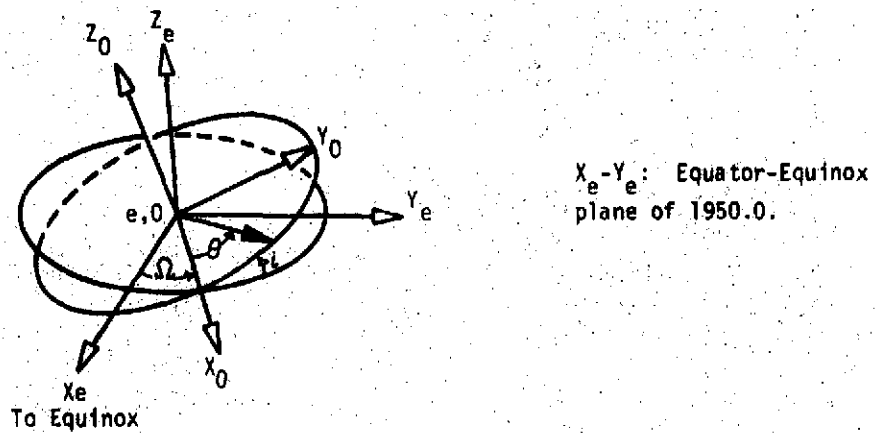


Figure 1b Inertial e and O Frames

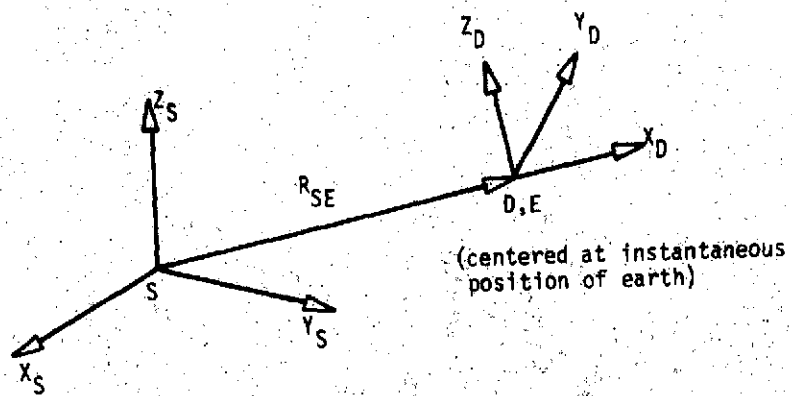


Figure 1c D-Frame

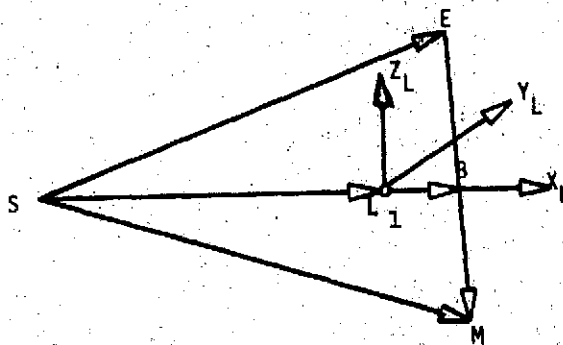


Figure 1d L-Frame

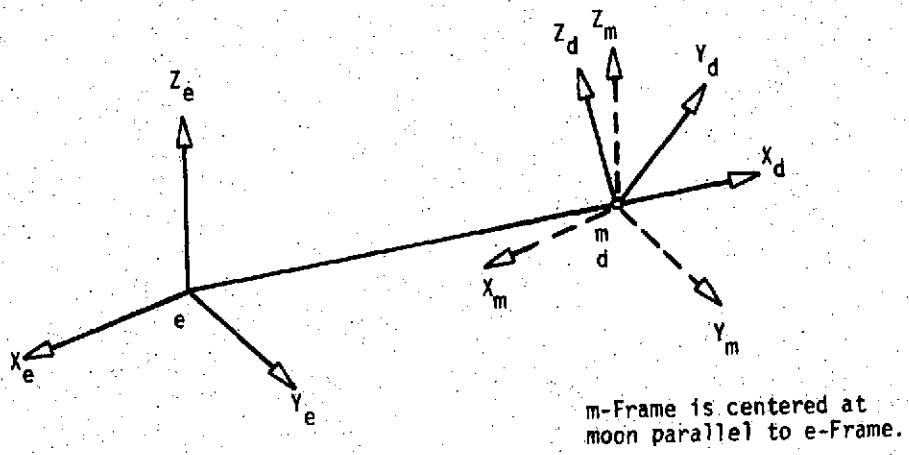


Figure 1e m and d Frames

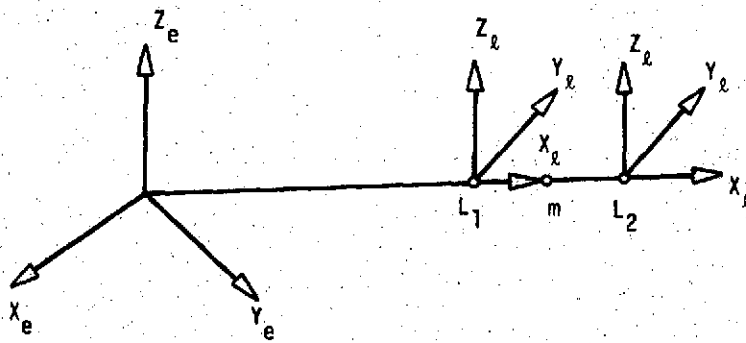


Figure 1f l Frame

## VARIABLE STEP INTEGRATION

The variable step integration technique used in subroutine FOURBY (THRBDY) is summarized here. The computation of step size and estimates of position errors, which is performed in subroutine CSTEP (CSTEPS), is an extension of D'Amario's derivation for the 3-body problem (Ref. 4). Let

$\bar{R}'(t+h)$  = approximate vehicle position determined by Stumpff-Weiss method

$\bar{R}(t+h)$  = true vehicle position

The position error at the end of a single step is

$$\bar{\epsilon}(t+h) = \bar{R}(t+h) - \bar{R}'(t+h) \quad (1)$$

Expanding Eq. (1) in Taylor series at  $t$  we obtain

$$\bar{\epsilon}(t+h) = \bar{\epsilon}(t) + \dot{\bar{\epsilon}}(t)h + \frac{1}{2}\ddot{\bar{\epsilon}}(t)h^2 + \frac{1}{6}\ddot{\bar{\epsilon}}(t)h^3 + \frac{1}{24}\overset{\text{IV}}{\bar{\epsilon}}(t)h^4 + O(h^5) \quad (2)$$

By definition,

$$\bar{\epsilon}(t) = 0 \quad (3)$$

It can be shown that

$$\dot{\bar{\epsilon}}(t) = \ddot{\bar{\epsilon}}(t) = \overset{\text{III}}{\bar{\epsilon}}(t) = 0 \quad (4)$$

The first non-zero term is the 4th derivative term. Neglecting all higher derivative terms the position error is given by

$$\bar{\epsilon}(t+h) \approx \overset{\text{IV}}{\bar{\epsilon}}(t) \frac{h^4}{24} \quad (5)$$

Let

$$\begin{aligned} |\bar{\epsilon}(t+h)| &= \epsilon_{\max} \\ &= \text{allowable single step position error} \end{aligned} \quad (6)$$

Then, an estimate of step size is given by

$$h = \left( 24 \frac{\epsilon_{\max}}{\overset{\text{IV}}{\bar{\epsilon}}(t)} \right)^{1/4} \quad (7)$$

In CSTEP we compute  $\overset{\dots}{\epsilon}_{RSV}$  to estimate step size  $h$  and also  $\overset{\dots}{\epsilon}_{RSE}$  and  $\overset{\dots}{\epsilon}_{RSM}$  for use in DELRV to compute error corrections. If we make the following definitions in a 4-body space,

$$\begin{aligned}
 G_{SV} &= \frac{\partial}{\partial \bar{R}_{SV}} \left[ -\mu_S \frac{\bar{R}_{SV}}{R_{SV}^3} \right] = \frac{\partial}{\partial \bar{R}_{SV}} \left[ \ddot{\bar{R}}_{SV} \right] \\
 G_{EV} &= \frac{\partial}{\partial \bar{R}_{EV}} \left[ -\mu_E \frac{\bar{R}_{EV}}{R_{EV}^3} \right] = \frac{\partial}{\partial \bar{R}_{EV}} \left[ \ddot{\bar{R}}_{EV} \right] \\
 G_{SE} &= \frac{\partial}{\partial \bar{R}_{SE}} \left[ -(\mu_S + \mu_E) \frac{\bar{R}_{SE}}{R_{SE}^3} \right] = \frac{\partial}{\partial \bar{R}_{SE}} \left[ \ddot{\bar{R}}_{SE} \right] \\
 G_{MV} &= \frac{\partial}{\partial \bar{R}_{MV}} \left[ -\mu_M \frac{\bar{R}_{MV}}{R_{MV}^3} \right] = \frac{\partial}{\partial \bar{R}_{MV}} \left[ \ddot{\bar{R}}_{MV} \right] \\
 G_{SM} &= \frac{\partial}{\partial \bar{R}_{SM}} \left[ -(\mu_S + \mu_M) \frac{\bar{R}_{SM}}{R_{SM}^3} \right] = \frac{\partial}{\partial \bar{R}_{SM}} \left[ \ddot{\bar{R}}_{SM} \right]
 \end{aligned} \tag{8}$$

$$\begin{aligned}
 \delta \bar{g}_{SV} &= \ddot{\bar{R}}_{SV} - \left[ \ddot{\bar{R}}_{SV} \right] \\
 \delta \bar{g}_{EV} &= \ddot{\bar{R}}_{EV} - \left[ \ddot{\bar{R}}_{EV} \right] \\
 \delta \bar{g}_{SE} &= \ddot{\bar{R}}_{SE} - \left[ \ddot{\bar{R}}_{SE} \right] \\
 \delta \bar{g}_{MV} &= \ddot{\bar{R}}_{MV} - \left[ \ddot{\bar{R}}_{MV} \right] \\
 \delta \bar{g}_{SM} &= \ddot{\bar{R}}_{SM} - \left[ \ddot{\bar{R}}_{SM} \right]
 \end{aligned} \tag{9}$$

we can express the estimated errors by



$$\begin{aligned}
\bar{\epsilon}_{RSV}(t+h) &= \left\{ G_{SV} \delta \bar{g}_{SV} + G_{EV} \delta \bar{g}_{EV} + \frac{\mu_E}{\mu_S + \mu_E} G_{SE} \delta \bar{g}_{SE} \right. \\
&\quad \left. + G_{MV} \delta \bar{g}_{MV} + \frac{\mu_M}{\mu_S + \mu_M} G_{SM} \delta \bar{g}_{SM} \right\} \frac{h^4}{24} + O(h^5) \\
\bar{\epsilon}_{RSE}(t+h) &= \left\{ G_{SE} \delta \bar{g}_{SE} - \frac{\mu_M}{\mu_E + \mu_M} G_{EM} \delta \bar{g}_{EM} \right. \\
&\quad \left. + \frac{\mu_M}{\mu_S + \mu_M} G_{SM} \delta \bar{g}_{SM} \right\} \frac{h^4}{24} + O(h^5) \\
\bar{\epsilon}_{RSM}(t+h) &= \left\{ G_{SM} \delta \bar{g}_{SM} + \frac{\mu_E}{\mu_E + \mu_M} G_{EM} \delta \bar{g}_{EM} \right. \\
&\quad \left. + \frac{\mu_E}{\mu_S + \mu_E} G_{SE} \delta \bar{g}_{SE} \right\} \frac{h^4}{24} + O(h^5)
\end{aligned} \tag{10}$$

In CSTEP3 we compute  $\bar{\epsilon}_{REV}^{\dots}$  to determine h and the error may be expressed by

$$\bar{\epsilon}_{REV}(t+h) = \left\{ G_{EV} \delta \bar{g}_{EV} + G_{MV} \delta \bar{g}_{MV} \right\} \frac{h^4}{24} + O(h^5) \tag{11}$$

Note that  $\bar{\epsilon}_{REM}^{\dots} = 0$  in a 3-body space.

## CORRECTION OF ERRORS BY QUADRATURE FORMULAS

The quadrature formulas proposed by T. N. Edelbaum and used in subroutine DELRV (DELRV3) to compute a single step position and velocity corrections are presented here.

Assume the second derivative of the position error in a single step is given by

$$\ddot{\epsilon} = \frac{t^2}{2} (\bar{a} + \bar{b}t) \quad (12)$$

where

$$\bar{a} = \ddot{\epsilon}(0) = \ddot{\epsilon}_0 \quad (13)$$

Then the second derivative of the position error at the end of a single step is given by

$$\ddot{\epsilon}_1 = \frac{h^2}{2} (\bar{a} + \bar{b}h) \quad (14)$$

Solving Eq. (14) and using Eq. (13),

$$\bar{b} = \frac{2\ddot{\epsilon}_1}{h^3} - \frac{\ddot{\epsilon}_0}{h} \quad (15)$$

Integrating Eq. (12)

$$\dot{\epsilon} = \frac{\bar{a}t^3}{6} + \frac{\bar{b}t^4}{8} \quad (16)$$

$$\epsilon = \frac{\bar{a}t^4}{24} + \frac{\bar{b}t^5}{40} \quad (17)$$

Using Eqs.(13) and (15) in (16) and (17), we obtain the following corrections at the end of a single step,

$$\dot{\epsilon}_1 = \frac{h}{4} \left( \frac{\ddot{\epsilon}_0}{h} \frac{h^2}{6} + \ddot{\epsilon}_1 \right) \quad (18)$$

$$\epsilon_1 = \frac{h^2}{20} \left( \frac{\ddot{\epsilon}_0}{h} \frac{h^2}{3} + \ddot{\epsilon}_1 \right) \quad (19)$$

$\ddot{\epsilon}_0$  is the 4th derivative of position error estimate at the beginning of a step and  $\ddot{\epsilon}_1$  is the second derivative of Stumpff-Weiss error estimate at the end of the step.

## INPUT PARAMETERS

The following input parameters are required for all 4 options.

<u>Parameters</u>	<u>Symbol or Definition</u>	<u>Suggested Value</u>
<u>4-Body Programs:</u>		
GL1	$\gamma_{L1}$	$1.001098 \times 10^{-2}$
AUM	1 unit of distance in meters	$1.49597893 \times 10^{11}$ m
UTIME	1 unit of time in days	58.1323577631 days
UVELM	1 m/s in dimensionless velocity	$3.35742409867 \times 10^{-5}$
MS	$\mu_S$ (sun)	$9.99996959568 \times 10^{-1}$
ME	$\mu_E$ (earth)	$3.00348453188 \times 10^{-6}$
MM	$\mu_M$ (moon)	$3.69431224671 \times 10^{-8}$
RSEO	$\overline{R_{SE}}(t_o)$	
VSEO	$\overline{V_{SE}}(t_o)$	
RSMO	$\overline{R_{SM}}(t_o)$	
VSMO	$\overline{V_{SM}}(t_o)$	
<u>3-Body Programs (Ref. 4 and 5)</u>		
GAMMA	$\gamma$	$\gamma_{L1} = 0.150935$ $\gamma_{L2} = -0.167833$
UDM	1 unit of distance in meters	$3.8441 \times 10^8$ m
UTIME	1 unit of time	104.362/24 days
UVELM	1 m/s in dimensionless velocity	1/1023.17 m/s

ME	$\mu_E$ (earth)	.9878493
MM	$\mu_M$ (moon)	.0121507
REMO	$\overline{R}_{EM}(t_0)$	
VEMO	$\overline{V}_{EM}(t_0)$	

Common to both 4- and 3-Body Programs:

IPV	Flag to compute primer vector	IPV = 0, no 1, yes
IPTRAJ (IPTJX)	Flag to print trajectory each time step	IPTRAJ (IPTJX) = 0, no 1, yes
IFILEX (IFILE)	Flag to file data each time step	IFILEX (IFILE) = 0, no 1, yes
ERRMXM	Allowable single step position error in meters	150 m or higher
ERRMAX	Dimensionless ERRMXM	$10^{-9}$ or higher
TDAY0	Starting time in days from reference date	
TRIPD	Trip time in days	

The following input parameters are required for Davidson's variance algorithm used in ETP2I (ETP2I3) and PTP3I (PTP3I3). See Ref. 2, Appendix A.

EPS	$\epsilon$	$10^{-11}$ for ETP2I (ETP2I3) $10^{-3}$ for PTP3I (PTP3I3)
-----	------------	---

V

V

ICOMV

EPSV

Davidon's Variance matrix.

If ICOMV = 1, this input is not needed and the program will compute an estimate. Once a working V matrix has been developed through iterations, it should be used in a new run to continue iteration.

Flag to compute V.

ICOMV = 0, no (input V)

1, yes

Small number to scale down

V matrix if ICOMV = 1. It specifies the magnitude of the first step in the gradient direction. A typical value is about  $10^{-5}$ .

PROGRAM TRAJ (TRAJ3)

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition/Value</u>
IMODE		Mode to select input IMODE = 1, 2, or 3
	<u>IMODE = 1</u>	
REVMAG	$ \bar{R}_{EV}(t_o) $	$4.38739157152 \times 10^{-5}$ for 100 n. m. parking orbit.
VEVMAG	$ \bar{V}_{EV}^+(t_o) $	Estimate of departure velocity magnitude with respect to earth.
OINCD	$i$ (deg)	Inclination of earth parking orbit.
OBLD (not used in 3-body)	OBL (deg)	Obliquity angle = 23.44578743018259 deg between E-frame and e-frame.
THED	$\theta$ (deg)	Position of vehicle from ascending node in orbital plane.
LOND	$\Omega$ (deg)	Longitude of ascending node.
	<u>IMODE = 2</u>	
REVO	$\bar{R}_{EV}(t_o)$	
VEVOP	$\bar{V}_{EV}^+(t_o)$	
	<u>IMODE = 3</u>	
RSVO	$\bar{R}_{SV}(0)$	
VSVI	$\bar{V}_{SV}^+(0)$	

B. Computation

Inputs of IMODE = 1 or 2 are converted into position and velocity of vehicle in S-frame or e-frame. The initial states are propagated to the final time using subroutine FOURBY (THRBDY).

C. Output Parameters

Terminal states are printed out. States at each time step will be printed out if IPTRAJ = 1 and they will be filed if IFILE = 1.

PROGRAM EXLAM (EXLAM3)

A. Input Parameters

<u>Parameters</u>	<u>Symbol/ Unit</u>	<u>Definition/ Value</u>
IMODE		Mode to select input IMODE = 1 or 2
Same inputs for IMODE = 1, 2 as shown under PROGRAM TRAJ (TRAJ3)		
ERRMIN	$\epsilon_{MIN}$	Allowable error for stopping Newton-Raphson method ( $10^{-10}$ )
KNR		Initial estimate of change in initial velocity magnitude (1)
ITLMAX		Maximum number of iterations (50)
RSVTAR	$\vec{R}_{SV}(t_f)$	Target position
IPTJX		Flag to compute and print trajectory after convergence IPTJX = 0, no 1, yes
		Flag to compute and file trajectory after convergence IFILEX = 0, no 1, yes

B. Computation

Program calls subroutine LAMB (LAMB3) to solve the boundary value problem.

C. Output Parameters

Outputs are given under subroutine LAMB (LAMB3). If either IPTJX = 1 or IFILEX = 1, the trajectory after convergence will be printed or filed or both.

PROGRAM ETP21 (ETP213)

A. Input Parameters

<u>Parameter</u>	<u>Symbol</u>	<u>Definition</u>
ITAR		Flag to input or compute target ITAR = 0, compute 1, input
REVMAG	$ \bar{R}_{EV}(t_0) $	
OINCD	$i$ (deg)	
OBLD	OBL (deg)	
TDAY0	$t_0$ (deg)	
TTRIPD		$t_1 - t_0$ (day)
VEVMAG	$ \bar{V}_{EV}(t_0) $	
LOND	$\Omega$ (deg)	
THED	$\theta$ (deg)	
KDX		Initial value of constraint restoration parameter (start with KDX = 1)
EPSV		V-matrix scaling
EPSTSI		Allowable constraint violation ( $10^{-10}$ )
ITERMX		Max. No. of iterations
ITAR = 0:		{ Parameter to define target on Halo orbit. $A_y = 2 \times 10^5$ km $A_z = 1 \times 10^5$ km $A_{TAR}$ = Target position measured from $X_L$ or $X_4$ axis.
AYM	$A_y$ (m)	
AZM	$A_z$ (m)	
ATARD	$A_{TAR}$ (deg)	
4-Body:		
RSE0	$\bar{R}_{SE}(t_0)$	
VSE0	$\bar{V}_{SE}(t_0)$	
RSM0	$\bar{R}_{SM}(t_0)$	
VSM0	$\bar{V}_{SM}(t_0)$	



ITAR = 1:

RSVTAR  $\bar{R}_{SVTAR}(t_f)$

VSVTAR  $\bar{V}_{SVTAR}(t_f)$

3-Body:

REMO  $\bar{R}_{EM}(t_o)$

VEMO  $\bar{V}_{EM}(t_o)$

ITAR = 1:

REVTAR  $\bar{R}_{EVTAR}(t_f)$

VEVTAR  $\bar{V}_{EVTAR}(t_f)$

B. Computation

See Ref. 2 appendix, program listing and flow chart.

C. Internal Parameters

ALPHA, BETA

Davidon algorithm parameters

ALPHAM, BETAM

Davidon algorithm parameters

<u>Current Variables</u>	<u>Trial Variables</u>	<u>Dummy Variables</u>	
X	XS	XD	Independent variables
F	FS	FD	Cost
G	GS	GD	Cost gradient
FG	FGS	FGD	Augmented cost
TSI	TSIS	TSID	Constraint violation
LT	LTS	LTD	Constraint gradient
GG	GGs	GGD	Augmented cost gradient
TESTR	TESTRS	TESTRD	Magnitude of constraint violation
KDX			Constraint restoration scaling (internally adjusted)

KV

V matrix internal scaling  
(internally determined)

R

Parameters to compute quantities  
to update V matrix

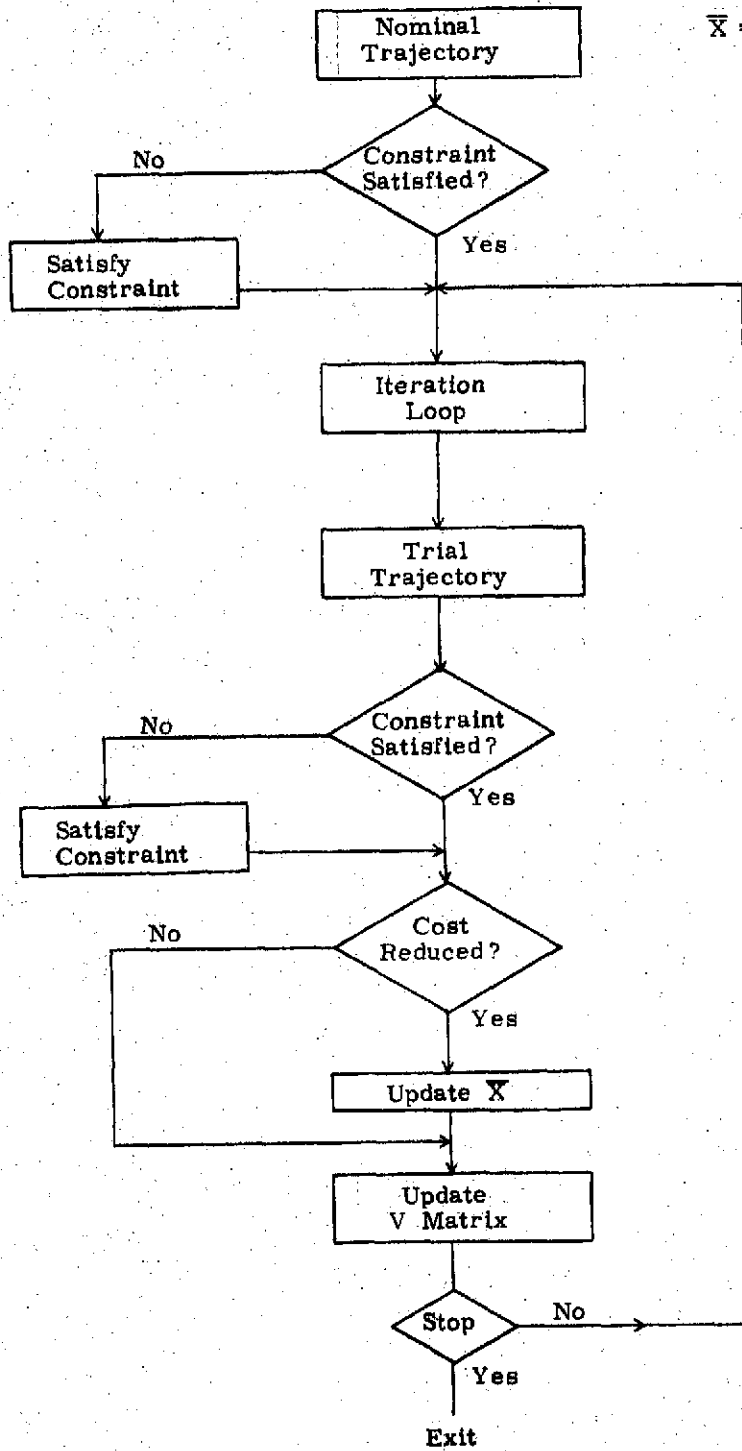
P

Estimate of twice the excess of cost  
above its minimum value. If  
 $P < EPS$ , the problem is considered  
to be converged.

Program ETP21 and ETP213

$\bar{X}$  = Independent Variables

$$\bar{X} = (\bar{V}_{EV}(t_0), \Omega, \theta)$$



PROGRAM PTP31 (PTP313)

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
4-Body:		
RSV0	$\bar{R}_{SV}(t_0)$	
VSV0	$\bar{V}_{SV}^-(t_0)$	Velocity before initial impulse
VSVI	$\bar{V}_{SV}^+(t_0)$	Velocity after initial impulse
RSE0	$\bar{R}_{SE}(t_0)$	
VSE0	$\bar{V}_{SE}(t_0)$	
RSM0	$\bar{R}_{SM}(t_0)$	
VSM0	$\bar{V}_{SM}(t_0)$	
VSVMP	$\bar{V}_{SM}^+(t_m)$	Velocity after interior impulse
TM	$t_m$	Time of interior impulse
RSVTAR	$\bar{R}_{SVTAR}(t_f)$	Target position
VSVTAR	$\bar{V}_{SVTAR}(t_f)$	Target velocity
3-Body:		
REVO	$\bar{R}_{EV}(t_0)$	
VEVO	$\bar{V}_{EV}^-(t_0)$	Velocity before impulse
VEVOP	$\bar{V}_{EV}^+(t_0)$	Velocity after impulse
REMO	$\bar{R}_{EM}(t_0)$	
VEMO	$\bar{V}_{EM}(t_0)$	
VEVMP	$\bar{V}_{EV}^+(t_m)$	Velocity after interior impulse
TM	$t_m$	Time of interior impulse

Common to 4-Body and 3-Body:

ITLMAX	Maximum number of Lambert iterations
ILINC	Number of increments to solve Lambert problem
ITDMAX	Maximum number of Davidon iterations
FMINM	Estimated minimum value of $\Delta V$ (m/s)
KNR	Lambert iteration parameter (use KNR = 1 to start)
ERRMIN	Allowable constraint violation ( $10^{-10}$ )

B. Computation

See Ref. 2, program listing and flow chart.

C. Internal Parameters

SG	Projection of cost gradient on search direction.
ALPHA	Step size in search direction
SGS	Projection of trial point cost gradient on search direction. Its sign is used to determine whether as interpolation or a reduction of interval is to be made.
R, RC	Parameters to update V matrix. R used in trial step. RC used in interpolation.
ALPHAC	Step size in search direction determined by cubic interpolation.
P, PC	Parameters to update V matrix. P used in trial step. PC used in interpolation

EPS	If magnitude of cost gradient is less than EPS, the problem is considered to be converged.
F	Cost
GMAG	Magnitude of cost gradient.
GSMAG	Magnitude of cost gradient of a trial point.
GCMAG	Magnitude of cost gradient of an interpolated point.
RMAG	Magnitude of R.
RCMAG	Magnitude of RC.
FS	Cost of a trial point.
FC	Cost of an interpolated point.
DG	Difference in cost gradient.
DX	Change in independent variable to compute a trial point.
DXC	Change in independent variable to compute an interpolated point.
PVMMAG, VVPMAG	See subroutine COMG

D. Stopping Options

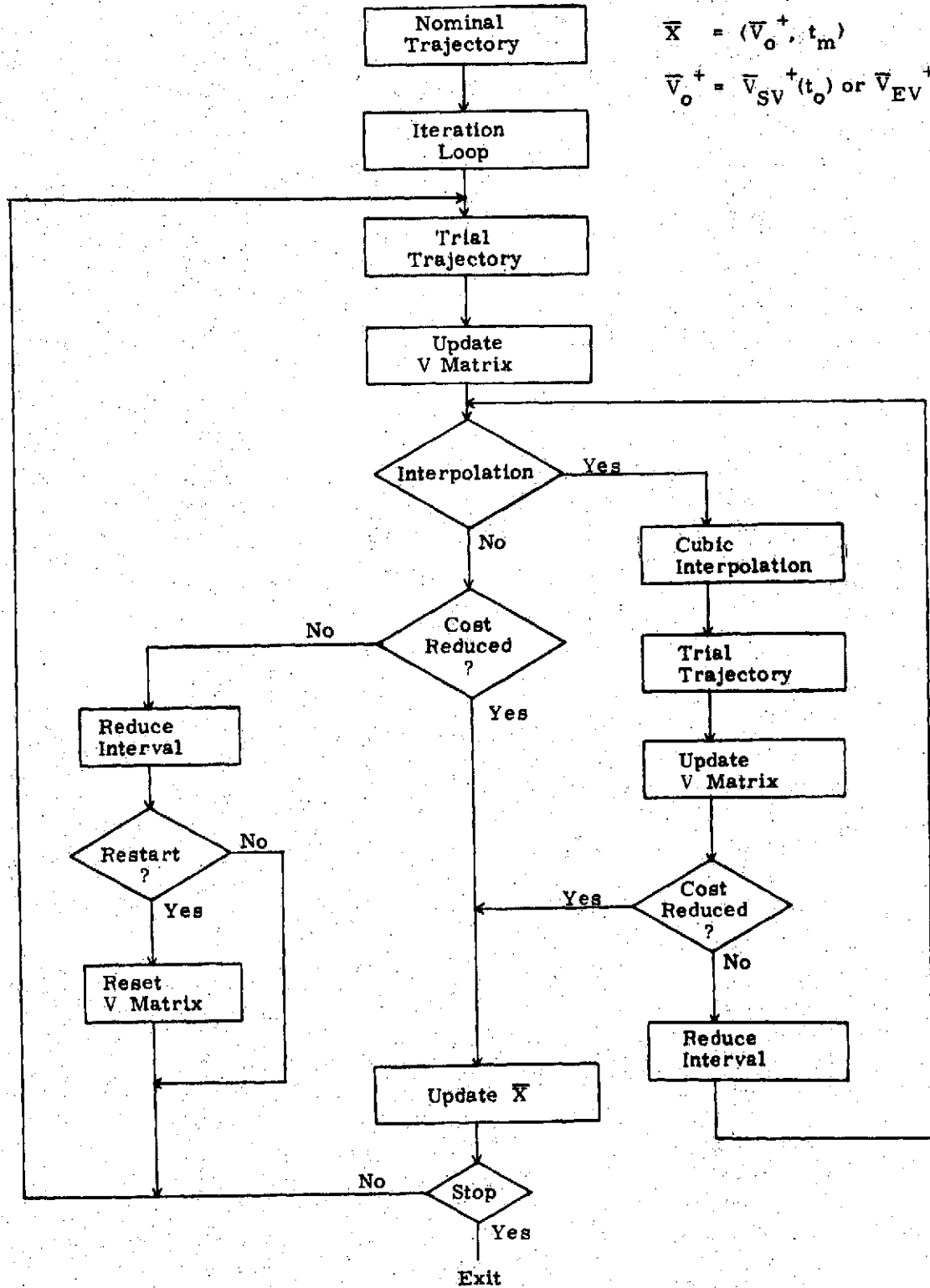
The programmed stopping condition is when GMAG is reduced to less than EPS. However, one should stop the iteration if F is not decreasing significantly from one iteration to the next. On the other hand, it may be necessary to continue iteration to satisfy the condition that PVMMAG is nearly equal to VVPMAG. (See comment in Ex. 4.)

Program PTP3I and PTP3I3

$\bar{X}$  = Independent Variables

$\bar{X} = (\bar{V}_0^+, t_m)$

$\bar{V}_0^+ = \bar{V}_{SV}^+(t_0) \text{ or } \bar{V}_{EV}^+(t_0)$



## INPUT DATA DECKS

Program TRAJ

<u>Card No.</u>	<u>Format</u>	<u>Parameters</u>
1	4D20.11	GL1, AUM, UTIME, UVELM
2	3D20.11	MS, ME, MM
3	4I5	IMODE, IPV, IPTRAJ, IFILE
4	4D20.11	RSE0(1), RSE0(2), RSE0(3), VSE0(1)
5	4D20.11	VSE0(2), VSE0(3), RSM0(1), RSM0(2)
6	4D20.11	RSM0(3), VSM0(1), VSM0(2), VSM0(3)
7	3D20.11	TDAY0, TRIPD, ERRMXM

The above 7 input cards are needed for all modes. The following 2 cards depend on the mode selected

IMODE = 1

8	4D20.11	REVMAG, VEV MAG, OINCD, OBLD
9	2D20.11	LOND, THED

IMODE = 2

8	4D20.11	REV0(1), REV0(2), REV0(3), VEV0P(1)
9	2D20.11	VEV0P(2), VEV0P(3)

IMODE = 3

8	4D20.11	RSV0(1), RSV0(2), RSV0(3), VSVI(1)
9	2D20.11	VSVI(2), VSVI(3)

The following 2 input cards are needed if IPV = 1

10	4D20.11	PV0(1), PV0(2), PV0(3), PV0(4)
11	2D20.11	PV0(5), PV0(6)



Program TRAJ3

<u>Card No.</u>	<u>Format</u>	<u>Parameters</u>
1	4D20.11	GAMMA, UDM, UTIME, UVELM
2	2D20.11	ME, MM
3	4D20.11	IMODE, IPV, IPTRAJ, IFILE
4	4D20.11	REM0(1), REM0(2), REM0(3), VEM0(1)
5	2D20.11	VEM0(2), VEM0(3)
6	3D20.11	TDAY0, TRIPD, ERRMXM

The above 6 input cards are needed for all modes. The following 2 cards depend on the mode selected.

		IMODE = 1
7	4D20.11	REVMAG, VEV MAG, OINCD, LOND
8	1D20.11	THED
		IMODE = 2
7	4D20.11	REV0(1), REV0(2), REV0(3), VEV0P(1)
8	2D20.11	VEV0P(2), VEV0P(3)

The following 2 cards are needed if IPV = 1

9	4D20.11	PV0(1), PV0(2), PV0(3), PV0(4)
10	2D20.11	PV0(5), PV0(6)

Program EXLAM

<u>Card No.</u>	<u>Format</u>	<u>Parameters</u>
1	4D20.11	GLI, AUM, UTIME, UVELM
2	3D20.11	MS, ME, MM
3	4I5	IMODE, ITLMAX, IPTJX, IFILEX
4	4D20.11	RSE0(1), RSE0(2), RSE0(3), VSE0(1)
5	4D20.11	VSE0(2), VSE0(3), RSM0(1), RSM0(2)
6	4D20.11	RSM0(3), VSM0(1), VSM0(2), VSM0(3)
7	3D20.11	TDAY0, TRIPD, ERRMXM
8	4D20.11	RSVTAR(1), RSVTAR(2), RSVTAR(3), ERRMIN
9	1D20.11	KNR

The above 9 cards are needed for all modes. The following cards depend on the mode selected.

IMODE = 1		
10	4D20.11	REVMAG, VEV MAG, OINCD, OBLD
11	2D20.11	LOND, THED
IMODE = 2		
10	4D20.11	REV0(1), REV0(2), REV0(3), VEV0P(1)
11	2D20.11	VEV0P(2), VEV0P(3)
IMODE = 3		
10	4D20.11	RSV0(1), RSV0(2), RSV0(3), VSVI(1)
11	2D20.11	VSVI(2), VSVI(3)

Program EXLAM3

<u>Card No.</u>	<u>Format</u>	<u>Parameters</u>
1	4D20.11	GAMMA, UDM, UTIME, UVELM
2	2D20.11	ME, MM
3	4I5	IMODE, ITLMAX, IPTJX, IFILEX
4	4D20.11	REM0(1), REM0(2), REM0(3), VEM0(1)
5	2D20.11	VEM0(2), VEM0(3)
6	3D20.11	TDAY0, TRIPD, ERRMXM
7	4D20.11	REVTAR(1), REVTAR(2), REVTAR(3), ERRMIN
8	1D20.11	KNR

The above 8 cards are needed for all modes. The following cards depend on the mode selected.

IMODE = 1

9	4D20.11	REVMAG, VEV MAG, OINCD, LOND
10	1D20.11	THED

IMODE = 2

9	4D20.11	REV0(1), REV0(2), REV0(3), VEV0P(1)
10	2D20.11	VEV0P(2), VEV0P(3)

Program ETP21

<u>Card No.</u>	<u>Format</u>	<u>Parameters</u>
1	4D20.11	GLI, AUM, UTIME, UVELM
2	3D20.11	MS, ME, MM
3	4D20.11	REVMAG, OINCD, OBLD, ERRMXM
4	2D20.11	TDAY0, TTRIPD
5	3D20.11	VEVMAG, LOND, THED
6	4D20.11	RSE0(1), RSE0(2), RSE0(3), VSE0(1)
7	4D20.11	VSE0(2), VSE0(3), RSM0(1), RSM0(2)
8	4D20.11	RSM0(3), VSM0(1), VSM0(2), VSM0(3)
9	4D20.11	EPS, EPSTSI, KDX, EPSV
10	4I5	ICOMV, ITERMX, IFILEX, ITAR

The next 1 or 2 cards depend on ITAR.

		ITAR = 0
11	3D20.11	AYM, AYZ, ATARD
		ITAR = 1
11	4D20.11	RSVTAR(1), RSVTAR(2), RSVTAR(3), VSVTAR(1)
12	2D20.11	VSVTAR(2), VSVTAR(3)

The following 3 cards are needed only if ICOMV = 0.

13	4D20.11	V(1, 1), V(1, 2), V(1, 3), V(2, 1)
14	4D20.11	V(2, 2), V(2, 3), V(3, 1), V(3, 2)
15	4D20.11	V(3, 3)

Program ETP213

<u>Card No.</u>	<u>Format</u>	<u>Parameters</u>
1	4D20.11	GAMMA, UDM, UTIME, UVELM
2	2D20.11	ME, MM
3	3D20.11	REVMAG, OINCD, ERRMXM
4	2D20.11	TDAY0, TTRIPD
5	3D20.11	VEVMAG, LOND, THED
6	4D20.11	REM0(1), REM0(2), REM0(3), VEM0(1)
7	2D20.11	VEM0(2), VEM0(3)
8	4D20.11	EPS, EPSTSI, KDX, EPSV
9	415	ICOMV, ITERMV, IFILEX, ITAR

The next 1 or 2 cards depend on ITAR.

		ITAR = 0
10	3D20.11	AYM, AZM, ATARD
		ITAR = 1
11	4D20.11	REVTAR(1), REVTAR(2), REVTAR(3), VEVTAR(1)
12	2D20.11	VEVTAR(2), VEVTAR(3)

The following 3 cards are needed only if ICOMV = 0.

13	4D20.11	V(1, 1), V(1, 2), V(1, 3), V(2, 1)
14	4D20.11	V(2, 2), V(2, 3), V(3, 1), V(3, 2)
15	4D20.11	V(3, 3)

Program PTP3I

<u>Card No.</u>	<u>Format</u>	<u>Parameters</u>
1	4D20.11	GLI, AUM, UTIME, UVELM
2	3D20.11	MS, ME, MM
3	3D20.11	TSTART, TM, TEND
4	4D20.11	RSE0(1), RSE0(2), RSE0(3), VSE0(1)
5	4D20.11	VSE0(2), VSE0(3), RSM0(1), RSM0(2)
6	4D20.11	RSM0(3), VSM0(1), VSM0(2), VSM0(3)
7	4D20.11	RSV0(1), RSV0(2), RSV0(3), VSV0(1)
8	2D20.11	VSV0(2), VSV0(3)
9	4D20.11	VSVI(1), VSVI(2), VSVI(3), VSVMP(1)
10	2D20.11	VSVMP(2), VSVMP(3)
11	4D20.11	RSVTAR(1), RSVTAR(2), RSVTAR(3), VSVTAR(1)
12	2D20.11	VSVTAR(2), VSVTAR(3)
13	3D20.11	KNR, ERRMIN, ERMXM
14	3D20.11	FMINM, EPS, EPSV
15	5I5	ICOMV, ITLMAX, ILINC, ITDMAX, IFILEX

The following 4 cards are needed only if ICOMV = 0.

16	4D20.11	V(1, 1), V(1, 2), V(1, 3), V(1, 4)
17	4D20.11	V(2, 1), V(2, 2), V(2, 3), V(2, 4)
18	4D20.11	V(3, 1), V(3, 2), V(3, 3), V(3, 4)
19	4D20.11	V(4, 1), V(4, 2), V(4, 3), V(4, 4)

Program PTP313

<u>Card No.</u>	<u>Format</u>	<u>Parameters</u>
1	4D20.11	GAMMA, UDM, UTIME, UVELM
2	2D20.11	ME, MM
3	3D20.11	TSTART, TM, TEND
4	4D20.11	REM0(1), REM0(2), REM0(3), VEM0(1)
5	2D20.11	VEM0(2), VEM0(3)
6	4D20.11	REV0(1), REV0(2), REV0(3), VEV0(1)
7	2D20.11	VEV0(2), VEV0(3)
8	4D20.11	VEV0P(1), VEV0P(2), VEV0P(3), VEVMP(1)
9	2D20.11	VEVMP(2), VEVMP(3)
10	4D20.11	REVTAR(1), REVTAR(2), REVTAR(3), VEVTAR(1)
11	2D20.11	VEVTAR(2), VEVTAR(3)
12	3D20.11	KNR, ERRMIN, ERRMXM
13	3D20.11	FMINM, EPS, EPSV
14	515	ICOMV, ITLMAX, ILINC, ITDMAX, IFILEX

The following 4 cards are needed only if ICOMV = 0.

15	4D20.11	V(1, 1), V(1, 2), V(1, 3), V(1, 4)
16	4D20.11	V(2, 1), V(2, 2), V(2, 3), V(2, 4)
17	4D20.11	V(3, 1), V(3, 2), V(3, 3), V(3, 4)
18	4D20.11	V(4, 1), V(4, 2), V(4, 3), V(4, 4)

## PROGRAM PRINTOUT DESCRIPTION

ALPHA	Step size in search direction or Davidon algorithm parameter
ALPHAC	Step size in cubic interpolation
ANGE	Subtended angle at earth (deg)
ANGM	Subtended angle at moon (deg)
ANGS	Subtended angle at sun (deg)
ANGV	Subtended angle at spacecraft (deg)
ATARD	Parameter to define Halo orbit (deg)
AUM	1 Unit of distance in meters (4-body)
AYM	Parameter to define Halo orbit (meter)
AZM	Parameter to define Halo orbit (meter)
BETA	Davidon algorithm parameter
DG	Cost gradient difference
DELX	Change in independent variables
DGC	Cost gradient difference
DRM	Change in interior impulse position
DUVI	Primer vector derivative at initial time
DUVMM	Primer vector derivative before interior impulse
DUVMP	Primer vector derivative after interior impulse
DV	Cost
DVI	Impulse at initial time
DVIMAG	Magnitude of initial impulse
DVM	Interior impulse
DVMMAG	Magnitude of interior impulse



DVMPX	Change in interior impulse
DVF	Impulse at terminal time
DVFMAG	Magnitude of terminal impulse
DVFMPS	Cost of terminal impulse (MPS)
DVIMPS	Cost of initial impulse (MPS)
DVMPS	Total cost (MPS)
DVMMP	Cost of interior impulse (MPS)
DX	Change in independent variables
DXC	Change in independent variables
EPS	Parameter to terminate Davidon iterations
EPSTSI	Allowable terminal constraint violation
EPSV	Factor to scale down variance matrix
ERR	Constraint violation in Lambert iteration
ERRMAX	Allowable single step position error
ERRMIN	Allowable Lambert constraint violation
ERRMXM	Allowable single step position error (meter)
F	Cost
FC	Cost in cubic interpolation
FD	Cost
FG	Augmented cost
FGS	Augmented cost
FMINM	Estimated cost (MPS)
FS	Cost
G	Cost gradient

GAMMA	Parameter to define libration point (3-body)
GC	Cost gradient in cubic interpolation
GCMAG	Magnitude of cost gradient
GD	Cost gradient
GG	Augmented cost gradient
GGS	Augmented cost gradient
GL1	Parameter to define L1 libration point (4-body)
GMAG	Magnitude of cost gradient
GS	Cost gradient
GSMAG	Magnitude of cost gradient
H	Computed step size in state extrapolation
HDAY	Step size in days
ICOMV	Flag to compute variance matrix
IFILE	Flag to file trajectory data
IFILEX	Flag to file trajectory data
ILINC	Number of Lambert increments
IMODE	Flag to select mode of initial condition
IMTX	Flag to compute state transition matrix
IPTJX	Flag to print trajectory
IPTRAJ	Flag to print trajectory
IPV	Flag to compute primer vector
IPVTM	Flag to monitor primer vector magnitude history
ISTEP	Number of integration steps
ITAR	Flag to compute target position and velocity on Halo orbit

ITDMAX	Maximum number of Davidon iterations
ITER	Number of iteration in Lambert routine
ITERD	Number of Davidon iterations
ITERL	Number of Lambert increment
ITERMX	Maximum of Davidon iterations
ITLMAX	Maximum number of Lambert iterations
KDX	Parameter to scale independent variables in constraint restoration
KNR	Parameter to scale independent variables in Lambert iteration.
KV	Parameter to scale variance matrix
LDM	Derivative of primer vector magnitude
LDTM	Derivative of maximum primer vector magnitude
LOND	Longitude of ascending node (deg)
LM	Magnitude of primer vector
LT	Constraint gradient matrix
LTM	Maximum primer vector magnitude
ME	Gravitational constant of earth
MM	Gravitational constant of moon
MS	Gravitational constant of sun
OBLD	Obliquity angle (deg)
OINCD	Orbital inclination (deg)
P	Davidon algorithm parameter (iteration stops if P is less than EPS in ETP21 or ETP213)
PC	Davidon algorithm parameter
PV	Primer vector and derivative
PVO	Primer vector and derivative at initial time

PVMMAG	Projection of primer vector derivative on primer vector before interior impulse
PVPMAG	Projection of primer vector derivative on primer vector after interior impulse
R	Davidon algorithm parameters
RC	Davidon algorithm parameters in cubic interpolation
RCMAG	Magnitude of RC
RDED	Position of earth in rotating coordinates
RDLID	Position of L1 libration point in rotating coordinates
RDMD	Position of moon in rotating coordinates
RDSO	Position of sun in rotating coordinates
RDVD	Position of spacecraft in rotating coordinates
REL	Position of libration point wrt earth
REM	Position of moon wrt earth
REMO	Position of moon wrt earth at initial time
REMF	Position of moon wrt earth at terminal time
REV	Position of spacecraft wrt earth
REVF	Position of spacecraft wrt earth at terminal time
REVM	Position of spacecraft wrt earth at interior impulse time
REVMAG	Distance of spacecraft to earth
REVMX	Change of position of spacecraft wrt earth at interior impulse time
REVO	Position of spacecraft wrt earth at initial time
REVMO	Position of spacecraft wrt earth at interior impulse time
REVTAR	Target position wrt earth

RLTARL	Target position wrt libration point
RMAG	Magnitude of R
RMV	Position of spacecraft wrt moon
RMVF	Position of spacecraft wrt moon at terminal time
RMVMAG	Distance of spacecraft to moon
RSE	Position of earth wrt sun
RSEO	Position of earth wrt sun at initial time
RSEF	Position of earth wrt sun at terminal time
RSL1	Position of L1 libration point wrt sun
RSM	Position of moon wrt sun
RSMO	Position of moon wrt sun at initial time
RSMF	Position of moon wrt sun at terminal time
RSV	Position of spacecraft wrt sun
RSVO	Position of spacecraft wrt sun at initial time
RSVF	Position of spacecraft wrt sun at terminal time
RSVM	Position of spacecraft wrt sun at interior impulse time
RSVMAG	Distance of spacecraft to sun
RSVMD	Position of moon wrt sun at interior impulse time
RSVMX	Spacecraft position wrt sun at interior impulse
RSVTAR	Target position wrt sun
RTM	Position of spacecraft at maximum primer vector magnitude
S	Davidon algorithm search direction
SG	Projection of cost gradient on search direction
SGS	Projection of cost gradient on search direction

T	Time
TO	Initial time
TDAY	Time in day
TDAYO	Initial time in day
TDAYF	Terminal time in day
TEND	Terminal time
TESTR	Magnitude of constraint violation
TESTRD	Magnitude of constraint violation
TESTRS	Magnitude of constraint violation
TF	Terminal time
THED	Position of spacecraft from line of node (deg)
TM	Interior impulse time
TRIPD	Trip time in day
TSI	Constraint violation
TSID	Constraint violation
TSIS	Constraint violation
TSTART	Initial time
TTM	Time of maximum primer vector magnitude
TTRIP	Trip time
TTRIPD	Trip time in day
UDM	1 Unit of distance in meters (3-body)
UTIME	1 Unit of time in days
UVELM	1 MPS in dimensionless velocity
UVF	Unit vector of terminal impulse

UVI	Unit vector of initial impulse
UVM	Unit vector of interior impulse
V	Variance matrix
VDED	Velocity of earth in rotating coordinates
VDLID	Velocity of L1 libration point in rotating coordinates
VDMD	Velocity of moon in rotating coordinates
VDSO	Velocity of sun in rotating coordinates
VDVD	Velocity of spacecraft in rotating coordinates
VEL	Velocity of libration point wrt earth
VEM	Velocity of moon wrt earth
VEMO	Velocity of moon wrt earth at initial time
VEMF	Velocity of moon wrt earth at terminal time
VEV	Velocity of spacecraft wrt earth
VEVO	Velocity of spacecraft wrt earth at initial time before impulse
VEVOP	Velocity of spacecraft wrt earth after initial impulse
VEVMP	Velocity of spacecraft wrt earth after interior impulse
VEVMPX	Spacecraft velocity wrt earth after interior impulse
VEVF	Velocity of spacecraft wrt earth at terminal time
VEVMAG	Magnitude of spacecraft velocity wrt earth
VEVMPD	Velocity of spacecraft wrt earth after interior impulse
VEVMPS	Magnitude of spacecraft velocity wrt earth in MPS
VEVTAR	Target velocity wrt earth
VIT	Initial velocity in Lambert iteration
VL TARL	Velocity of target wrt libration point

VMV	Velocity of spacecraft wrt moon
VMVF	Velocity of spacecraft wrt moon at terminal time
VSE	Velocity of earth wrt sun
VSEO	Velocity of earth wrt sun at initial time
VSEF	Velocity of earth wrt sun at terminal time
VSLI	Velocity of L1 libration point wrt sun
VSM	Velocity of moon wrt sun
VSMO	Velocity of moon wrt sun at initial time
VSMF	Velocity of moon wrt sun at terminal time
VSV	Velocity of spacecraft wrt sun
VSVI	Velocity of spacecraft wrt sun after initial impulse
VSVMPX	Spacecraft velocity wrt sun after interior impulse
VSVF	Velocity of spacecraft wrt sun at terminal time
VSVMP	Velocity of spacecraft wrt sun after interior impulse
VSVMPD	Velocity of spacecraft wrt sun after interior impulse
VSVO	Velocity of spacecraft wrt sun at initial time before impulse
VSVTAR	Target velocity wrt sun
VTM	Velocity of spacecraft at maximum primer vector magnitude
X	Independent variables in Davidon iterations
XC	Independent variables in cubic interpolation
XD	Independent variables
XS	Independent variables



Subroutine FOURBY

A. Input Parameters

<u>Parameter</u>	<u>Symbol</u>	<u>Definition</u>
XT0	$t_0$	
XTF	$t_f$	
RSV0	$\bar{R}_{SV}(t_0)$	
VSV0	$\bar{V}_{SV}(t_0)$	
RSE0	$\bar{R}_{SE}(t_0)$	
VSE0	$\bar{V}_{SE}(t_0)$	
RSM0	$\bar{R}_{SM}(t_0)$	
VSM0	$\bar{V}_{SM}(t_0)$	
PV0	$\bar{\lambda}(t_0), \dot{\bar{\lambda}}(t_0)$	Primer vector and derivative

B. Output Parameters

RSVD	$\bar{R}_{SV}(t_f)$	
VSVD	$\bar{V}_{SV}(t_f)$	
RSED	$\bar{R}_{SE}(t_f)$	
VSED	$\bar{V}_{SE}(t_f)$	
RSMD	$\bar{R}_{SM}(t_f)$	
VSMD	$\bar{V}_{SM}(t_f)$	
S11, S12, S21, S22	$\varphi(t_f, t_0)$	State transition matrix

C. Computation

- Step 1. Initialize running variables equal to input states.
- Step 2. Call RVEMV to compute  $\bar{R}_{EV}, \bar{V}_{EV}, \bar{R}_{MV}, \bar{V}_{MV}, \bar{R}_{EM}, \bar{V}_{EM}$ .
- Step 3. Call CSTEP to compute step size  $h$ ,  $\bar{\epsilon}_{RSV}, \bar{\epsilon}_{RSE}, \bar{\epsilon}_{RSM}$  and  $J$ .

Step 4. Call DISP

Step 5. If IPV = 1, call PVEC.  
If IFILE = 1, call FDATA.  
If IPTRAJ = 1, call PTRAJ.

Step 6. Call TWOBODY to compute 6 conics:

$$[\bar{R}_{SV}, \bar{V}_{SV}]$$

$$[\bar{R}_{SE}, \bar{V}_{SE}]$$

$$[\bar{R}_{SM}, \bar{V}_{SM}]$$

$$[\bar{R}_{EV}, \bar{V}_{EV}]$$

$$[\bar{R}_{MV}, \bar{V}_{MV}]$$

$$[\bar{R}_{EM}, \bar{V}_{EM}]$$

Step 7. Compute 3 perturbation vectors. Let  
 $[\bar{R}_{i, j}] = \text{CONIC of } j \text{ with respect to } i$

$\bar{R}_{i, j} = \text{reference trajectory of } j \text{ with respect to } i.$

$$d\bar{R}_{SV} = [\bar{R}_{SV}] - \bar{R}_{SV}^{(0)} - h \bar{V}_{SV}^{(0)}$$

$$d\bar{V}_{SV} = [\bar{V}_{SV}] - \bar{V}_{SV}^{(0)}$$

$$d\bar{R}_{SE} = [\bar{R}_{SE}] - \bar{R}_{SE}^{(0)} - h \bar{V}_{SE}^{(0)}$$

$$d\bar{V}_{SE} = [\bar{V}_{SE}] - \bar{V}_{SE}^{(0)}$$

$$d\bar{R}_{SM} = [\bar{R}_{SM}] - \bar{R}_{SM}^{(0)} - h \bar{V}_{SM}^{(0)}$$

$$d\bar{V}_{SM} = [\bar{V}_{SM}] - \bar{V}_{SM}^{(0)}$$

$$d\bar{R}_{EV} = [\bar{R}_{EV}] - \bar{R}_{EV(0)} - h \bar{V}_{EV(0)}$$

$$d\bar{V}_{EV} = [\bar{V}_{EV}] - \bar{V}_{EV(0)}$$

$$d\bar{R}_{MV} = [\bar{R}_{MV}] - \bar{R}_{MV(0)} - h \bar{V}_{MV(0)}$$

$$d\bar{V}_{MV} = [\bar{V}_{MV}] - \bar{V}_{MV(0)}$$

$$d\bar{R}_{EM} = [\bar{R}_{EM}] - \bar{R}_{EM(0)} - h \bar{V}_{EM(0)}$$

$$d\bar{V}_{EM} = [\bar{V}_{EM}] - \bar{V}_{EM(0)}$$

Approximate perturbations

$$\bar{P}_{RSV} = \frac{\mu_E}{\mu_S + \mu_E} d\bar{R}_{SE} + d\bar{R}_{EV} + \frac{\mu_M}{\mu_S + \mu_M} d\bar{R}_{SM} + d\bar{R}_{MV}$$

$$\bar{P}_{VSV} = \frac{\mu_E}{\mu_S + \mu_E} d\bar{V}_{SE} + d\bar{V}_{EV} + \frac{\mu_M}{\mu_S + \mu_M} d\bar{V}_{SM} + d\bar{V}_{MV}$$

$$\bar{P}_{RSE} = \frac{\mu_M}{\mu_S + \mu_M} d\bar{R}_{SM} - \frac{\mu_M}{\mu_E + \mu_M} d\bar{R}_{EM}$$

$$\bar{P}_{VSE} = \frac{\mu_M}{\mu_S + \mu_M} d\bar{V}_{SM} - \frac{\mu_M}{\mu_E + \mu_M} d\bar{V}_{EM}$$

$$\bar{P}_{RSM} = \frac{\mu_E}{\mu_S + \mu_E} d\bar{R}_{SE} + \frac{\mu_E}{\mu_E + \mu_M} d\bar{R}_{EM}$$

$$\bar{P}_{VSM} = \frac{\mu_E}{\mu_S + \mu_E} d\bar{V}_{SE} + \frac{\mu_E}{\mu_E + \mu_M} d\bar{V}_{EM}$$

Step 8. Compute reference trajectories

$$\bar{R}_{SV} = [\bar{R}_{SV}] + \bar{P}_{RSV}$$

$$\bar{V}_{SV} = [\bar{V}_{SV}] + \bar{P}_{VSV}$$

$$\bar{R}_{SE} = [\bar{R}_{SE}] + \bar{P}_{RSE}$$

$$\bar{V}_{SE} = [\bar{V}_{SE}] + \bar{P}_{VSE}$$

$$\bar{R}_{SM} = [\bar{R}_{SM}] + \bar{P}_{RSM}$$

$$\bar{V}_{SM} = [\bar{V}_{SM}] + \bar{P}_{VSM}$$

$$\bar{R}_{EV} = -\bar{R}_{SE} + \bar{R}_{SV}$$

$$\bar{V}_{EV} = -\bar{V}_{SE} + \bar{V}_{SV}$$

$$\bar{R}_{MV} = -\bar{R}_{SM} + \bar{R}_{SV}$$

$$\bar{V}_{MV} = -\bar{V}_{SM} + \bar{V}_{SV}$$

$$\bar{R}_{EM} = -\bar{R}_{SE} + \bar{R}_{SM}$$

$$\bar{V}_{EM} = -\bar{V}_{SE} + \bar{V}_{SM}$$

$$t = t + h$$

- Step 9. Call DELRV to compute corrections.
- Step 10. Correct state vectors.
- Step 11. If IMTX = 1, update state transition matrix.
- Step 12. Call DISP
- Step 13. If IPV = 1, call PVEC.  
If IFILE = 1, call FDATA.  
If IPTRAJ = 1, call PTRAJ
- Step 14. If  $t \geq t_p$ , exit.
- Step 15. Call CSTEP.  
Go to Step 6.

Subroutine THRBODY

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
XTO	$t_o$	
XTF	$t_f$	
REVO	$\bar{R}_{EV}(t_o)$	
VEVO	$\bar{V}_{EV}(t_o)$	
REMO	$\bar{R}_{EM}(t_o)$	
VEMO	$\bar{V}_{EM}(t_o)$	
PVO	$\bar{\lambda}, \dot{\lambda}$	

B. Output Parameters

REVD	$\bar{R}_{EV}(t_f)$	
VEVD	$\bar{V}_{EV}(t_f)$	
REMD	$\bar{R}_{EM}(t_f)$	
VEMD	$\bar{V}_{EM}(t_f)$	
S11, S12, S21, S22	$\phi(t_f, t_o)$	State transition matrix

C. Computation

- Step 1. Initialize running variables equal to input states.
- Step 2. Call CSTEP3 to compute step size  $h$ ,  $\epsilon_{REV}$  and  $J$ .
- Step 3. Call DISP3
- Step 4. If IPV = 1, call PVEC  
If IFILE = 1, call FDATA3  
If IPTRAJ = 1, call PTRAJ3.
- Step 5. Call TWOBODY to compute 3 conics:

$$[\bar{R}_{EV}, \bar{V}_{EV}]$$

$$[\bar{R}_{EM}, \bar{V}_{EM}]$$

$$[\bar{R}_{MV}, \bar{V}_{MV}]$$

Step 6. Compute perturbation vector

$$\bar{P}_{REV} = [\bar{R}_{MV}] - \bar{R}_{MV(0)} - h \bar{V}_{MV(0)}$$

$$\bar{P}_{VEV} = [\bar{V}_{MV}] - \bar{V}_{MV(0)}$$

Step 7. Compute reference trajectories

$$\bar{R}_{EV} = [\bar{R}_{EV}] + \bar{P}_{REV}$$

$$\bar{V}_{EV} = [\bar{V}_{EV}] + \bar{P}_{VEV}$$

$$\bar{R}_{EM} = [\bar{R}_{EM}]$$

$$\bar{V}_{EM} = [\bar{V}_{EM}]$$

$$\bar{R}_{MV} = -\bar{R}_{EM} + \bar{R}_{EV}$$

$$\bar{V}_{MV} = -\bar{V}_{EM} + \bar{V}_{EV}$$

$$t = t + h$$

Step 8. Call DELRV3 to compute corrections.

Step 9. Correct state vectors.

Step 10. If IMTX = 1, update state transition matrix.

Step 11. Call DISP3

Step 12. If IPV = 1, call PVEC.  
If IFILE = 1, call FDATA3.  
If IPTRAJ = 1, call PTRAJ3.

Step 13. If  $t \geq t_p$ , exit.

Step 14. Call CSTEP3.  
Go to Step 5.

Subroutine TWOBDY

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
XRO	$\bar{R}(t)$	
XVO	$\bar{V}(t)$	
TAU	$h$	Step size
MU	$\mu$	Gravitational constant
PSI	$\psi$	Generalized eccentric anomaly
IMTX	$I_{MTX}$	Flag to compute state transition matrix: $I_{MTX} = 0$ , no 1, yes

B. Output Parameters

XRF	$\bar{R}(t+h)$	
VRF	$\bar{V}(t+h)$	
PSI	$\psi$	
P	$\phi(t+h, t)$	State transition matrix

C. Computation

See Ref. (3).



Subroutine CSTEP

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
ERRMAX	$\epsilon_{\max}$	Allowable single step position error
REV	$\bar{R}_{EV}(t)$	
RSE	$\bar{R}_{SE}(t)$	
RSV	$\bar{R}_{SV}(t)$	
RMV	$\bar{R}_{MV}(t)$	
RSM	$\bar{R}_{SM}(t)$	
REM	$\bar{R}_{EM}(t)$	
TGO	tgo	$t_f - t$

B. Output Parameters

H	h	Step size
DR4RSV	$\left. \begin{array}{l} \epsilon_{RSV}^{(4)}(t) \\ \epsilon_{RSE}^{(4)}(t) \\ \epsilon_{RSM}^{(4)}(t) \end{array} \right\}$	4th Derivative of position error estimates
DR4RSE		
DR4R5M		
BIGJ	J	$J = \begin{pmatrix} I_3 & hI_3 \\ O_3 & I_3 \end{pmatrix}$

C. Computation

$$\begin{aligned}
\epsilon_{RSV}^{\dots}(t) = & \mu_S \mu_E \left[ \frac{1}{R_{SV}} (1 - 3 \bar{u}_{SV} \bar{u}_{SV}^T) \left( \frac{\bar{R}_{EV}}{R_{EV}} + \frac{\bar{R}_{SE}}{R_{SE}} \right) \right. \\
& + \left. \frac{1}{R_{EV}} (1 - 3 \bar{u}_{EV} \bar{u}_{EV}^T) \left( \frac{\bar{R}_{SV}}{R_{SV}} - \frac{\bar{R}_{SE}}{R_{SE}} \right) \right] \\
& + \mu_S \mu_M \left[ \frac{1}{R_{SV}} (1 - 3 \bar{u}_{SV} \bar{u}_{SV}^T) \left( \frac{\bar{R}_{MV}}{R_{MV}} + \frac{\bar{R}_{SM}}{R_{SM}} \right) \right. \\
& + \left. \frac{1}{R_{MV}} (1 - 3 \bar{u}_{MV} \bar{u}_{MV}^T) \left( \frac{\bar{R}_{SV}}{R_{SV}} - \frac{\bar{R}_{SM}}{R_{SM}} \right) \right] \\
& + \mu_E \mu_M \left[ \frac{1}{R_{EV}} (1 - 3 \bar{u}_{EV} \bar{u}_{EV}^T) \left( \frac{\bar{R}_{MV}}{R_{MV}} + \frac{\bar{R}_{EM}}{R_{EM}} \right) \right. \\
& - \left. \frac{1}{R_{SE}} (1 - 3 \bar{u}_{SE} \bar{u}_{SE}^T) \left( \frac{\bar{R}_{EM}}{R_{EM}} - \frac{\bar{R}_{SM}}{R_{SM}} \right) \right. \\
& + \left. \frac{1}{R_{MV}} (1 - 3 \bar{u}_{MV} \bar{u}_{MV}^T) \left( \frac{\bar{R}_{EV}}{R_{EV}} - \frac{\bar{R}_{EM}}{R_{EM}} \right) \right. \\
& + \left. \frac{1}{R_{SM}} (1 - 3 \bar{u}_{SM} \bar{u}_{SM}^T) \left( \frac{\bar{R}_{EM}}{R_{EM}} + \frac{\bar{R}_{SE}}{R_{SE}} \right) \right]
\end{aligned}$$

$$h = \left( 24 \frac{\epsilon_{\max}^{\dots}}{\epsilon_{RSV}} \right)^{1/4}$$

$$\bar{u}_{SV} = \text{UNIT}(\bar{R}_{SV})$$

$$\bar{u}_{EV} = \text{UNIT}(\bar{R}_{EV})$$

$$\bar{u}_{MV} = \text{UNIT}(\bar{R}_{MV})$$

$$\bar{u}_{SE} = \text{UNIT}(\bar{R}_{SE})$$

$$\bar{u}_{SM} = \text{UNIT}(\bar{R}_{SM})$$

$$\begin{aligned}
\frac{\dots}{\epsilon_{RSE}}(t) &= -\frac{(\mu_S + \mu_E)}{R_{SE}^3} \mu_M (1 - 3 \bar{u}_{SE} \bar{u}_{SE}^T) \left( \frac{\bar{R}_{EM}}{R_{EM}^3} - \frac{\bar{R}_{SM}}{R_{SM}^3} \right) \\
&\quad - \frac{\mu_M \mu_S}{R_{EM}^3} (1 - 3 \bar{u}_{EM} \bar{u}_{EM}^T) \left( \frac{\bar{R}_{SM}}{R_{SM}^3} - \frac{\bar{R}_{SE}}{R_{SE}^3} \right) \\
&\quad + \frac{\mu_M \mu_E}{R_{SM}^3} (1 - 3 \bar{u}_{SM} \bar{u}_{SM}^T) \left( \frac{\bar{R}_{EM}}{R_{EM}^3} + \frac{\bar{R}_{SE}}{R_{SE}^3} \right) \\
\frac{\dots}{\epsilon_{RSM}}(t) &= \frac{(\mu_S + \mu_M)}{R_{SM}^3} \mu_E (1 - 3 \bar{u}_{SM} \bar{u}_{SM}^T) \left( \frac{\bar{R}_{EM}}{R_{EM}^3} + \frac{\bar{R}_{SE}}{R_{SE}^3} \right) \\
&\quad + \frac{\mu_E \mu_S}{R_{EM}^3} (1 - 3 \bar{u}_{EM} \bar{u}_{EM}^T) \left( \frac{\bar{R}_{SM}}{R_{SM}^3} - \frac{\bar{R}_{SE}}{R_{SE}^3} \right) \\
&\quad - \frac{\mu_E \mu_M}{R_{SE}^3} (1 - 3 \bar{u}_{SE} \bar{u}_{SE}^T) \left( \frac{\bar{R}_{EM}}{R_{EM}^3} - \frac{\bar{R}_{SM}}{R_{SM}^3} \right)
\end{aligned}$$

All quantities are evaluated at the beginning of an integration step. If  $h$  is greater than  $t_{go}$ , it is set to  $t_{go}$  for the last step.

Subroutine CSTEP3

A. Input Parameters

<u>Parameters</u>	<u>Symbols</u>	<u>Definition</u>
ERRMAX	$\epsilon_{\max}$	Allowable single step position error
REV	$\bar{R}_{EV}(t)$	
RMV	$\bar{R}_{MV}(t)$	
TGO	tgo	Terminal time - present time

B. Output Parameters

H	h	Step size
DR4REV	$\frac{\dots}{\epsilon_{REV}}(t)$	4th Derivative of position error estimate
BIGJ	J	$J = \begin{pmatrix} I_3 & hI_3 \\ O_3 & I_3 \end{pmatrix}$

C. Computation

$$\frac{\dots}{\epsilon_{REV}}(t) = \mu_E \mu_M \left\{ \frac{1}{R_{EV}^3} (I - 3 \bar{u}_{EV} \bar{u}_{EV}^T) \left( \frac{\bar{R}_{MV}}{R_{MV}^3} + \frac{\bar{R}_{EM}}{R_{EM}^3} \right) + \frac{1}{R_{MV}^3} (I - 3 \bar{u}_{MV} \bar{u}_{MV}^T) \left( \frac{\bar{R}_{EV}}{R_{EV}^3} - \frac{\bar{R}_{EM}}{R_{EM}^3} \right) \right\}$$

$$\frac{\dots}{\epsilon_{REM}}(t) = 0$$

$$h = \left( 24 \frac{\epsilon_{\max}}{\frac{\dots}{\epsilon_{REV}}} \right)^{1/4}$$

$$\bar{u}_{EV} = \text{UNIT}(\bar{R}_{EV})$$

$$\bar{u}_{MV} = \text{UNIT}(\bar{R}_{MV})$$

All quantities are evaluated at the beginning of an itegration step. If h is greater then tgo, it is set to tgo for the last step.

Subroutine DELRV

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
H	h	Step size
RSV	$\bar{R}_{SV}(t+h)$	Conics plus perturbations at end of step.
REV	$\bar{R}_{EV}(t+h)$	
RMV	$\bar{R}_{MV}(t+h)$	
RSE	$\bar{R}_{SE}(t+h)$	
RSM	$\bar{R}_{SM}(t+h)$	
REM	$\bar{R}_{EM}(t+h)$	
CRSV	$[\bar{R}_{SV}(t+h)]$	Conics at end of step
CREV	$[\bar{R}_{EV}(t+h)]$	
CRMV	$[\bar{R}_{MV}(t+h)]$	
CRSE	$[\bar{R}_{SE}(t+h)]$	
CRSM	$[\bar{R}_{SM}(t+h)]$	
CREM	$[\bar{R}_{EM}(t+h)]$	
DR4RSV	$\overset{\dots}{\epsilon}_{RSV}(t)$	4th derivative of position error estimates at beginning of step.
DR4RSE	$\overset{\dots}{\epsilon}_{RSE}(t)$	
DR4RSM	$\overset{\dots}{\epsilon}_{RSM}(t)$	

B. Output Parameters

RRSV	$d\bar{R}_{SV}(t+h)$	Position and velocity corrections computed by quadrature formula.
RVSV	$d\bar{V}_{SV}(t+h)$	
RHSE	$d\bar{R}_{SE}(t+h)$	
RVSE	$d\bar{V}_{SE}(t+h)$	
RRSM	$d\bar{R}_{SM}(t+h)$	
RVSM	$d\bar{V}_{SM}(t+h)$	

C. Computation

Define

$$\bar{s}_{i,j} = \frac{[\bar{R}_{i,j}(t+h)] - \bar{R}_{i,j}}{|\bar{R}_{i,j}(t+h)| |\bar{R}_{i,j}|}$$

$$(i,j) = (SV, EV, MV, SE, SM, EM)$$

The second derivatives of Stumpff-Weiss position errors are given by

$$\ddot{\epsilon}_{1,RSV} = \mu_S \mu_M \bar{s}_{SV} + \mu_E (\bar{s}_{SE} + \bar{s}_{EV}) + \mu_M (\bar{s}_{SM} + \bar{s}_{MV})$$

$$\ddot{\epsilon}_{1,RSE} = (\mu_S + \mu_E) \bar{s}_{SE} + \mu_M (\bar{s}_{SM} - \bar{s}_{EM})$$

$$\ddot{\epsilon}_{1,RSM} = (\mu_S + \mu_M) \bar{s}_{SM} + \mu_E (\bar{s}_{SE} + \bar{s}_{EM})$$

The position and velocity corrections are computed by quadrature formulas below.

$$d\bar{R}_{SV} = \frac{h^2}{20} \left( \overset{\dots}{\epsilon}_{RSV} \frac{h^2}{3} + \overset{\dots}{\epsilon}_{1,RSV} \right)$$

$$d\bar{V}_{SV} = \frac{h}{4} \left( \overset{\dots}{\epsilon}_{RSV} \frac{h^2}{8} + \overset{\dots}{\epsilon}_{1,RSV} \right)$$

$$d\bar{R}_{SE} = \frac{h^2}{20} \left( \overset{\dots}{\epsilon}_{RSE} \frac{h^2}{3} + \overset{\dots}{\epsilon}_{1,RSE} \right)$$

$$d\bar{V}_{SE} = \frac{h}{4} \left( \overset{\dots}{\epsilon}_{RSE} \frac{h^2}{8} + \overset{\dots}{\epsilon}_{1,RSE} \right)$$

$$d\bar{R}_{SM} = \frac{h^2}{20} \left( \overset{\dots}{\epsilon}_{RSM} \frac{h^2}{3} + \overset{\dots}{\epsilon}_{1,RSM} \right)$$

$$d\bar{V}_{SM} = \frac{h}{4} \left( \overset{\dots}{\epsilon}_{RSM} \frac{h^2}{8} + \overset{\dots}{\epsilon}_{1,RSM} \right)$$

Subroutine DELRV3

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
H	h	Step size
REV	$\bar{R}_{EV}(t+h)$	Conics plus perturbations at end of step.
RMV	$\bar{R}_{MV}(t+h)$	
CREV	$[\bar{R}_{EV}(t+h)]$	Conics at end of step.
CRMV	$[\bar{R}_{MV}(t+h)]$	
DR4REV	$\overset{\dots}{\ddot{\epsilon}}_{REV}(t)$	4th derivative of position error estimate at beginning of step

B. Output Parameters

RREV	$d\bar{R}_{EV}(t+h)$	Position and velocity corrections computed by quadrature formula
RVEV	$d\bar{V}_{EV}(t+h)$	

C. Computation

Define

$$\bar{s}_{i,j} = \frac{[\bar{R}_{i,j}(t+h)]}{|[\bar{R}_{i,j}(t+h)]|}$$

$$(i, j) = (EV, MV)$$

The second derivative of Stumpff-Weiss position errors is given by

$$\overset{\dots}{\ddot{\epsilon}}_{1, REV} = \mu_E \bar{s}_{EV} + \mu_M \bar{s}_{MV}$$

The position and velocity corrections by quadrature formula are

$$d\bar{R}_{EV} = \frac{h^2}{20} \left( \overset{\dots}{\ddot{\epsilon}}_{REV} \frac{h^2}{3} + \overset{\dots}{\ddot{\epsilon}}_{1, REV} \right)$$

$$d\bar{V}_{EV} = \frac{h}{4} \left( \overset{\dots}{\ddot{\epsilon}}_{REV} \frac{h^2}{8} + \overset{\dots}{\ddot{\epsilon}}_{1, REV} \right)$$

Note that there is no error in moon's position and velocity in a 3-body space.

Subroutine COMIC

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
REVMAG	$r_o$	$ \bar{R}_{EV}(t_o) $
VEVMAG	$v_o$	$ \bar{V}_{EV}^+(t_o) $
LON	$\Omega$ (rad)	Longitude of ascending node
THE	$\theta$ (rad)	Position of vehicle from ascending line of node in orbital plane.
OINC	$i$ (rad)	Orbital inclination
OBL	OBL (rad)	Obliquity angles
RSEO	$\bar{R}_{SE}(t_o)$	
VSEO	$\bar{V}_{SE}(t_o)$	

B. Output Parameters

REVO	$\bar{R}_{EV}(t_o)$	
VEVO	$\bar{V}_{EV}^-(t_o)$	
RSVO	$\bar{R}_{SV}(t_o)$	
VSV0	$\bar{V}_{SV}^-(t_o)$	Velocity before impulse
VSVI	$\bar{V}_{SV}^+(t_o)$	Velocity after impulse

C. Computation

Matrix to transform a vector from O-frame to e-frame:

$$C_O^e = \begin{pmatrix} \cos \Omega & -\sin \Omega & 0 \\ \sin \Omega & \cos \Omega & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos i & -\sin i \\ 0 & \sin i & \cos i \end{pmatrix} \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Matrix to transform a vector from e-frame to E-frame:



$$C_e^E = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(OBL) & \sin(OBL) \\ 0 & -\sin(OBL) & \cos(OBL) \end{pmatrix}$$

$$\bar{R}_{EV}(t_o) = C_e^E C_O^e \begin{pmatrix} r_o \\ 0 \\ 0 \end{pmatrix}$$

$$\bar{V}_{EV}^+(t_o) = C_e^E C_O^e \begin{pmatrix} 0 \\ v_o \\ 0 \end{pmatrix}$$

$$\bar{R}_{SV}(t_o) = \bar{R}_{SE}(t_o) + \bar{R}_{EV}(t_o)$$

$$\bar{V}_{SV}^+(t_o) = \bar{V}_{SE}(t_o) + \bar{V}_{EV}^+(t_o)$$

$$\bar{V}_{EV}^-(t_o) = \text{UNIT} \left( \bar{V}_{EV}^+(t_o) \right) \sqrt{\frac{\mu_E}{r_o}}$$

$$\bar{V}_{SV}^-(t_o) = \bar{V}_{SE}(t_o) + \bar{V}_{EV}^-(t_o)$$

Subroutine COMIC3

A. Input Parameters

<u>Parameter</u>	<u>Symbol</u>	<u>Definition</u>
REVMAG	$r_o$	$ \bar{R}_{EV}(t_o) $
VEVMAG	$v_o$	$ \bar{V}_{EV}^+(t_o) $
LON	$\Omega$ (rad)	Longitude of ascending node
THE	$\theta$ (rad)	Position of vehicle from ascending node in orbital plane
OINC	$i$ (rad)	Orbital inclination

B. Output Parameters

REVO	$\bar{R}_{EV}(t_o)$
VEVO	$\bar{V}_{EV}^-(t_o)$
VEVOP	$\bar{V}_{EV}^+(t_o)$

C. Computation

$C_O^e$  = matrix to transform a vector from O-frame to e-frame  
(given in COMIC)

$$\bar{R}_{EV}(t_o) = C_O^e \begin{pmatrix} r_o \\ 0 \\ 0 \end{pmatrix}$$

$$\bar{V}_{EV}^+(t_o) = C_O^e \begin{pmatrix} 0 \\ v_o \\ 0 \end{pmatrix}$$

$$\bar{V}_{EV}^-(t_o) = \text{UNIT}(\bar{V}_{EV}^+(t_o)) \sqrt{\frac{\mu_E}{r_o}}$$

Subroutine COMFG

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
TSTART	$t_o$	
TEND	$t_f$	
REVMAG	$r_o$	$ \bar{R}_{EV}(t_o) $
VEVMAG	$v_o$	$ \bar{V}_{EV}^+(t_o) $
LON	$\Omega$ (rad)	Longitude of ascending node
THE	$\theta$ (rad)	Position of vehicle from ascending node in orbital plane
OINC	$i$ (rad)	Orbital inclination
OBL	OBL (rad)	Obliquity angle
RSEO	$\bar{R}_{SE}(t_o)$	
VSEO	$\bar{V}_{SE}(t_o)$	
RSMO	$\bar{R}_{SM}(t_o)$	
VSMO	$\bar{V}_{SM}(t_o)$	

B. Output Parameters

FD	$\Delta V$	$\Delta V =  \bar{V}_{SVTAR} - \bar{V}_{SV}(t_f) $
TESTRD	$\epsilon$	$\epsilon =  \bar{\psi} $
GD	$\bar{g}$	$\bar{g} = \frac{\partial \Delta V}{\partial (v_o, \Omega, \theta)}$
TSID	$\bar{\psi}$	$\bar{\psi} = \bar{R}_{SV}(t_f) - \bar{R}_{SVTAR}$
LTD	$L^T$	$L^T = \frac{\partial \bar{\psi}}{\partial (v_o, \Omega, \theta)}$
S11, S12, S21, S22	$\phi(t_f, t_o)$	$\phi = \frac{\partial (\bar{R}_{SV}(t_f), \bar{V}_{SV}(t_f))}{\partial (\bar{R}_{SV}(t_o), \bar{V}_{SV}(t_o))}$
UVI	$\bar{u}_{VI}$	$\bar{u}_{VI} = \text{UNIT} (\bar{V}_{SV}^+(t_o) - \bar{V}_{SV}^-(t_o))$

UVF

 $u_{VF}$ 

$$u_{VF} = \text{UNIT} (\bar{V}_{SVTAR} - \bar{V}_{SV}(t_f))$$

C. Computation

$$\bar{x} = (v_o, \Omega, \theta)$$

Call COMIC to compute  $\bar{R}_{SV}(t_o)$ ,  $\bar{V}_{SV}^-(t_o)$ ,  $\bar{V}_{SV}^+(t_o)$ .

Call FOURBY to advance states to  $t_f$  and compute  $\phi$ .

For the first iteration, call CTAR to compute  $\bar{R}_{SVTAR}$ ,  $\bar{V}_{SVTAR}$ .

$$\Delta \bar{v}_o = \bar{V}_{SV}^+(t_o) - \bar{V}_{SV}^-(t_o)$$

$$u_{VI} = \text{UNIT} (\Delta \bar{v}_o)$$

$$\Delta \bar{v}_f = \bar{V}_{SVTAR} - \bar{V}_{SV}(t_f)$$

$$u_{VF} = \text{UNIT} (\Delta \bar{v}_f)$$

$$\Delta v = |\Delta \bar{v}_f|$$

$$\bar{g} = \frac{\partial \Delta v}{\partial \bar{x}} = - \frac{\Delta \bar{v}_f^T}{\Delta v} \left( \phi_{21} \frac{\partial \bar{R}_{SV}(t_o)}{\partial \bar{x}} + \phi_{22} \frac{\partial \bar{V}_{SV}(t_o)}{\partial \bar{x}} \right)$$

$$\bar{\psi} = \bar{R}_{SV}(t_f) - \bar{R}_{SVTAR}$$

$$L^T = \frac{\partial \bar{\psi}}{\partial \bar{x}} = \phi_{11} \frac{\partial \bar{R}_{SV}(t_o)}{\partial \bar{x}} + \phi_{12} \frac{\partial \bar{V}_{SV}(t_o)}{\partial \bar{x}}$$

$$\bar{R}_{SV}(t_o) = \bar{R}_{SE}(t_o) + \bar{R}_{EV}(t_o)$$

$$\bar{V}_{SV}(t_o) = \bar{V}_{SE}(t_o) + \bar{V}_{EV}^+(t_o)$$

$$\frac{\partial \bar{R}_{SV}(t_o)}{\partial \bar{x}} = \frac{\partial \bar{R}_{EV}(t_o)}{\partial \bar{x}} = A$$

$$A(1, 1) = 0$$

$$A(1, 2) = r_o (-\sin \Omega \cos \theta - \cos \Omega \cos i \sin \theta)$$

$$A(1, 3) = r_o (-\cos \Omega \sin \theta - \sin \Omega \cos i \cos \theta)$$

$$A(2, 1) = 0$$

$$A(2, 2) = r_o \cos(\text{OBL.}) (\cos \Omega \cos \theta - \sin \Omega \cos i \sin \theta)$$

$$A(2, 3) = r_o [\cos(\text{OBL.}) (-\sin \Omega \sin \theta + \cos \Omega \cos i \cos \theta) \\ + \sin(\text{OBL.}) \sin i \cos \theta]$$

$$A(3, 1) = 0$$

$$A(3, 2) = -r_o \sin(\text{OBL.}) (\cos \Omega \cos \theta - \sin \Omega \cos i \sin \theta)$$

$$A(3, 3) = r_o [-\sin(\text{OBL.}) (-\sin \Omega \sin \theta + \cos \Omega \cos i \cos \theta) \\ + \sin(\text{OBL.}) \sin i \cos \theta]$$

$$\frac{\partial \bar{V}_{SV}(t_o)}{\partial \bar{x}} = \frac{\partial \bar{V}_{EV}^+(t_o)}{\partial \bar{x}} = B$$

$$B(1, 1) = -\cos \Omega \sin \theta - \sin \Omega \cos i \cos \theta$$

$$B(1, 2) = v_o (\sin \Omega \sin \theta - \cos \Omega \cos i \cos \theta)$$

$$B(1, 3) = v_o (-\cos \Omega \cos \theta + \sin \Omega \cos i \sin \theta)$$

$$B(2, 1) = \cos(\text{OBL.}) (-\sin \Omega \sin \theta + \cos \Omega \cos i \cos \theta) \\ + \sin(\text{OBL.}) \sin i \cos \theta$$

$$B(2, 2) = v_o \cos(\text{OBL.}) (-\cos \Omega \sin \theta - \sin \Omega \cos i \cos \theta)$$

$$B(2, 3) = v_o [\cos(\text{OBL.}) (-\sin \Omega \cos \theta - \cos \Omega \cos i \sin \theta) \\ - \sin(\text{OBL.}) \sin i \sin \theta]$$

$$B(3, 1) = -\sin(\text{OBL}) (-\sin \Omega \sin \theta + \cos \Omega \cos i \cos \theta) \\ + \cos(\text{OBL}) \sin i \cos \theta$$

$$B(3, 2) = -v_o \sin(\text{OBL}) (-\cos \Omega \sin \theta - \sin \Omega \cos i \cos \theta)$$

$$B(3, 3) = v_o [-\sin(\text{OBL}) (-\sin \Omega \cos \theta - \cos \Omega \cos i \sin \theta) \\ - \cos(\text{OBL}) \sin i \sin \theta]$$

Subroutine COMFG3

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
TSTART	$t_o$	
TEND	$t_f$	
REVMAG	$r_o$	$ \bar{R}_{EV}(t_o) $
VEVMAG	$v_o$	$ \bar{V}_{EV}^+(t_o) $
LON	$\Omega$ (rad)	Longitude of ascending node
THE	$\theta$	Position of vehicle from ascending node in orbital plane
OINC	$i$ (rad)	Orbital inclination
REMO	$\bar{R}_{EM}(t_o)$	
VEMO	$\bar{V}_{EM}(t_o)$	

B. Output Parameters

FD	$\Delta V$	$\Delta V =  \bar{V}_{EVTAR} - \bar{V}_{EV}(t_f) $
TESTRD	$\epsilon$	$\epsilon =  \bar{\psi} $
GD	$\bar{g}$	$\bar{g} = \frac{\partial \Delta V}{\partial (v_o, \Omega, \theta)}$
TSID	$\bar{\psi}$	$\bar{\psi} = \bar{R}_{EV}(t_f) - \bar{R}_{EVTAR}$
LTD	$L^T$	$L^T = \frac{\partial \bar{\psi}}{\partial (v_o, \Omega, \theta)}$
S11, S12, S21, S22	$\phi(t_f, t_o)$	$\phi = \frac{\partial (\bar{R}_{EV}(t_f), \bar{V}_{EV}(t_f))}{\partial (\bar{R}_{EV}(t_o), \bar{V}_{EV}(t_o))}$
UVI u	$\bar{u}_{VI}$	$\bar{u}_{VI} = \text{UNIT} (\bar{V}_{EV}^+(t_o) - \bar{V}_{EV}^-(t_o))$
UVF	$\bar{u}_{VF}$	$\bar{u}_{VF} = \text{UNIT} (\bar{V}_{EVTAR} - \bar{V}_{EV}(t_f))$

C. Computation

Call COMIC3 to compute  $\bar{R}_{EV}(t_0)$ ,  $\bar{V}_{EV}^-(t_0)$ ,  $\bar{V}_{EV}^+(t_0)$ .

Call TURBDY to advance states to  $t_1$  and compute  $\phi$ .

For the first iteration, call CTAR3 to compute  $\bar{R}_{EVTAR}$  and  $\bar{V}_{EVTAR}$ .

Equations are same as in COMFG with  $\cos(OBL) = 1$  and  $\sin(OBL) = 0$ .



Subroutine COMAUG

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
F	f	$\Delta V$
G	$\bar{g}$	$\bar{g} = \frac{\partial \Delta V}{\partial (v_o, \Omega, \theta)}$ Constraint gradient transposed
L <sup>T</sup>	L <sup>T</sup>	$L^T = \frac{\partial \bar{\psi}}{\partial (v_o, \Omega, \theta)}$
V	V	Davidon's Variance matrix
TSI	$\bar{\psi}$	Constraint violation $\bar{\psi} = \bar{R}(t_f) - \bar{R}_{fd}$ $\bar{R}_{fd} = \bar{R}_{SVTAR}$ or $\bar{R}_{EVTAR}$

B. Output Parameters

L	L	$L = (L^T)^T$
FG		$f + \bar{v}^T \bar{\psi}$
GG		$\bar{g} + L \bar{v}$

C. Computation

$$L = (L^T)^T$$

$$\bar{v} = -[L^T V L]^{-1} L^T V \bar{g}$$

$$FG = f + \bar{v}^T \bar{\psi}$$

$$GG = \bar{g} + L \bar{v}$$

## Subroutine CTAR

### A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
RSEF	$\bar{R}_{SE}(t_f)$	
VSEF	$\bar{V}_{SE}(t_f)$	
RSMF	$\bar{R}_{SM}(t_f)$	
VSMF	$\bar{V}_{SM}(t_f)$	
REMF	$\bar{R}_{EM}(t_f)$	
VEMF	$\bar{V}_{EM}(t_f)$	
AY	A <sub>y</sub>	Parameters to define a point on a Halo orbit.
AZ	A <sub>z</sub>	
ATAR	A <sub>TAR</sub>	

### B. Output Parameters

RSVTAR	$\bar{R}_{SVTAR}^S$
VSVTAR	$\bar{V}_{SVTAR}^S$

### C. Computation

Position and velocity of L<sub>1</sub> in S-frame:

$$\bar{R}_{SL1}^S = (1 - \gamma_{L1}) \left( \bar{R}_{SE}^S + \frac{\mu_M}{\mu_E + \mu_M} \bar{R}_{EM}^S \right)$$

$$\bar{V}_{SL1}^S = (1 - \gamma_{L1}) \left( \bar{V}_{SE}^S + \frac{\mu_M}{\mu_E + \mu_M} \bar{V}_{EM}^S \right)$$

If both A<sub>y</sub> and A<sub>z</sub> are zero, the following computation is omitted and target position and velocity are set equal to that of L<sub>1</sub>. Define L-frame unit vectors.

$$\bar{u}_{XL}^S = \text{UNIT} \left( \bar{R}_{SL1}^S \right)$$

$$\bar{u}_{ZL}^S = \text{UNIT} \left( \mu_E \bar{R}_{SE}^S \times \bar{V}_{SE}^S + \mu_M \bar{R}_{SM}^S \times \bar{V}_{SM}^S \right)$$

$$\bar{u}_{YL}^S = \bar{u}_{ZL}^S \times \bar{u}_{XL}^S$$

The matrix to transform a vector from S-frame to L-frame is given by

$$C_S^L = \left( (\bar{u}_{XL}^S)^T, (\bar{u}_{YL}^S)^T, (\bar{u}_{ZL}^S)^T \right)$$

The angular velocity of L-frame with respect to S-frame in S-frame is defined by

$$\bar{\omega}_{SL}^S = \bar{u}_{ZL}^S \left( \bar{u}_{YL}^S \cdot \frac{\bar{V}_{SL1}^S}{|\bar{R}_{SL1}^S|} \right)$$

The target position and velocity in L-frame are defined by

$$\bar{R}_{LITAR}^L = \begin{pmatrix} k & A_y & \sin A_{TAR} \\ & A_y & \cos A_{TAR} \\ & A_z & \sin A_{TAR} \end{pmatrix}$$

$$\bar{V}_{LITAR}^L = \begin{pmatrix} k & \omega_n & A_y & \cos A_{TAR} \\ - & \omega_n & A_y & \sin A_{TAR} \\ & \omega_n & A_z & \cos A_{TAR} \end{pmatrix}$$

where

$$\omega_n = \sqrt{1 - \frac{B_L}{2}} + \sqrt{\left(\frac{3B_L}{2}\right)^2 - 2B_L}$$

$$B_L = \frac{1 - \mu}{(1 - \gamma_{L1})^3} + \frac{\mu}{\gamma_{L1}^3}$$

$$\mu = \frac{\mu_E + \mu_M}{\mu_S + \mu_E + \mu_M}$$

$$k = \frac{2 \omega_n}{\omega_n^2 + (2B_L + 1)}$$

The target position and velocity in S-frame are then given by

$$\bar{R}_{SVTAR}^S = \bar{R}_{SL1}^S + (C_S^L)^T \bar{R}_{LITAR}^L$$

$$\bar{V}_{SVTAR}^S = \bar{V}_{SL1}^S + (C_S^L)^T \bar{V}_{LITAR}^L + \bar{\omega}_{SL}^S \times (C_S^L)^T \bar{R}_{LITAR}^L$$

Subroutine CTAR3

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
REMF	$\bar{R}_{EM}(t_f)$	In e-frame
VEMF	$\bar{V}_{EM}(t_f)$	
AY	Ay	Parameters to define a point on a Halo orbit
AZ	Az	
ATAR	A <sub>TAR</sub>	

B. Output Parameters

REVTAR	$\bar{R}_{EVTAR}^e$
VEVTAR	$\bar{V}_{EVTAR}^e$

C. Computation

Position and velocity of libration point in e-frame

$$\bar{R}_{EL}^e = (1 - \gamma) \bar{R}_{EM}^e \quad \gamma > 0 \text{ for } L_1$$

$$\bar{V}_{EL}^e = (1 - \gamma) \bar{V}_{EM}^e \quad < 0 \text{ for } L_2$$

If both Ay and Az are zero, the following computation is omitted and target position and velocity are set equal to that of the libration point. Define l-frame by unit vectors:

$$\bar{u}_{XL}^e = \text{UNIT}(\bar{R}_{EL}^e)$$

$$\bar{u}_{ZL}^e = \text{UNIT}(\bar{R}_{EM}^e \times \bar{V}_{EM}^e)$$

$$\bar{u}_{YL}^e = \bar{u}_{ZL}^e \times \bar{u}_{XL}^e$$

The matrix to transform a vector from e-frame to l-frame is given by

$$C_e^l = ((\bar{u}_{XL}^e)^T, (\bar{u}_{YL}^e)^T, (\bar{u}_{ZL}^e)^T)$$

The angular velocity of  $\ell$ -frame with respect to  $e$ -frame in  $e$ -frame is defined by

$$\bar{\omega}_{EL}^e = \bar{u}_{ZL}^e \left( \bar{u}_{YL}^e \cdot \frac{\bar{V}_{EL}^e}{|\bar{R}_{EL}^e|} \right)$$

The target position and velocity in  $\ell$ -frame are defined by

$$\bar{R}_{LTAR}^{\ell} = \begin{pmatrix} k & A_y & \sin A_{TAR} \\ & A_y & \cos A_{TAR} \\ & A_z & \sin A_{TAR} \end{pmatrix}$$

$$\bar{V}_{LTAR}^{\ell} = \begin{pmatrix} k & \omega_n & A_y & \cos A_{TAR} \\ - & \omega_n & A_y & \sin A_{TAR} \\ & \omega_n & A_z & \cos A_{TAR} \end{pmatrix}$$

where

$$\omega_n = \sqrt{1 - \frac{B_L}{2}} + \sqrt{\left(\frac{3B_L}{2}\right)^2 - 2B_L}$$

$$B_L = \frac{1 - \mu}{(1 - \gamma)^3} + \frac{\mu}{\gamma^3}$$

$$\mu = \frac{\mu_M}{\mu_E + \mu_M}$$

$$k = \frac{2\omega_n}{\omega_n^2 + (2B_L + 1)}$$

The target position and velocity in  $e$ -frame are then given by

$$\bar{R}_{EVTAR}^e = \bar{R}_{EL}^e + (C_e^{\ell})^T \bar{R}_{LTAR}^{\ell}$$

$$\bar{V}_{EVTAR}^e = \bar{V}_{EL}^e + (C_e^{\ell})^T \bar{V}_{LTAR}^{\ell} + \bar{\omega}_{EL}^e \times (C_e^{\ell})^T \bar{R}_{LTAR}^{\ell}$$

Subroutine COMDX

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
LTS	$L^T$	$\frac{\partial \bar{\psi}}{\partial (v_o, \Omega, \theta)}$
LS	L	
TSIS	$\psi$	$\bar{\psi} = \bar{R}(t_f) - \bar{R}_{fd}$ $\bar{R}_{fd} = \bar{R}_{SVTAR}$ or $\bar{R}_{EVTAR}$

B. Output Parameters

DX	$d\bar{x}$	$d\bar{x} = (dv_o, d\Omega, d\theta)$
----	------------	---------------------------------------

C. Computation

$$d\bar{x} = -L(L^T L)^{-1} \bar{\psi}$$

Subroutine RVEMV

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
RSV	$\bar{R}_{SV}(t)$	
VSV	$\bar{V}_{SV}(t)$	
RSE	$\bar{R}_{SE}(t)$	
VSE	$\bar{V}_{SE}(t)$	
RSM	$\bar{R}_{SM}(t)$	
VSM	$\bar{V}_{SM}(t)$	

B. Output Parameters

REV	$\bar{R}_{EV}(t)$
VEV	$\bar{V}_{EV}(t)$
RMV	$\bar{R}_{MV}(t)$
VMV	$\bar{V}_{MV}(t)$
REM	$\bar{R}_{EM}(t)$
VEM	$\bar{V}_{EM}(t)$

C. Computation

$$\bar{R}_{EV} = -\bar{R}_{SE} + \bar{R}_{SV}$$

$$\bar{V}_{EV} = -\bar{V}_{SE} + \bar{V}_{SV}$$

$$\bar{R}_{MV} = -\bar{R}_{SM} + \bar{R}_{SV}$$

$$\bar{V}_{MV} = -\bar{V}_{SM} + \bar{V}_{SV}$$

$$\bar{R}_{EM} = -\bar{R}_{SE} + \bar{R}_{SM}$$

$$\bar{V}_{EM} = -\bar{V}_{SE} + \bar{V}_{SM}$$

This subroutine is used in 4-body program only.

Subroutine UPX

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
XD	$\bar{X}_d$	
FD	$f_d$	
TESTRD	$ \bar{\psi}_d $	
GD	$\bar{g}_d$	
TSID	$\bar{\psi}_d$	
LTD	$(L^T)_d$	
LD	$L_d$	
FGD		$(f + \bar{v}^T \bar{\psi})_d$
GGD		$(\bar{g} + L\bar{v})_d$
S1D, S12D, S21D, S22D	$\varphi_d(t_f, t_o)$	
UVID, UVFD	$\bar{\lambda}_d(t_o), \bar{\lambda}_d(t_f)$	

B. Output Parameters

Same as input parameters relabelled as X, F, TESTR G, TSL, LT, L, FG, GG, S11, S12, S21, S22, UVI, UVF.

C. Computation

Set output parameters = input parameters with new names.



Subroutine LAMB

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
T0	$t_0$	
TF	$t_f$	
RSV	$\bar{R}_{SV}(t_0)$	
VSV	$\bar{V}_{SV}(t_0)$	Initial estimate
RSE	$\bar{R}_{SE}(t_0)$	
VSE	$\bar{V}_{SE}(t_0)$	
RSM	$\bar{R}_{SM}(t_0)$	
VSM	$\bar{V}_{SM}(t_0)$	
KNR		Lambert iteration parameter
ITLMAX		Maximum number of iterations
ERRMIN		Convergence criterion
RSVTAR	$\bar{R}_{SVTAR}$	

B. Output Parameters

RSVF	$\bar{R}_{SV}(t_f)$	
VSVF	$\bar{V}_{SV}(t_f)$	
RSEF	$\bar{R}_{SE}(t_f)$	
VSEF	$\bar{V}_{SE}(t_f)$	
RSMF	$\bar{R}_{SM}(t_f)$	
VSMF	$\bar{V}_{SM}(t_f)$	
VSV	$\bar{V}_{SV}(t_0)$	Final solution
S11, S12, S21, S22	$\phi(t_f, t_0)$	State transition matrix

C. Computation

Call FOURBY to advance states to  $t_f$ . Newton-Raphson method is used which iterates on  $\bar{V}_{SV}(t_0)$  until terminal constraint violation is less than ERRMIN or the number of iteration exceeds ITERL. The desired change in initial velocity is given by

$$d\bar{V} = -(\phi_{12})^{-1} (\bar{R}_{SVTAR} - \bar{R}_{SV}(t_f))$$

The subroutine has a built-in safeguard against inversion of a singular matrix when the transfer is 2-dimensional.

D. Internal Parameters

VIV

Dummy iteration variables of initial velocity.

ERR

Error vector of terminal constraint violation

TESTR

Magnitude of ERR.

Subroutine LAMB3

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
T0	$t_0$	
TF	$t_f$	
REV	$\bar{R}_{EV}(t_0)$	Initial estimate
VEV	$\bar{V}_{EV}(t_0)$	
REM	$\bar{R}_{EM}(t_0)$	
VEM	$\bar{V}_{EM}(t_0)$	
KNR		
ITLMAX		
ERRMIN		
REVTAR		

B. Output Parameters

REVF	$\bar{R}_{EV}(t_f)$	
VEVF	$\bar{V}_{EV}(t_f)$	
REMF	$\bar{R}_{EM}(t_f)$	
VEMF	$\bar{V}_{EM}(t_f)$	
VEV	$\bar{V}_{EV}(t_0)$	Final Solution
S11, S12, S21, S22	$\phi(t_f, t_0)$	

C. Computation

Call THRBDY to advance states to  $t_f$ . Use Newton-Raphson method to iterate on  $\bar{V}_{EV}(t_0)$  until terminal constraint violation is less than ERRMIN or the number of iteration exceeds ITERL.

The desired change in initial velocity is given by

$$d\bar{V} = -(\phi_{12})^{-1} (\bar{R}_{EVTAR} - \bar{R}_{EV}(t_f))$$

The subroutine has a built-in safeguard against inversion of a singular matrix when the transfer is 2-dimensional.

d. Internal Parameters

Same as listed under LAMB.

Subroutine COMF

A. Input Parameters

<u>Parameter</u>	<u>Symbol</u>	<u>Definition</u>
TSTART	$t_o$	Starting time
TM	$t_m$	Interior impulse time
TEND	$t_f$	Terminal time
RSV0	$\bar{R}_{SV}(t_o)$	
VSVI	$\bar{V}_{SV}^+(t_o)$	
VSV0	$\bar{V}_{SV}^-(t_o)$	
RSE0	$\bar{R}_{SE}(t_o)$	
VSE0	$\bar{V}_{SE}(t_o)$	
RSM0	$\bar{R}_{SM}(t_o)$	
VSM0	$\bar{V}_{SM}(t_o)$	
ERRMIN		Allowable position error
ILJNC		Number of increments in solving Lambert problem
KNRSAV		Lambert iteration parameter
ITERD		Outer loop iteration number
ITLMAX		Maximum number of Lambert iterations
VSVMPD	$\bar{V}_{SV}^+(t_m), \text{ OLD}$	
RSVMD	$\bar{R}_{SV}(t_m), \text{ OLD}$	
RSVTAR	$\bar{R}_{SVTAR}$	Target position
VSVTAR	$\bar{V}_{SVTAR}$	Target velocity
SI1D	$\phi_{11}(t_f, t_m) \text{ OLD}$	
SI2D	$\phi_{12}(t_f, t_m) \text{ OLD}$	

B. Output Parameters

RSVM	$\bar{R}_{SV}(t_m)$	
VSVMP	$\bar{V}_{SV}^+(t_m)$	
VSVMM	$\bar{V}_{SV}^-(t_m)$	
DV	$\Delta V$	
UVI	$\bar{u}_{VI}$	
UVM	$\bar{u}_{VM}$	
UVF	$\bar{u}_{VF}$	
SMI11		
SMI12		
SMI21	$\phi(t_m, t_0)$	State transition matrix from $t_0$ to $t_m$ .
SMI22		
SFM11		
SFM12		
SFM21	$\phi(t_f, t_m)$	State transition matrix from $t_m$ to $t_f$ .
SFM22		

C. Computation

- Step 1. Call FOURBY to advance states from  $t_0$  to  $t_m$ .
- Step 2. If ITERD  $\neq$  0, go to step 4.
- Step 3. Set  $\bar{V}_{SVX}^+(t_m) = \bar{V}_{SV}^+(t_m)$   
 $\bar{R}_{SVX}(t_m) = \bar{R}_{SV}(t_m)$   
 Go to step 6.
- Step 4. Solve second leg Lambert problem in increments  
 $\bar{R}_{SVX}(t_m) = \bar{R}_{SV}(t_m)_{OLD}$   
 $\bar{V}_{SVX}^+(t_m) = \bar{V}_{SV}^+(t_m)_{OLD}$   
 $d\bar{R}_m = (\bar{R}_{SV}(t_m) - \bar{R}_{SV}(t_m)_{OLD}) / ILINC$   
 Set  $\phi_{11}(t_f, t_m) = \phi_{11, OLD}$   
 $\phi_{12}(t_f, t_m) = \phi_{12, OLD}$   
 ITERL = 0

Step 5.  $d\bar{V}_m = -\phi_{12}(t_f, t_m)^{-1} \phi_{11}(t_f, t_m) d\bar{R}_m$

$$\bar{R}_{SVX}(t_m) = \bar{R}_{SVX}(t_m) + d\bar{R}_m$$

$$\bar{V}_{SVX}^+(t_m) = \bar{V}_{SVX}^+(t_m) + d\bar{V}_m$$

Step 6. Call LAMB

$$ITERL = ITERL + 1$$

If ITERL < ILINC, go to step 5.

Step 7.  $\bar{V}_{SV}^+(t_m) = \bar{V}_{SVX}^+(t_m)$

Step 8.  $\Delta\bar{V}_o = \bar{V}_{SV}^+(t_o) - \bar{V}_{SV}^-(t_o)$

$$\Delta\bar{V}_m = \bar{V}_{SV}^+(t_m) - \bar{V}_{SV}^-(t_m)$$

$$\Delta\bar{V}_f = \bar{V}_{SVTAR} - \bar{V}_{SV}(t_f)$$

$$\bar{u}_{VI} = \text{UNIT}(\Delta\bar{V}_o)$$

$$\bar{u}_{VM} = \text{UNIT}(\Delta\bar{V}_m)$$

$$\bar{u}_{VF} = \text{UNIT}(\Delta\bar{V}_f)$$

$$\Delta V = |\Delta\bar{V}_o| + |\Delta\bar{V}_m| + |\Delta\bar{V}_f|$$

Step 9. EXIT

Subroutine COMF3

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
TSTART	$t_o$	
TM	$t_m$	
TEND	$t_f$	
REVO	$\bar{R}_{EV}(t_o)$	
VEVOP	$\bar{V}_{EV}^+(t_o)$	
VEVO	$\bar{V}_{EV}^-(t_o)$	
REMO	$\bar{R}_{EM}(t_o)$	
VEMO	$\bar{V}_{EM}(t_o)$	
ERRMIN		Allowable position error
ILINC		Number of increments to solve Lambert problem
KNRSAV		Lambert iteration parameter
ITERD		Iteration number of outer loop
ITLMAX		Maximum Lambert iterations
VEVMPD	$\bar{V}_{EV}^+(t_m)_{OLD}$	
REVMD	$\bar{R}_{EV}(t_m)_{OLD}$	
REVTAR	$\bar{R}_{EVTAR}$	
VEVTAR	$\bar{V}_{EVTAR}$	
S1D	$\phi_{11}(t_f, t_m)_{OLD}$	
S12D	$\phi_{12}(t_f, t_m)_{OLD}$	



B. Output Parameters

REVM	$\bar{R}_{EV}(t_m)$
VEVMP	$\bar{V}_{EV}^+(t_m)$
VEVMM	$\bar{V}_{EV}^-(t_m)$
DV	$\Delta V$
UVI	$\bar{u}_{VI}$
UVM	$\bar{u}_{VM}$
UVF	$\bar{u}_{VF}$
SMI1	$\varphi(t_m, t_0)$
SMI2	
SMI21	
SMI22	
SFM11	$\varphi(t_f, t_m)$
SFM12	
SFM21	
SFM22	
DVM	$\Delta \bar{V}_m$

C. Computation

Step 1. Call THRBODY to advance states from  $t_0$  to  $t_m$ .

Step 2. If ITERD  $\neq$  0, go to step 4.

Step 3. Set  $\bar{V}_{EVX}^+(t_m) = \bar{V}_{EV}^+(t_m)$

$$\bar{R}_{EVX}(t_m) = \bar{R}_{EV}(t_m)$$

Go to step 6.

Step 4. Solve second leg Lambert problem

$$\bar{R}_{EVX}(t_m) = \bar{R}_{EV}(t_m)_{OLD}$$

$$\bar{V}_{EVX}^+(t_m) = \bar{V}_{EV}^+(t_m)_{OLD}$$

$$d\bar{R}_M = (\bar{R}_{EV}(t_m) - \bar{R}_{EV}(t_m)_{OLD}) / ILINC$$

$$\text{Set } \varphi_{11}(t_f, t_m) = \varphi_{11, OLD}$$

$$\varphi_{12}(t_f, t_m) = \varphi_{12, OLD}$$

ITERL = 0

Step 5.  $d\bar{V}_m = -\phi_{12}(t_f, t_m)^{-1} \phi_{11}(t_f, t_m) d\bar{R}_m$

$$\bar{R}_{EVX}(t_m) = \bar{R}_{EVX}(t_m) + d\bar{R}_m$$

$$\bar{V}_{EVX}^+(t_m) = \bar{V}_{EVX}^+(t_m) + d\bar{V}_m$$

Step 6. Call LAMB3

$$ITERL = ITERL + 1$$

If ITERL < ILINC, go to step 5.

Step 7.  $\bar{V}_{EV}^+(t_m) = \bar{V}_{EVX}^+(t_m)$

Step 8.  $\Delta\bar{V}_o = \bar{V}_{EV}^+(t_o) - \bar{V}_{EV}^-(t_o)$

$$\Delta\bar{V}_m = \bar{V}_{EV}^+(t_m) - \bar{V}_{EV}^-(t_m)$$

$$\Delta\bar{V}_f = \bar{V}_{EVTAR} - \bar{V}_{EV}(t_f)$$

$$\bar{u}_{V1} = \text{unit}(\Delta\bar{V}_o)$$

$$\bar{u}_{VM} = \text{unit}(\Delta\bar{V}_m)$$

$$\bar{u}_{VF} = \text{unit}(\Delta\bar{V}_f)$$

$$\Delta V = |\Delta\bar{V}_o| + |\Delta\bar{V}_m| + |\Delta\bar{V}_f|$$

Step 9. EXIT

Subroutine COMG

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
SMI11	$\phi_{11}(t_m, t_o)$	
SMI12	$\phi_{12}(t_m, t_o)$	
SMI21	$\phi_{21}(t_m, t_o)$	
SMI22	$\phi_{22}(t_m, t_o)$	
SFM11	$\phi_{11}(t_f, t_m)$	
SFM12	$\phi_{12}(t_f, t_m)$	
SFM21	$\phi_{21}(t_f, t_m)$	
SFM22	$\phi_{22}(t_f, t_m)$	
UVI	$\bar{u}_{VI}$	
UVM	$\bar{u}_{VM}$	
UVF	$\bar{u}_{VF}$	
VTMM	$v^-(t_m)$	
VTMP	$v^+(t_m)$	

B. Output Parameter

G	$\bar{g}$	Cost gradient $g = \frac{\partial \Delta V}{\partial (V^+(t_o), t_m)}$
DUVI	$\dot{\lambda}^-(t_o)$	
DUVMM	$\dot{\lambda}^-(t_m)$	
DUVMP	$\dot{\lambda}^+(t_m)$	

C. Computation

$$\dot{\lambda}^-(t_o) = \phi_{mo, 12}^{-1} (\bar{u}_{VM} - \phi_{mo, 11} \bar{u}_{VI})$$

$$\dot{\lambda}^-(t_m) = \varphi_{mo, 21} \bar{u}_{VI} + \varphi_{mo, 22} \dot{\lambda}^-(t_o)$$

$$\dot{\lambda}^+(t_m) = \varphi_{fm, 12}^{-1} (\bar{u}_{VF} - \varphi_{fm, 11} \bar{u}_{VM})$$

$$\frac{d}{dt} |\dot{\lambda}_m^-| = \dot{\lambda}_m^- \cdot \bar{u}_{VM}$$

$$\frac{d}{dt} |\dot{\lambda}_m^+| = \dot{\lambda}_m^+ \cdot \bar{u}_{VM}$$

$$\frac{\partial \Delta V}{\partial \bar{v}_o} = \varphi_{mo, 12}^T (\dot{\lambda}_m^+ - \dot{\lambda}_m^-)$$

$$\frac{\partial \Delta V}{\partial t_m} = -\dot{\lambda}_m^+ \cdot (v_m^+ - v_m^-)$$

$$\bar{g} = \left( \frac{\partial \Delta V}{\partial \bar{v}_o}, \frac{\partial \Delta V}{\partial t_m} \right)$$

NOTE: In addition to the output parameters the following parameters are also printed out after each iteration.

<u>Parameter</u>	<u>Symbol</u>
PVMMAG	$\frac{d}{dt}  \dot{\lambda}_m^- $
PVPMAG	$\frac{d}{dt}  \dot{\lambda}_m^+ $

Subroutine PVEC

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
T	t	
R	R	
V	V	
PV0	$PV(t_0)$	
STM	$\phi(t, t_0)$	State transition matrix
IPVTM		IPVTM = 0, Advance PV only 1, Also monitor max. value of primer vector magnitude.

B. Output Parameters

PV	$PV(t)$	$PV(t) = (\bar{\lambda}(t), \dot{\bar{\lambda}}(t))$
LM	$ \bar{\lambda} $	
LDM	$\frac{d}{dt}  \bar{\lambda} $	

The following parameters are transmitted through COMMON/CTM/.

TTM	$t_m$	Time of maximum $ \bar{\lambda} $
RTM	$\bar{R}(t_m)$	
VTM	$\bar{V}(t_m)$	
LTM	$ \bar{\lambda}(t_m) $	
LDTM	$\frac{d}{dt}  \bar{\lambda}(t_m) $	
PVTM	$\bar{\lambda}(t_m)$	
PVDTM	$\dot{\bar{\lambda}}(t_m)$	
STNTM	$\phi(t_m', t_0)$	$t_m'$ = time when the primer vector magnitude is maximum.

C. Computation

$$\begin{pmatrix} \bar{\lambda}(t) \\ \dot{\bar{\lambda}}(t) \end{pmatrix} = \varphi(t, t_0) \begin{pmatrix} \bar{\lambda}(t_0) \\ \dot{\bar{\lambda}}(t_0) \end{pmatrix}$$

If IPVTM  $\neq 0$ , monitor  $|\bar{\lambda}(t)|$  to determine maximum value.

Subroutine DISP

Both input and output parameters are transmitted through COMMON/TDATA/.

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
RSV	$\bar{R}_{SV}^S$	
VSV	$\bar{V}_{SV}^S$	
REV	$\bar{R}_{EV}^E$	
VEV	$\bar{V}_{EV}^E$	
RMV	$\bar{R}_{MV}^M$	
VMV	$\bar{V}_{MV}^M$	
RSE	$\bar{R}_{SE}^S$	
VSE	$\bar{V}_{SE}^S$	
RSM	$\bar{R}_{SM}^S$	
VSM	$\bar{V}_{SM}^S$	
REM	$\bar{R}_{EM}^E$	
VEM	$\bar{V}_{EM}^E$	

B. Output Parameters

RSL1	$\bar{R}_{SL1}^S$	
VSL1	$\bar{V}_{SL1}^S$	
RDVD	$\bar{R}_{DV}^D$	} Position and velocity of vehicle in D-frame
VDVD	$\bar{V}_{DV}^D$	
RDS1	$\bar{R}_{DS}^D$	
VDS1	$\bar{V}_{DS}^D$	

RDMD	$\bar{R}_{DM}^D$	
VDMD	$\bar{V}_{DM}^D$	
RDLID	$\bar{R}_{DLI}^D$	
VDLID	$\bar{V}_{DLI}^D$	
ANGV	ANG <sub>V</sub>	Angle at vehicle between LOS to sun and earth
ANGE	ANG <sub>E</sub>	Angle at earth between LOS to sun and vehicle
ANGS	ANG <sub>S</sub>	Angle at sun between LOS to vehicle and earth

C. Computation

The position and velocity of 4-body L<sub>1</sub> point in S-frame are computed by

$$\bar{R}_{SL1} = (1 - \gamma_{L1}) \left( \bar{R}_{SE} + \frac{\mu_M}{\mu_E + \mu_M} \bar{R}_{EM} \right)$$

$$\bar{V}_{SL1} = (1 - \gamma_{L1}) \left( \bar{V}_{SE} + \frac{\mu_M}{\mu_E + \mu_M} \bar{V}_{EM} \right)$$

An earth centered rotating frame (D-frame) for display is used to suppress the motion of the earth which is significant due to its orbital eccentricity around the sun and the presence of the moon.

Define:

$$\bar{u}X_D^S = \text{unit}(\bar{R}_{SE})$$

$$\bar{u}Z_D^S = \text{unit}(\bar{R}_{SE} \times \bar{V}_{SE})$$

$$\bar{u}Y_D^S = \bar{u}Z_D^S \times \bar{u}X_D^S$$

The transformation from the inertial S-frame to a frame parallel to the rotating D-frame is given by the matrix

$$C_S^D = \left( (\bar{u}X_D^S)^T, (\bar{u}Y_D^S)^T, (\bar{u}Z_D^S)^T \right)$$

The D-frame rotates with respect to S-frame with an angular velocity defined by



$$\bar{\omega}_{SD}^S = \bar{u}_{Z_D}^S (\bar{u}_{Y_{DS}}^S \cdot \bar{v}_{SE}) / |\bar{R}_{SE}|$$

The transforming from S-frame to D-frame for typical position and velocity vectors are as follows:

$$\bar{R}_{SD}^S = \bar{R}_{SE}^S$$

$$\bar{v}_{SD}^S = \bar{v}_{SE}^S$$

$$\bar{R}_{DV}^D = C_S^D \bar{R}_{DV}^S = C_S^D (\bar{R}_{SV}^S - \bar{R}_{SD}^S)$$

$$\bar{v}_{DV}^D = C_S^D (\bar{v}_{SV}^S - \bar{v}_{SD}^S - \bar{\omega}_{SD}^S \times \bar{R}_{DV}^S)$$

The angles at the vehicle, sun and earth are computed by

$$\bar{u}_{R_{SE}} = \text{UNIT}(\bar{R}_{SE})$$

$$\bar{u}_{R_{EV}} = \text{UNIT}(\bar{R}_{EV})$$

$$\bar{u}_{R_{SV}} = \text{UNIT}(\bar{R}_{SV})$$

$$\text{ANG}_V = \cos^{-1}(\bar{u}_{R_{EV}} \cdot \bar{u}_{R_{SV}}) / \text{DTR}$$

$$\text{ANG}_E = \cos^{-1}(-\bar{u}_{R_{SE}} \cdot \bar{u}_{R_{EV}}) / \text{DTR}$$

$$\text{ANG}_S = 180 - (\text{ANG}_V + \text{ANG}_E)$$

$$\text{DTR} = \text{Degree to radian conversion}$$

Subroutine DISP3

Both input and output parameters are transmitted through COMMON/TDATA3/.

A. Input Parameters

<u>Parameters</u>	<u>Symbol</u>	<u>Definition</u>
REV	$\bar{R}_{EV}^e$	
VEV	$\bar{V}_{EV}^e$	
RMV	$\bar{R}_{MV}^m$	
VMV	$\bar{V}_{MV}^m$	
REM	$\bar{R}_{EM}^e$	
VEM	$\bar{V}_{EM}^e$	

B. Output Parameters

REL	$\bar{R}_{EL}^e$
VEL	$\bar{V}_{EL}^e$
RDVD	$\bar{R}_{DV}^d$
VDVD	$\bar{V}_{DV}^d$
RDED	$\bar{R}_{DE}^d$
VDED	$\bar{V}_{DE}^d$
RDLD	$\bar{R}_{DL}^D$
VDLD	$\bar{V}_{DL}^D$
ANGV	$ANG_V$
ANGE	$ANG_E$
ANGM	$ANG_M$

C. Computation

The position and velocity of 3-body libration point L in e-frame are computed by

$$\bar{R}_{EL} = (1 - \gamma) \bar{R}_{EM}$$

$$\bar{V}_{EL} = (1 - \gamma) \bar{V}_{EM}$$

where  $\gamma > 0$  for  $L_1$  and  $\gamma < 0$  for  $L_2$ .

A moon centered rotating frame (d-frame) for display is used to suppress the motion of the moon. Define

$$\bar{u}_{XD}^e = \text{UNIT}(\bar{R}_{EM}^e)$$

$$\bar{u}_{ZD}^e = \text{unit}(\bar{R}_{EM}^e \times \bar{V}_{EM}^e)$$

$$\bar{u}_{YD}^e = \bar{u}_{ZD}^e \times \bar{u}_{XD}^e$$

The transformation from the inertial e-frame to a frame parallel to the rotating d-frame is given by the matrix

$$C_e^d = ((\bar{u}_{XD}^e)^T, (\bar{u}_{YD}^e)^T, (\bar{u}_{ZD}^e)^T)$$

The d-frame rotates with respect to the e-frame with an angular velocity

$$\bar{\omega}_{ed}^e = \bar{u}_{ZD}^e \left( \bar{u}_{YD}^e \cdot \frac{\bar{V}_{EM}^e}{|\bar{R}_{EM}^e|} \right)$$

The transformations from the e-frame to d-frame for typical position and velocity vectors are as follows:

$$\bar{R}_{ED}^e = \bar{R}_{EM}^e$$

$$\bar{V}_{ED}^e = \bar{V}_{EM}^e$$

$$\bar{R}_{DV}^d = C_e^d \bar{R}_{DV}^e = C_e^d (\bar{R}_{EV}^e - \bar{R}_{ED}^e)$$

$$\bar{V}_{DV}^d = C_e^d (\bar{V}_{EV}^e - \bar{V}_{ED}^e - \bar{\omega}_{ed}^e \times \bar{R}_{DV}^e)$$

The angles at the vehicle, earth and moon are computed by:

$$\bar{u}R_{MV} = \text{UNIT}(\bar{R}_{MV})$$

$$\bar{u}R_{EM} = \text{UNIT}(\bar{R}_{EM})$$

$$\bar{u}R_{EV} = \text{UNIT}(\bar{R}_{EV})$$

$$\text{ANG}_V = \cos^{-1}(\bar{u}R_{MV} \cdot \bar{u}R_{EV}) / \text{DTR}$$

$$\text{ANG}_M = \cos^{-1}(-\bar{u}R_{EM} \cdot \bar{u}R_{MV}) / \text{DTR}$$

$$\text{ANG}_E = 180 - (\text{ANG}_V + \text{ANG}_M)$$

$$\text{DTR} = \text{Degree to radian conversion}$$

Subroutine PTRAJ

Input parameters are transmitted through COMMON/TDATA/. If  
IPTRAJ = 1, the following parameters are printed at each time step.

T	T(DAY)	H	$ \bar{R}_{SV} $	$ \bar{R}_{EV} $	$ \bar{R}_{MV} $
$\bar{R}_{SV}^S$			$\bar{V}_{SV}^S$		
$\bar{R}_{EV}^E$			$\bar{V}_{EV}^E$		
$\bar{R}_{MV}^M$			$\bar{V}_{MV}^M$		
$\bar{R}_{SE}^S$			$\bar{V}_{SE}^S$		
$\bar{R}_{SM}^S$			$\bar{V}_{SM}^S$		
$\bar{R}_{EM}^E$			$\bar{V}_{EM}^E$		
$\bar{R}_{SL1}^S$			$\bar{V}_{SL1}^S$		
$\bar{R}_{DV}^D$			$\bar{V}_{DV}^D$		
$\bar{R}_{DS}^D$			$\bar{V}_{DS}^D$		
$\bar{R}_{DM}^D$			$\bar{V}_{DM}^D$		
$\bar{R}_{DL1}^D$			$\bar{V}_{DL1}^D$		
ANG <sub>V</sub>	ANG <sub>E</sub>	ANG <sub>S</sub>	H(DAY)		

If IPV = 1, the primer vector history below is printed.

$\bar{\lambda}$		$\dot{\bar{\lambda}}$
$ \bar{\lambda} $	$\frac{d}{dt}  \bar{\lambda} $	

Subroutine PTRAJ3

Input parameters are transmitted through COMMON/TDATA3/. If  
 IPTRAJ = 1, the following parameters are printed at each time step.

T	TDAY	H	H(DAY)	$ \bar{R}_{EV} $	$ \bar{R}_{MV} $
$\bar{R}_{EV}^e$			$\bar{V}_{EV}^e$		
$\bar{R}_{MV}^m$			$\bar{V}_{MV}^m$		
$\bar{R}_{EM}^e$			$\bar{V}_{EM}^e$		
$\bar{R}_{EL}^e$			$\bar{V}_{EL}^e$		
$\bar{R}_{DV}^d$			$\bar{V}_{DV}^d$		
$\bar{R}_{DE}^d$			$\bar{V}_{DE}^d$		
$\bar{R}_{DL}^d$			$\bar{V}_{DL}^d$		
ANG <sub>V</sub>	ANG <sub>E</sub>	ANG <sub>M</sub>			

If IPV = 1, the primer vector history is printed

$\bar{\lambda}$		$\dot{\bar{\lambda}}$
$ \bar{\lambda} $	$\frac{d}{dt}  \bar{\lambda} $	

Subroutine FDATA

Input parameters are transmitted through COMMON/TDATA/. If IFILEX or IFILE is position nonzero the following double precision parameters are filed sequentially in unformatted form.

<u>Parameter</u>	<u>File Location</u>	<u>Parameter</u>	<u>File Location</u>
t	+1	$\bar{V}_{DS}^D$	54 - 56
h	2	$\bar{R}_{DM}^D$	57 - 59
$\bar{R}_{SV}$	3 - 5	$\bar{V}_{DM}^D$	60 - 62
$\bar{V}_{SV}$	6 - 8	$\bar{R}_{DL1}^D$	63 - 65
$\bar{R}_{EV}$	9 - 11	$\bar{V}_{DL1}^D$	66 - 68
$\bar{V}_{EV}$	12 - 14	$\bar{\lambda}, \dot{\bar{\lambda}}$	69 - 74
$\bar{R}_{MV}$	15 - 17	$ \bar{\lambda} $	75
$\bar{V}_{MV}$	18 - 20	$\frac{d}{dt}  \bar{\lambda} $	76
$\bar{R}_{SE}$	21 - 23	ANG <sub>V</sub>	77
$\bar{V}_{SE}$	24 - 26	ANG <sub>E</sub>	78
$\bar{R}_{SM}$	27 - 29	ANG <sub>S</sub>	79
$\bar{V}_{SM}$	30 - 32	$ \bar{R}_{SV} $	80
$\bar{R}_{EM}$	33 - 35	$ \bar{V}_{SV} $	81
$\bar{V}_{EM}$	36 - 38	$ \bar{R}_{EV} $	82
$\bar{R}_{SL1}$	39 - 41	$ \bar{V}_{EV} $	83
$\bar{V}_{SL1}$	42 - 44	$ \bar{R}_{MV} $	84
$\bar{R}_{DV}^D$	45 - 47	$ \bar{V}_{MV} $	85
$\bar{V}_{DV}^D$	48 - 50	Blank	86 - 100
$\bar{R}_{DS}^D$	51 - 53	Total	100

Subroutine FDATA3

Input parameters are transmitted through COMMON/TDATA3/. If IFILEX or IFILE is positive nonzero, the following double precision parameters are filed sequentially in unformatted form.

<u>Parameter</u>	<u>File Location</u>	<u>Parameter</u>	<u>File Location</u>
t	+ 1	$\bar{V}_{DL}^d$	42 - 44
h	2	$\bar{\lambda}, \dot{\lambda}$	45 - 50
$\bar{R}_{EV}$	3 - 5	$ \bar{\lambda} $	51
$\bar{V}_{EV}$	6 - 8	$\frac{d}{dt}  \bar{\lambda} $	52
$\bar{R}_{MV}$	9 - 11	ANG <sub>V</sub>	53
$\bar{V}_{MV}$	12 - 14	ANG <sub>E</sub>	54
$\bar{R}_{EM}$	15 - 17	ANG <sub>M</sub>	55
$\bar{V}_{EM}$	18 - 20	$ \bar{R}_{EV} $	56
$\bar{R}_{EL}$	21 - 23	$ \bar{V}_{EV} $	57
$\bar{V}_{EL}$	24 - 26	$ \bar{R}_{MV} $	58
$\bar{R}_{DV}^d$	27 - 29	$ \bar{V}_{MV} $	59
$\bar{V}_{DV}^d$	30 - 32	$ \bar{R}_{EM} $	60
$\bar{R}_{DE}^d$	33 - 35	$ \bar{V}_{EM} $	61
$\bar{V}_{DE}^d$	36 - 38	Blank	62 - 75
$\bar{R}_{DL}^d$	39 - 41	Total	75



## TROUBLE SHOOTING

In this section we list a number of common troubles, their causes and suggested corrective actions.

<u>Trouble</u>	<u>Diagnosis</u>	<u>Action</u>
Integration too slow.	Allowable single step position error too small. Input error.	Increase ERRMAX or ERRMXM. Check input.
TWOBDY fails to converge or gives strange answers.	Bad input state vector or gravitational constants.	Check input.
FOURBY (THRBDY) gives strange answers.	Bad input to TWOBDY.	Check input.
Slow to satisfy constraint.	Allowable terminal error too small.	Increase ERRMIN or EPSTSI.
LAMB (LAMB3) fails to converge.	Bad nominal trajectory.	Search for new nominal closer to satisfy constraint. Reduce KNR.
ETP21 (ETP213) fails to satisfy constraint.	Bad nominal trajectory.	Search for new nominal closer to satisfy constraint. Reduce KDX.
ETP21 (ETP213) fails to reduce $\Delta V$ .	Bad V-matrix.	Reduce EPSV. Try new estimate of independent variables.
P'P'P31 (P'P'P313) fails to reduce $\Delta V$ .	Bad V-matrix.	Reduce EPSV. Try new estimate of independent variables.

### Helpful Suggestions

1. It pays to search for a nominal trajectory which comes close to satisfy terminal constraint. This is not so important in a short transfer, which will converge even if the nominal trajectory misses the target badly. For long transfer, a bad nominal trajectory will prolong iterations due to increased nonlinearity, which reduces permissible changes in the independent variables.

2. A capability to plot trajectory is very helpful to show whether a nominal trajectory is heading in the right direction. It is not difficult to adjust the independent variables properly after examining a plot of the trajectory but it is very difficult to determine what action to take by looking at a printout.
3. It is generally better to run the programs by a succession of short runs instead of a single long run. The pauses between short runs enable the user to interpret the results and make necessary changes. It is not possible nor practical to fully automate a program to take into account all possible contingencies.
4. When a run is continued, use the current independent variables, V-matrix, KDX, KNR, etc. to maintain continuity.

## EXAMPLE 1

### A. Purpose

This example may be used to check out Program TRAJ and associated sub-routines. It is a 4-body transfer from noon July 4, 1978 from a 100 n. mi. earth parking orbit to  $L_1$  in 36 days.

### B. Input Parameters

<u>Card No.</u>	<u>Format</u>	<u>Parameter</u>	<u>Value</u>
1	4D20.11	GLI	1.001098D-02
		AUM	1.49597893D+11
		UTIME	5.81323577632D+01
		UVELM	3.35742409867D-05
2	3D20.11	MS	9.99996959568D-01
		ME	3.00348453188D-06
		MM	3.60431224671D-08
3	4I5	IMODE	1
		IPV	1
		IPTRAJ	1
		IFILE	0
4	4D20.11	RSE0(1)	1.98930090999D-01
		(2)	-9.97050580850D-01
		(3)	6.79027012471D-05
		VSE0(1)	9.64906589348D-01
		(2)	1.92157060597D-01
5	4D20.11	(3)	-1.47749626270D-05

6	4D20.11	RSM0 (1)	1.99125783494D-01
		(2)	-9.94358817445D-01
		(3)	-1.68002571823D-04
		VSM0 (1)	9.32348120275D-01
		(2)	1.95028878329D-01
		(3)	-2.62545160637D-04
7	3D20.11	TDAY0	1.099D+03
		TTRIPD	3.6D+01
		ERRMXM	1.524D+02
8	4D20.11	REVMAG	4.38739157152D-05
		VEVMAG	3.69220795569D-01
		OINCD	2.85D+01
		OBLD	2.34457874302D+01
9	2D20.11	LOND	3.59181042981D+02
		THED	-3.51880454048D+01
10	4D20.11	λ(1)	5.86468563221D-01
		(2)	8.06463656568D-01
		(3)	7.53060089751D-02
		λ(1)	-3.43647891554D+03
11	2D20.11	(2)	2.47056772744D+03
		(3)	3.04177095939D+02

C. Output Parameters

A printout of the trajectory at each time step is submitted with this guide in a separate package. The state vectors at the terminal time is

RSV(1)	7.17957115972D-01
(2)	-7.01342803746D-01
(3)	4.27257573439D-05
VSV(1)	6.79829811710D-01
(2)	7.15967885505D-01
(3)	-6.21512714224D-04

The vehicle intercepts the  $I_1$  point after 36 days. The trajectories with respect to earth and moon are shown in Figures 2a to 2f. The primer vector magnitude history is shown in Figure 2g. The  $\Delta V$  of this transfer is 354.57 m/s. Note that the initial earth parking orbit has an inclination of 28.5 deg. If this restriction is removed, the  $\Delta V$  of the same transfer of 36 days is about 288 m/s (Ref. 2).

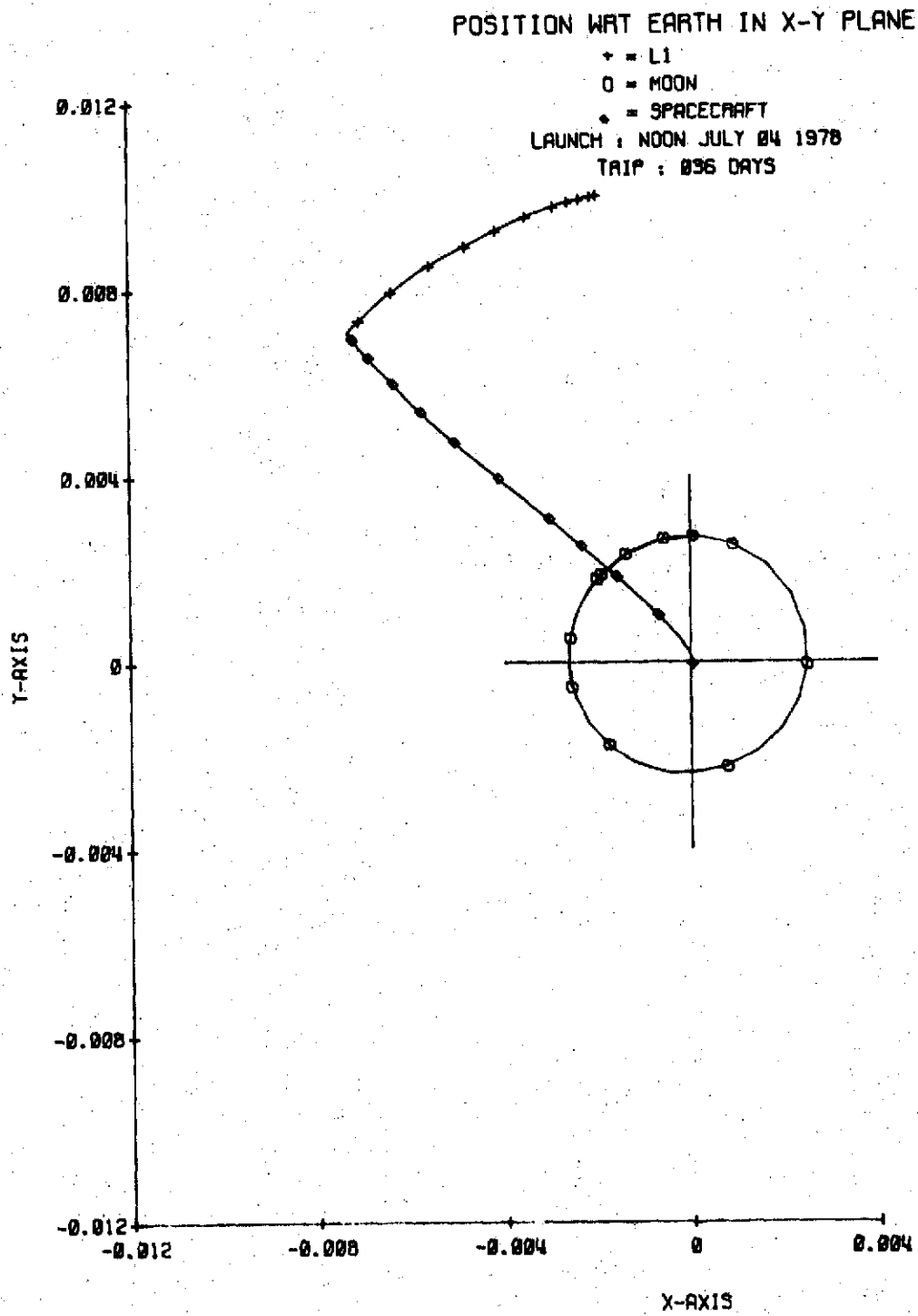


Figure 2a 2-Impulse Transfer (Example 1)

POSITION WRT MOON IN X-Y PLANE

X = EARTH

• = SPACECRAFT

LAUNCH : NOON JULY 04 1978

TRIP : 036 DAYS

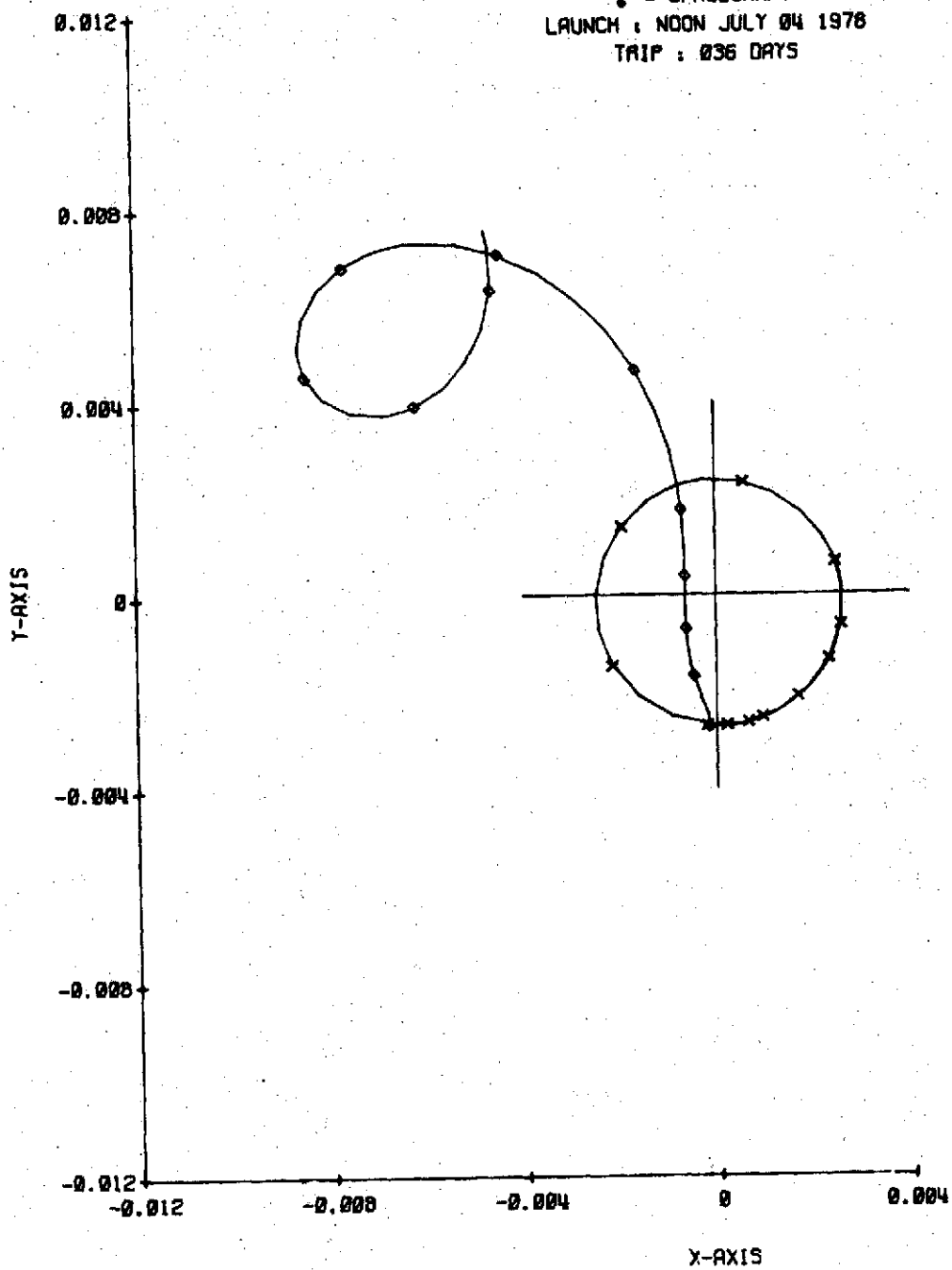
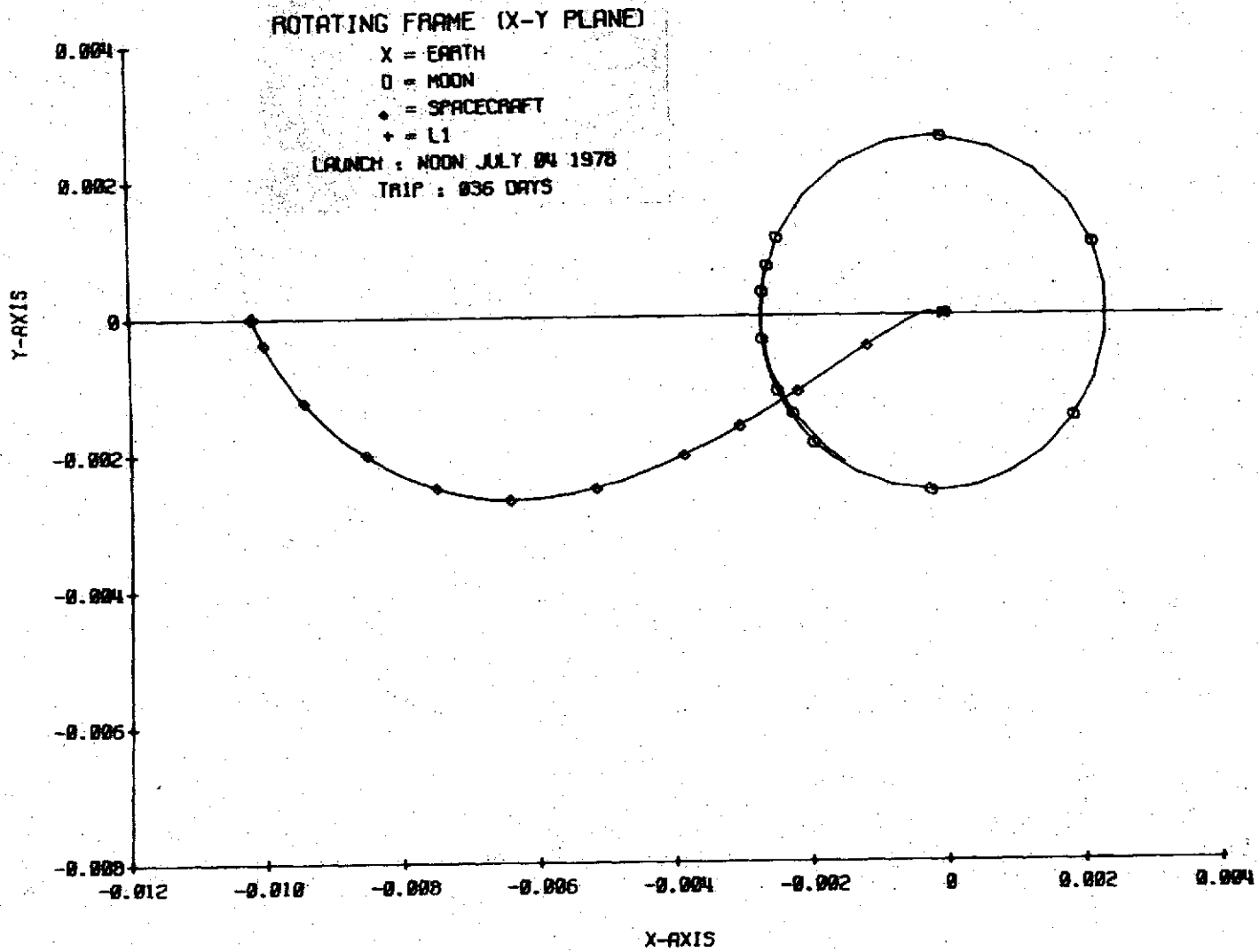


Figure 2b 2-impulse Transfer (Example 1)



101

Figure 2c 2-Impulse Transfer (Example 1)



106

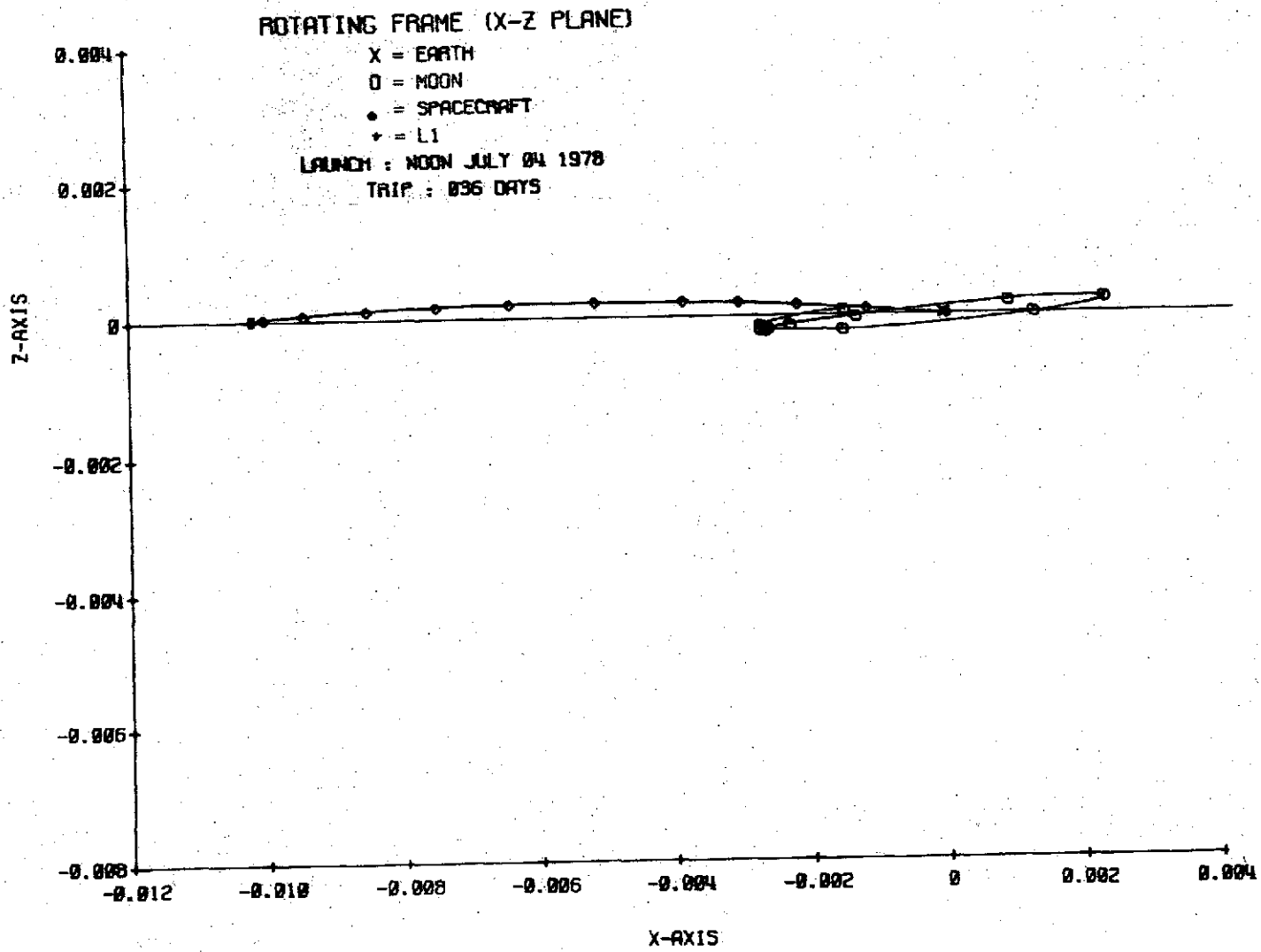


Figure 2d 2-Impulse Transfer (Example 1)

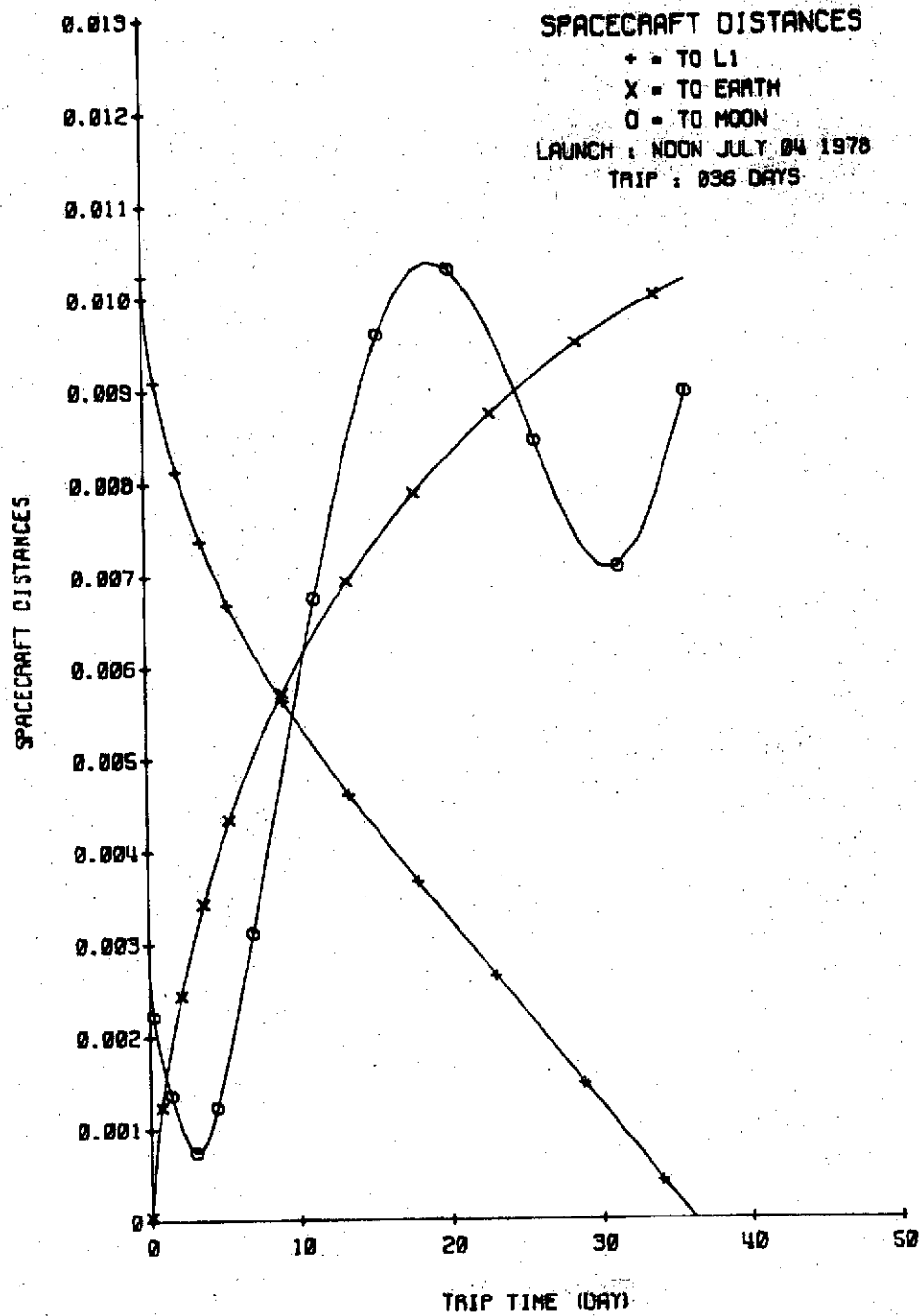


Figure 2e 2-Impulse Transfer (Example 1)

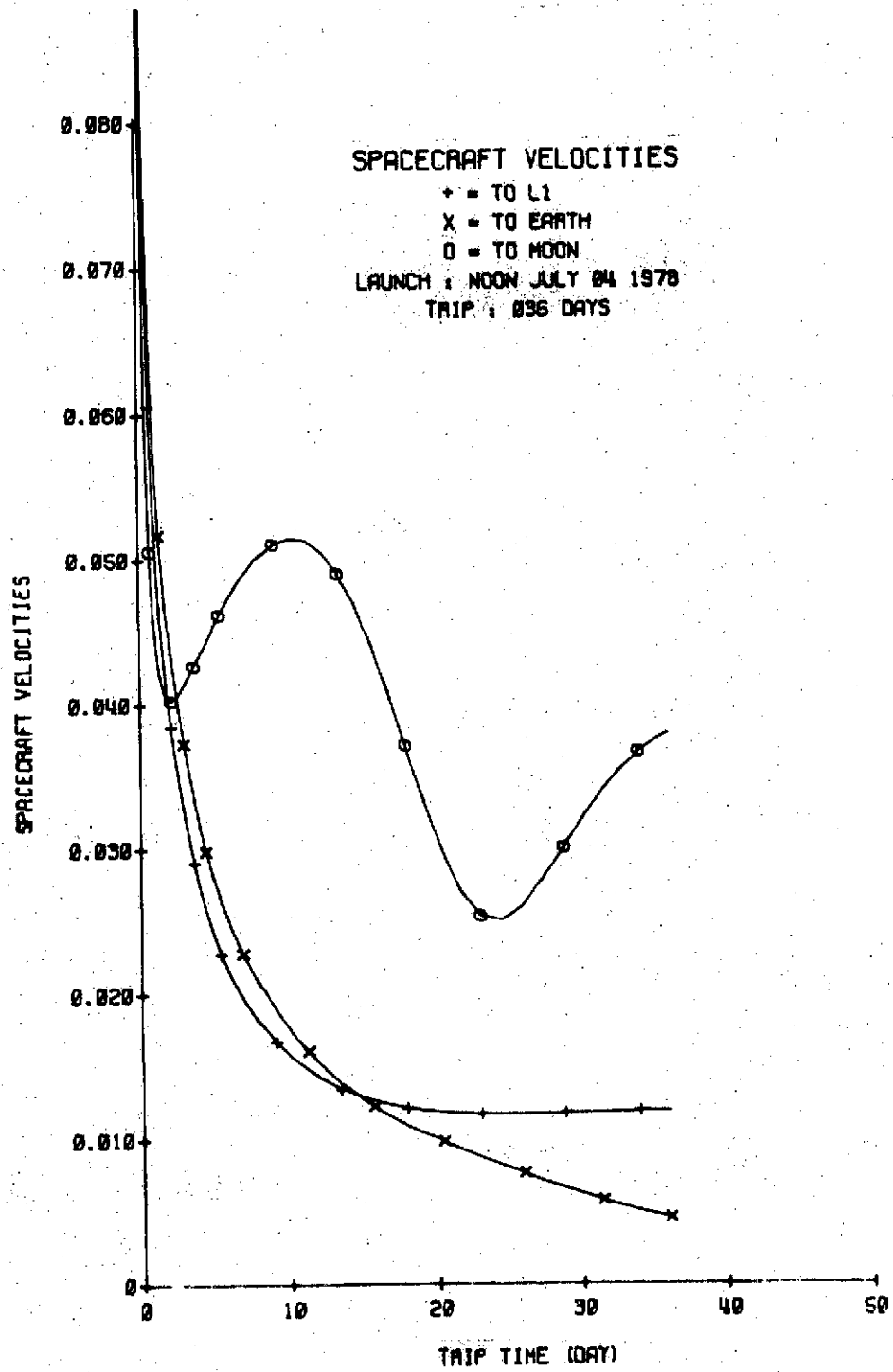


Figure 2f 2-Impulse Transfer (Example 1)

PRIME VECTOR MAGNITUDE

LAUNCH : NOON JULY 04 1978

TRIP : 036 DAYS

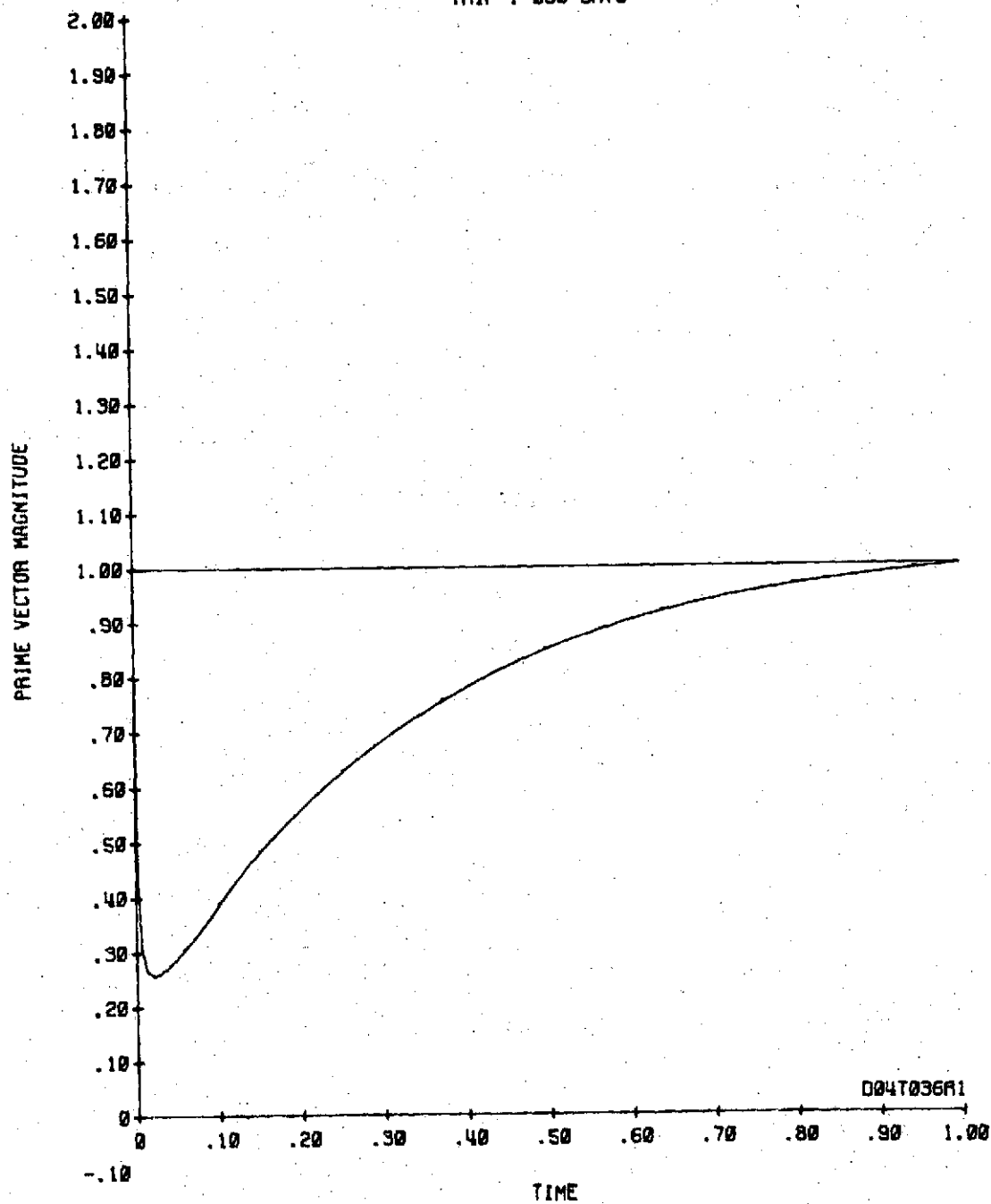


Figure 2g 2-Impulse Transfer (Example 1)

EXAMPLE 2

A. Purpose

This example may be used to check out Program EXLAM and associated subroutines. It is a 4-body transfer from noon July 4, 1978 from a given position at earth to a given point in space in 36 days.

B. Input Parameters

\* Same as in Ex. 1

<u>Card No.</u>	<u>Format</u>	<u>Parameter</u>	<u>Value</u>
1	4D20.11	GLI	*
		AUM	*
		UTIME	*
		UVELM	*
2	3D20.11	MS	*
		ME	*
		MM	*
3	4I5	IMODE	2
		ITLMAX	50
		IPTJX	0
		IFILEX	0
4	4D20.11	RSE0 (1)	1.98930091D-01
		(2)	-9.97050589850D-01
		(3)	6.79027012471D-05
		VSE0 (1)	9.64906589348D-01
		(2)	1.92157060597D-01
5	4D20.11	(3)	-1.47749626270D-05

		RSM0 (1)	1. 99125783494D-01
		(2)	-9. 94358817445D-01
6	4D20.11	(3)	-1. 68002571823D-04
		VSM0 (1)	9. 32348120275D-01
		(2)	1. 95028878329D-01
		(3)	-2. 62545160637D-04
7	3D20.11	TDAY0	1. 099D+03
		TRIPD	3. 6D+01
		ERRMXM	1. 524D+02
8	4D20.11	RSVTAR	7. 17957115972D-01
			-7. 01342803746D-01
			4. 27257573439D-05
		ERRMIN	1. 0D-10
9	1D20.11	KNR	1. 0D+00
10	4D20.11	REV0 (1)	3. 55098760149D-05
		(2)	-2. 56858251705D-05
		(3)	-2. 05123643240D-06
		VEV0P(1)	2. 16833393469D-01
11	2D20.11	(2)	2. 97557757082D-01
		(3)	2. 76469306727D-02

C. Output Parameters

A printout of Lambert iterations is submitted with this guide in a separate package. The initial state vector is:

RSV0 (1)	1.98965600875D-01
(2)	-9.97076275675D-01
(3)	6.58514648147D-05
VSVI(1)	1.18173998282D+00
(2)	4.89714817679D-01
(3)	2.76321557101D-02

The iteration history is as follows:

<u>ITER</u>	<u>Error (<math> \psi </math>)</u>
1	3.9725 D-05 initial miss
2	1.6349D-04
3	3.4139D-05
4	2.0172D-05
5	1.7986D-05
6	1.6351D-05
7	9.6429D-09
8	6.1105D-12 < ERRMIN

The converged solution is

VSVI(1)	1.18164961297D+00
(2)	4.89491098961D-01
(3)	3.06148929077D-02

### EXAMPLE 3

A. Purpose

This example may be used to check out Program ETP2I and associated subroutines. It is a 4-body transfer from noon July 4, 1978 to  $L_1$  in 44 days. The earth parking orbit has an inclination of 28.5 deg.

B. Output Parameters

\* Same as in Ex. 1

\*\* Same as in Ex. 2

<u>Card No.</u>	<u>Format</u>	<u>Parameter</u>	<u>Value</u>
1	4D20.11	GL1	*
		AUM	*
		UTIME	*
		UVELM	*
2	3D20.11	MS	*
		ME	*
		MM	*
3	4D20.11	REVMAG	*
		OINCD	*
		OBLD	*
		ERRMXM	*
4	2D20.11	TDAY0	*
		TTRIPD	4.40D+01
5	3D20.11	VEVMAG	3.69195894918D-01
		LOND	3.40D+02
		THED	0.1D+00



6	4D20.11	RSE0(1)	*
		(2)	*
		(3)	*
		VSE0(1)	*
7	4D20.11	(2)	*
		(3)	*
		RSM0(1)	*
		(2)	*
8	4D20.11	(3)	*
		VSM0(1)	*
		(2)	*
		(3)	*
9	4D20.11	EPS	1.0D-11
		EPSTSI	1.0D-10
		KDX	1.0D+00
		EPSV	1.0D-03
10	14	ICOMV	1
		ITERMX	50
		IFILEX	0
		ITAR	0
11	3D20.11	AYM	0
		AZM	0
		ATARD	0

C. Output Parameters

The state vector at the terminal time is:

RSV(1)	8.04147100281D-01
(2)	-5.98272665021D=01
(3)	3.54443603861D-05
VSV(1)	5.80886320061D-01
(2)	8.02081814322D-01
(3)	-4.43809910137D-04

The converged initial conditions are:

VEVMAG	10997.18 m/s
LOND	358.32 deg
THED	-23.43 deg

The costs are:

$ \Delta V_o $	3204.20 m/s
$ \Delta V_f $	$\frac{377.80}{3582.00}$ m/s

Ref. 2 indicates that  $|\Delta V_f|$  is 263.31 m/s for the 44 day transfer without the inclination angle constraint.

A printout of the iterations together with the trajectory at each time step is submitted in a separate package.

The trajectory is shown in Figure 3a to 3f. The primer vector history is shown in Figure 3g.

POSITION WRT EARTH IN X-Y PLANE

+ = L1

o = MOON

• = SPACECRAFT

LAUNCH : NOON JULY 04 1978

TRIP : 044 DAYS

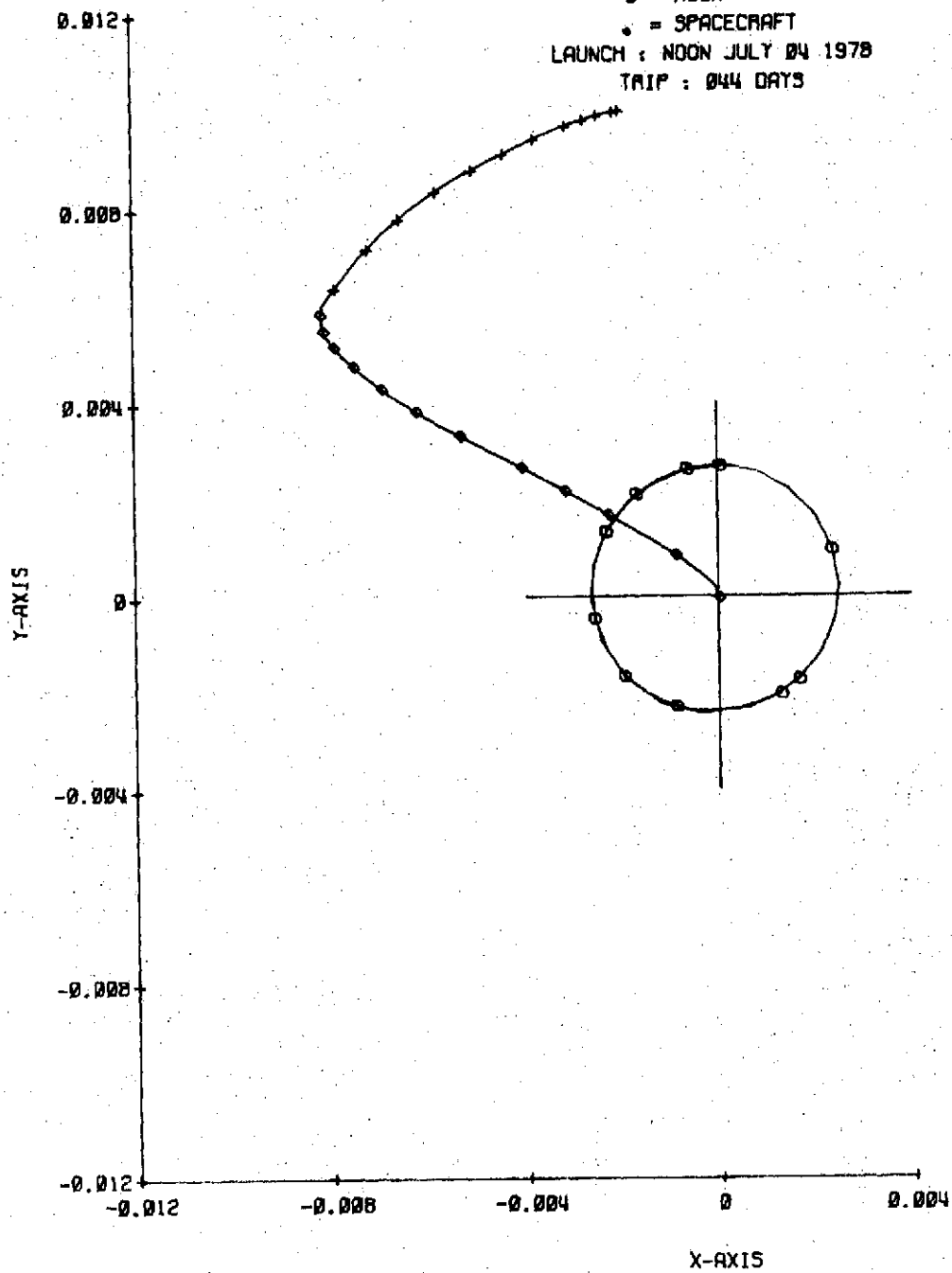


Figure 3a 2-Impulse Transfer (Example 3)

POSITION WRT MOON IN X-Y PLANE

X = EARTH

• = SPACECRAFT

LAUNCH : NOON JULY 04 1978

TRIP : 044 DAYS

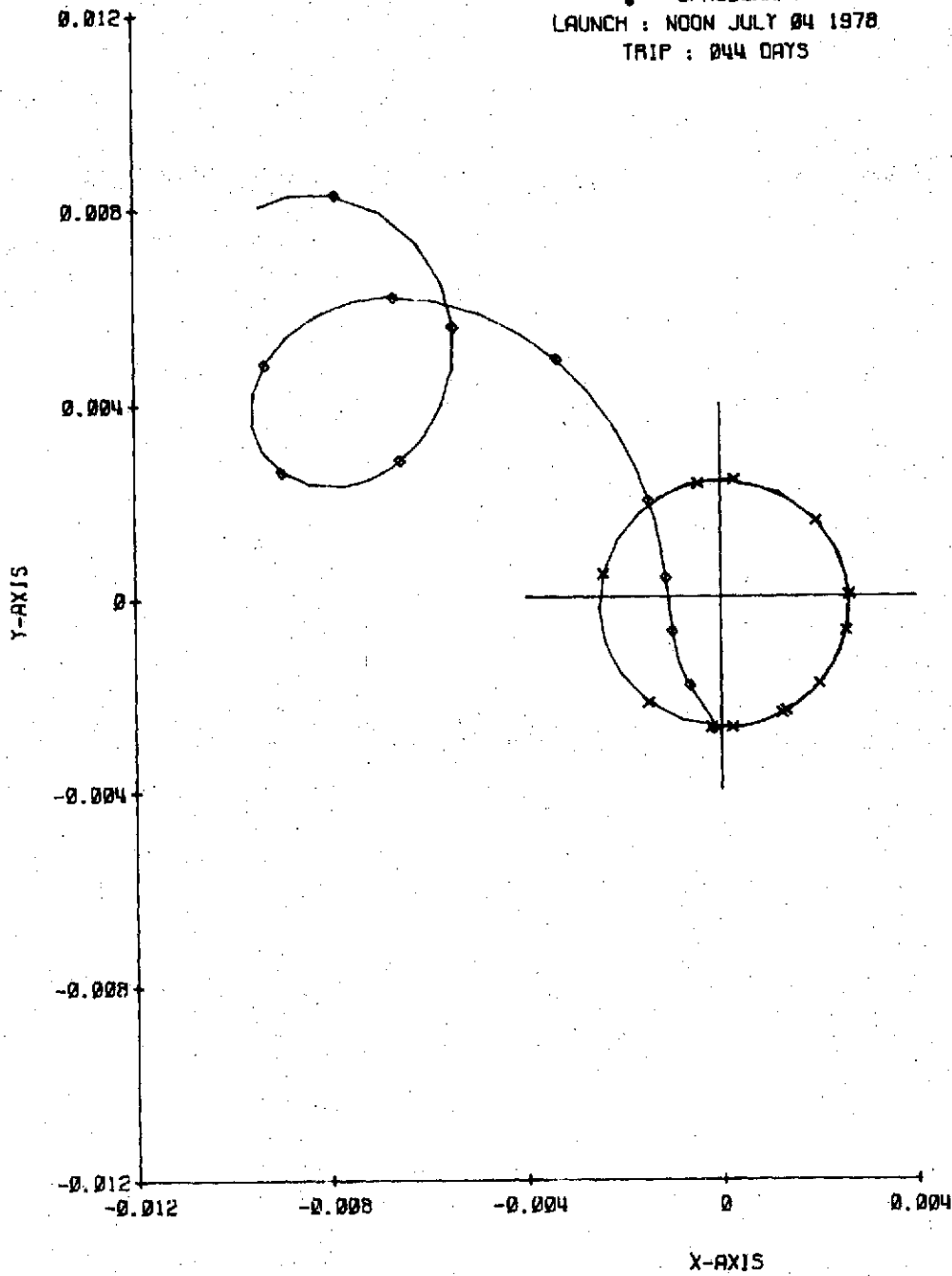


Figure 3b 2-Impulse Transfer (Example 3)

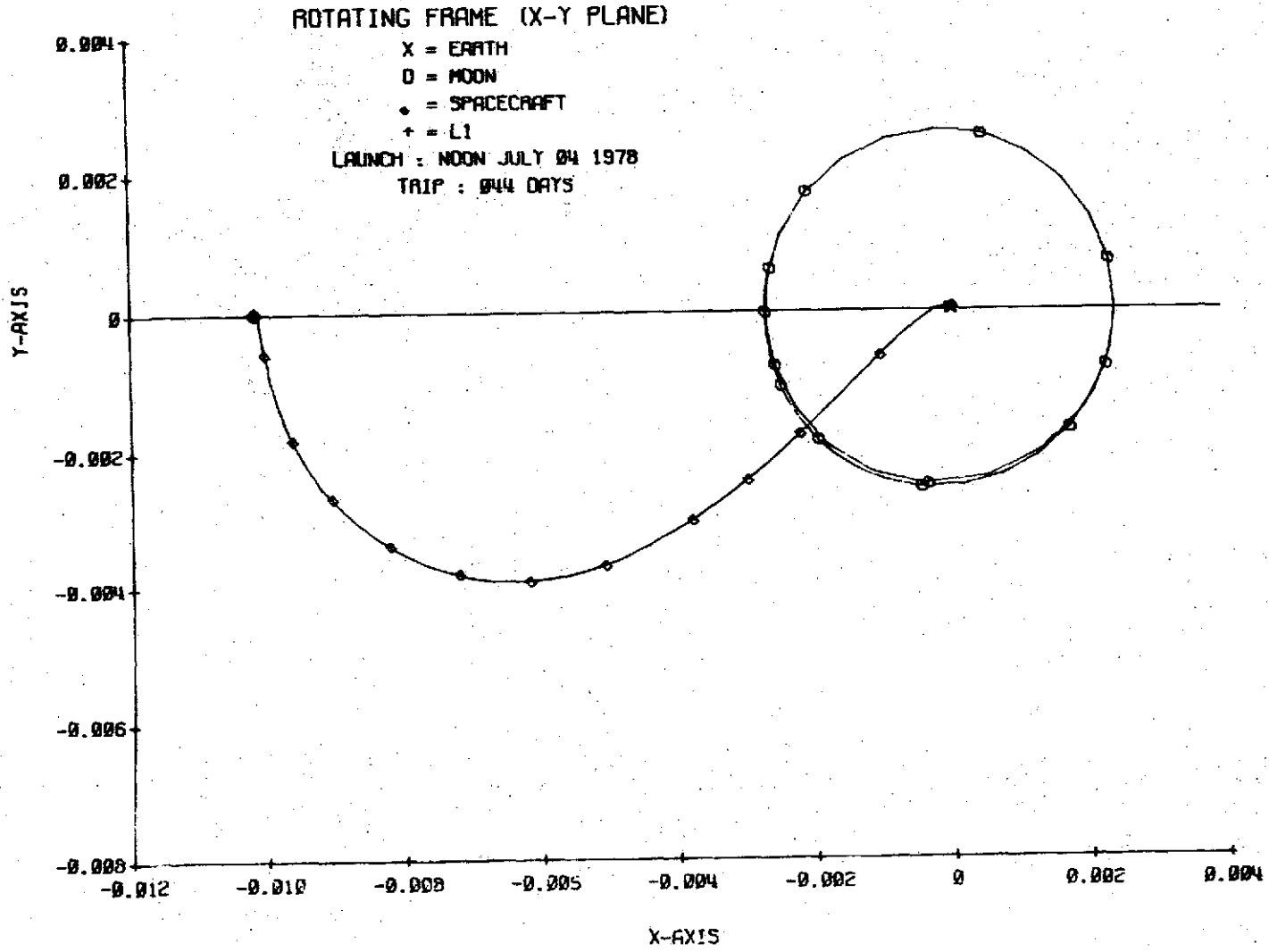
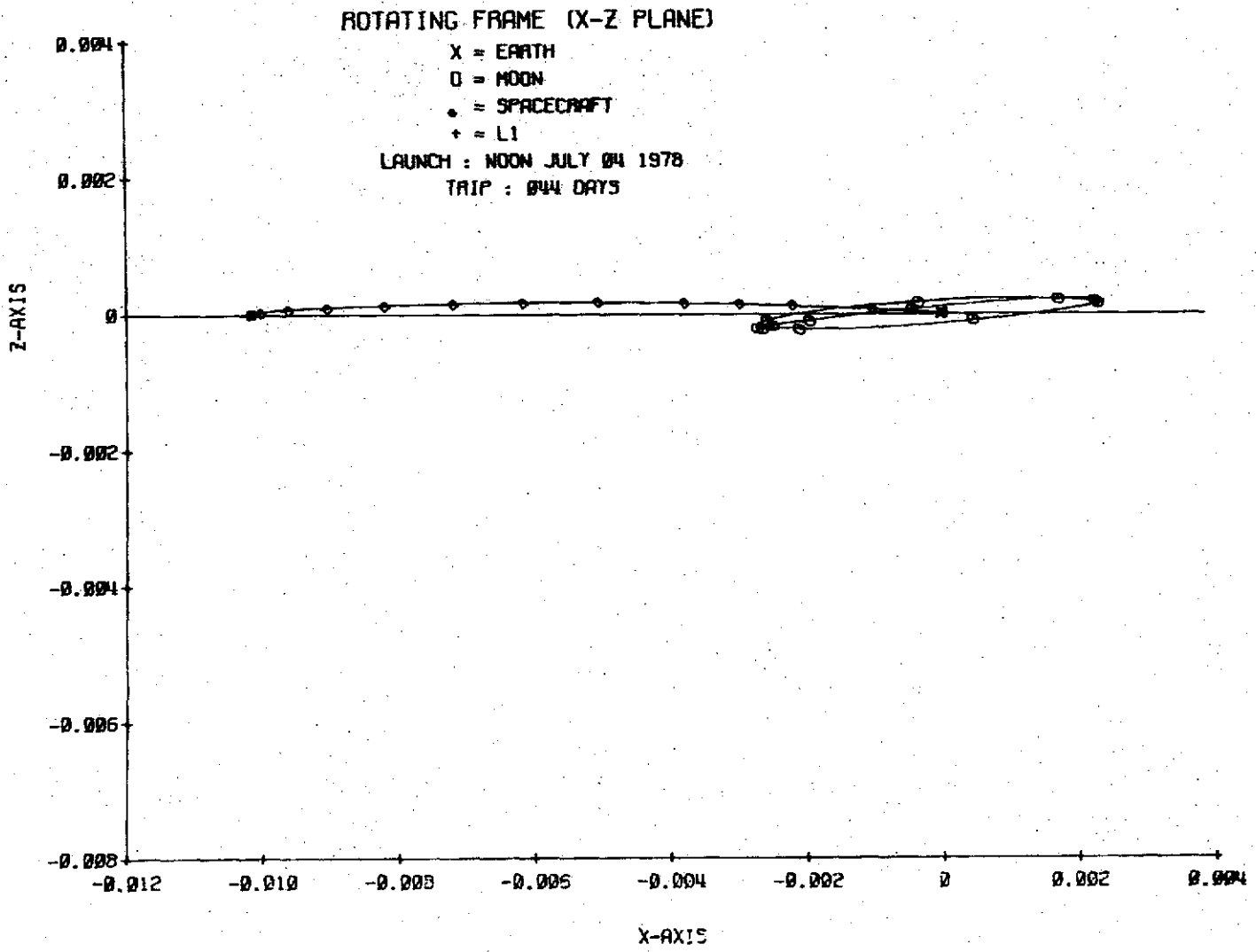


Figure 3c 2-Impulse Transfer (Example 3)



119

Figure 3d 2-Impulse Transfer (Example 3)

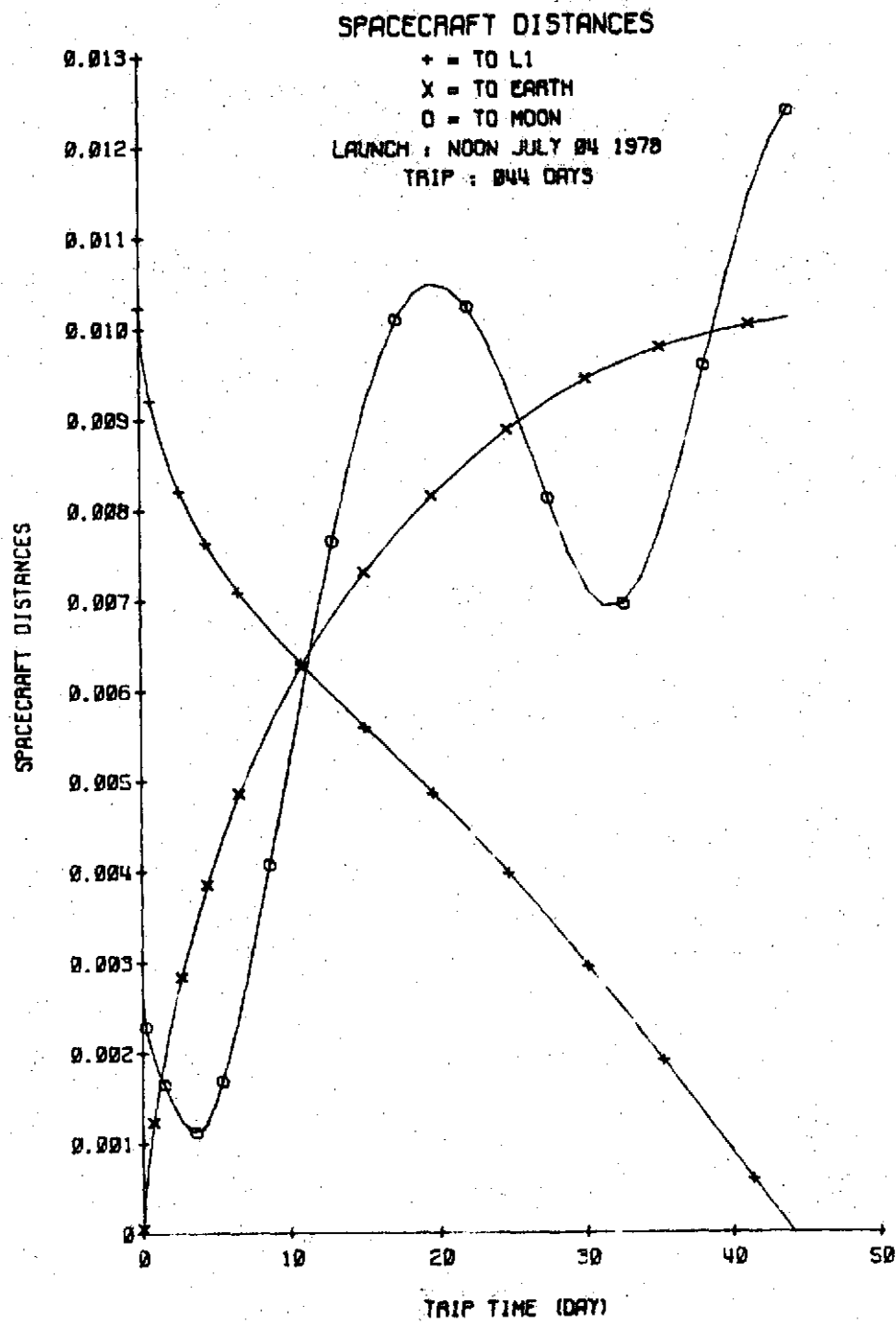


Figure 3e 2-Impulse Transfer (Example 3)

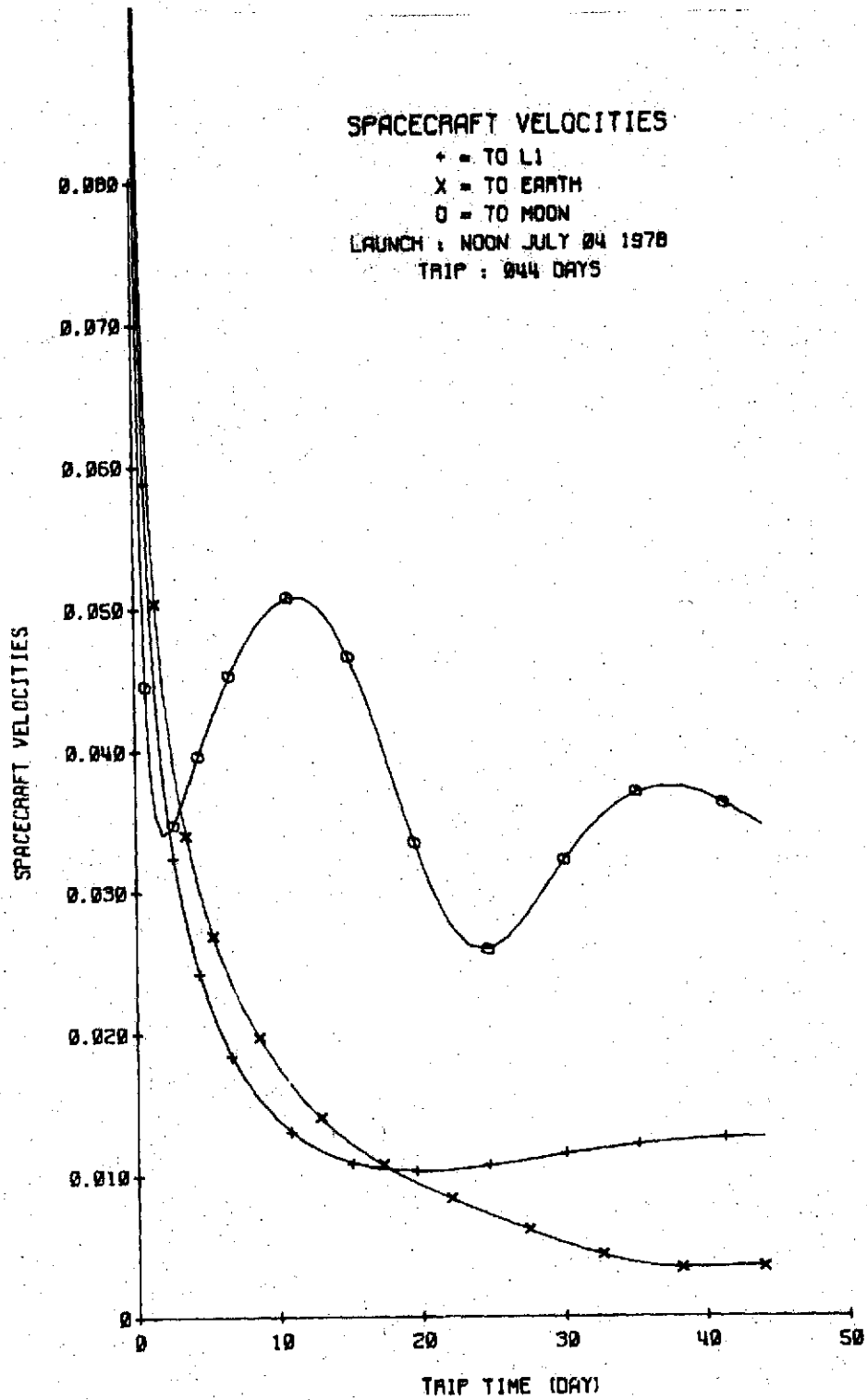


Figure 3f 2-Impulse Transfer (Example 3)



PRIMER VECTOR MAGNITUDE

LAUNCH : NOON JULY 04 1978

TRIP : 044 DAYS

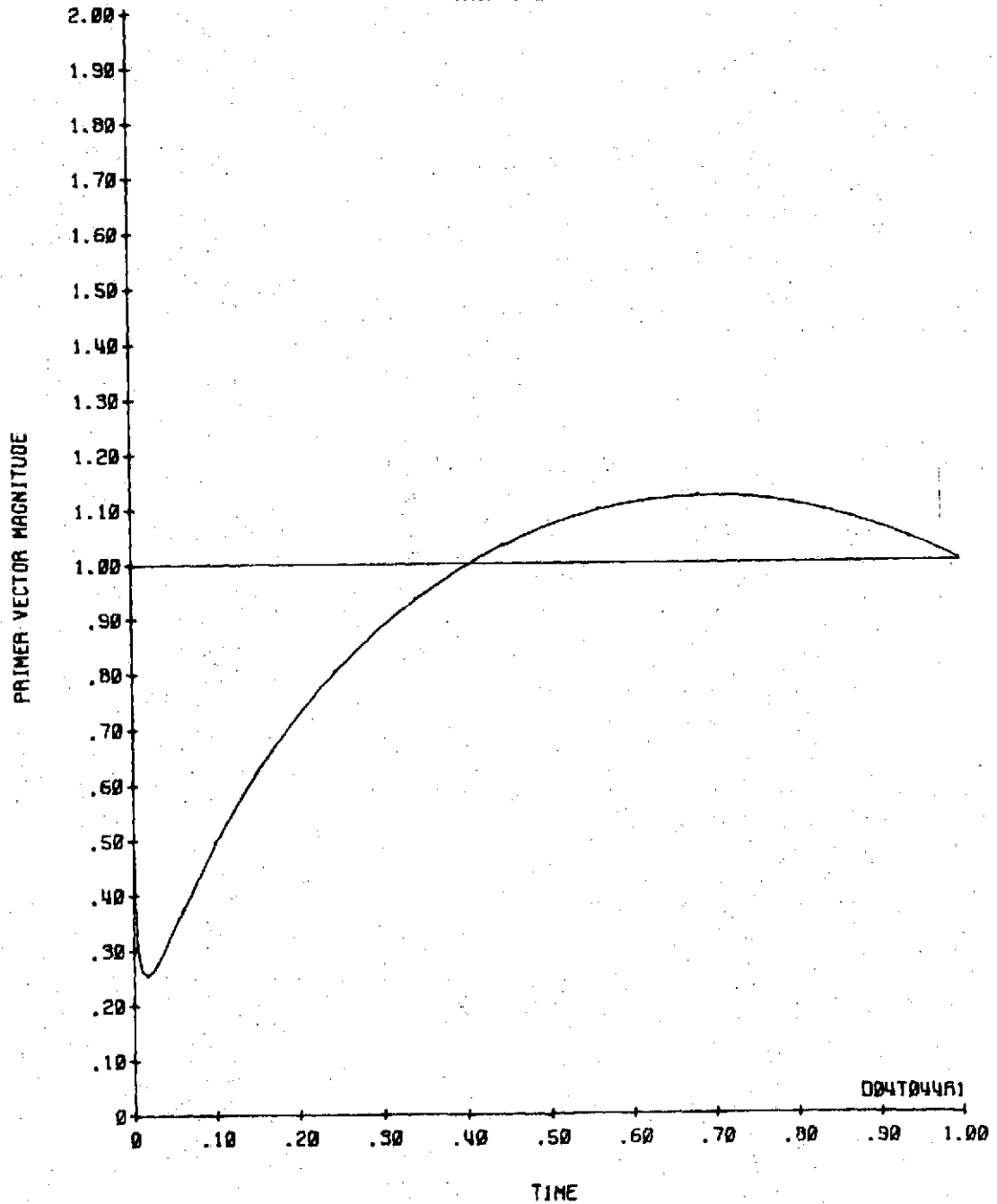


Figure 3g 2-Impulse Transfer (Example 3)

## EXAMPLE 4

### A. Purpose

This example may be used to check out Program PTP3I and associated subroutines. It is a 4-body 3-impulse transfer from noon July 4, 1978 to  $L_1$  in 44-days. The 2-impulse transfer shown in Ex. 3 was used as the reference.

### B. Input Parameters

\* Same as in Ex. 1.

The starting independent variables ( $\bar{v}_o, t_m$ ) shown here are values near the end of a sequence of computer runs.

<u>Card No.</u>	<u>Format</u>	<u>Parameter</u>	<u>Value</u>
1	4D20.11	GL1	*
		AUM	*
		UTIME	*
		UVELM	*
2	3D20.11	MS	*
		ME	*
		MM	*
3	3D20.11	TSTART	1.89051337719D+01
		TM	1.96168629221D+01
		TEND	1.96620272076D+01
4	4D20.11	RSE0(1)	*
		(2)	*
		(3)	*
		VSE0(1)	*
5	4D20.11	(2)	*
		(3)	*

		RSM0(1)	*
		(2)	*
6	4D20.11	(3)	*
		VSM0(1)	*
		(2)	*
		(3)	*
7	4D20.11	RSV0(1)	1. 98969880525D-01
		(2)	-9. 97069044391D-01
		(3)	6. 68331499258D-05
		VSV0(1)	1. 07508453453D+00
8	2D20.11	(2)	4. 28411625788D-01
		(3)	2. 23856828208D-02
9	4D20.11	VSVI(1)	1. 12559555237D+00
		(2)	5. 23005315320D-01
		(3)	3. 22363348080D-02
		VSVMP(1)	6. 12270768134D-01
10	2D20.11	(2)	7. 70222061894D-01
		(3)	-3. 40110040616D-04
11	4D20.11	RSVTAR(1)	8. 04147100222D-01
		(2)	-5. 98272665038D-01
		(3)	3. 54443659015D-05
		VSVTAR(1)	5. 74883737675D-01
12.	2D20.11	(2)	7. 90914481955D-01
		(3)	-5. 54211553533D-05

13	3D20.11	KNR	1.0D+00
		ERRMIN	1.0D-10
		ERRMXM	1.524D+02
14	3D20.11	FMINM	3.4D+03
		EPS	1.0D-02
		EPSV	1.0D-04
15	15	ICOMV	0
		ITLMAX	100
		ILINC	2
		ITDMAX	100
		IFILEX	0
16	4D20.11	V(1, 1)	1.50704497134D-04
		V(1, 2)	1.08127195736D-04
		V(1, 3)	-1.88527012972D-03
		V(1, 4)	-1.33164676422D-01
17	4D20.11	V(2, 1) = V(1, 2)	
		V(2, 2)	-1.72891113995D-04
		V(2, 3)	1.25136470933D-03
		V(2, 4)	1.00119585112D-01
18	4D20.11	V(3, 1) = V(1, 3)	
		V(3, 2) = V(2, 3)	
		V(3, 3)	-3.48824401009D-03
		V(3, 4)	-3.68585896226D-01

19            4D20.11             $V(4, 1) = V(1, 4)$   
     $V(4, 2) = V(2, 4)$   
     $V(4, 3) = V(3, 4)$   
     $V(4, 4) = -3.32401335566D+01$

C.    Output Parameters

The trajectory is shown in Figure 4a to 4f. The primer vector history is shown in Figure 4g. The interior impulse applied at about 41.9 days is nearly at the end of the 44-day transfer. The costs are

Initial impulse	3207.35 m/s
Interior impulse	273.44 m/s
Terminal impulse	<u>97.64</u>
	3578.43

The 3-impulse transfer costs about 3.50 m/s less than the 2-impulse transfer.

D.    Comment

The 3-impulse transfer is difficult to generate, especially when the 2-impulse transfer is nearly optimal. First, the difficulty is to break away from the 2-impulse transfer. Secondly, the cost gradient must be reduced several orders of magnitude lower than necessary in order to satisfy the condition

$$\dot{\lambda}_m^+ \cdot \bar{\lambda}_m \approx \dot{\lambda}_m^- \cdot \bar{\lambda}_m$$

while the cost remains practically unchanged. To accomplish this it may be necessary to reduce the single step allowable position error by an order of magnitude.

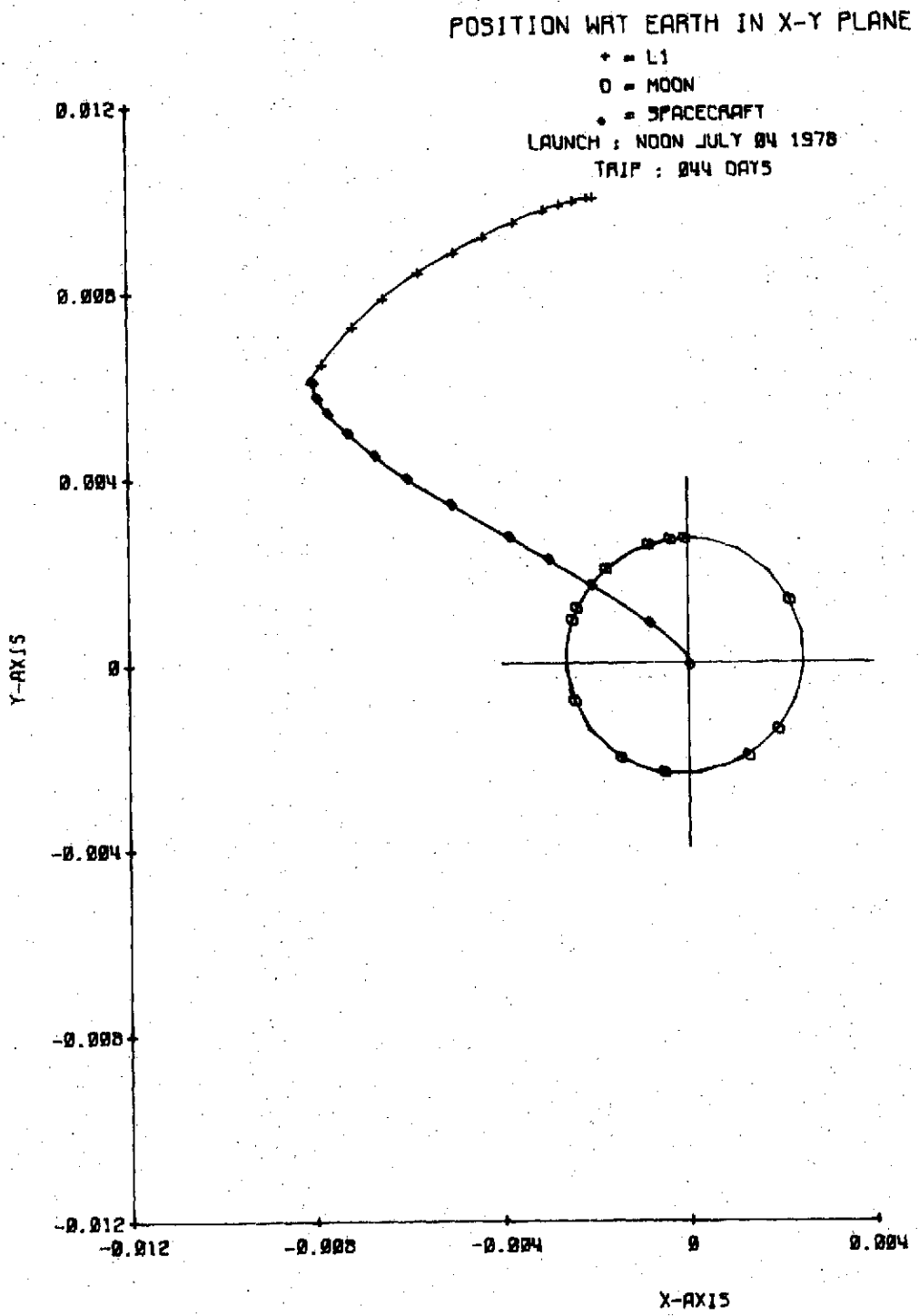


Figure 4a 3-Impulse Transfer (Example 4)

POSITION WRT MOON IN X-Y PLANE

X = EARTH

• = SPACECRAFT

LAUNCH : NOON JULY 04 1978

TRIP : 044 DAYS

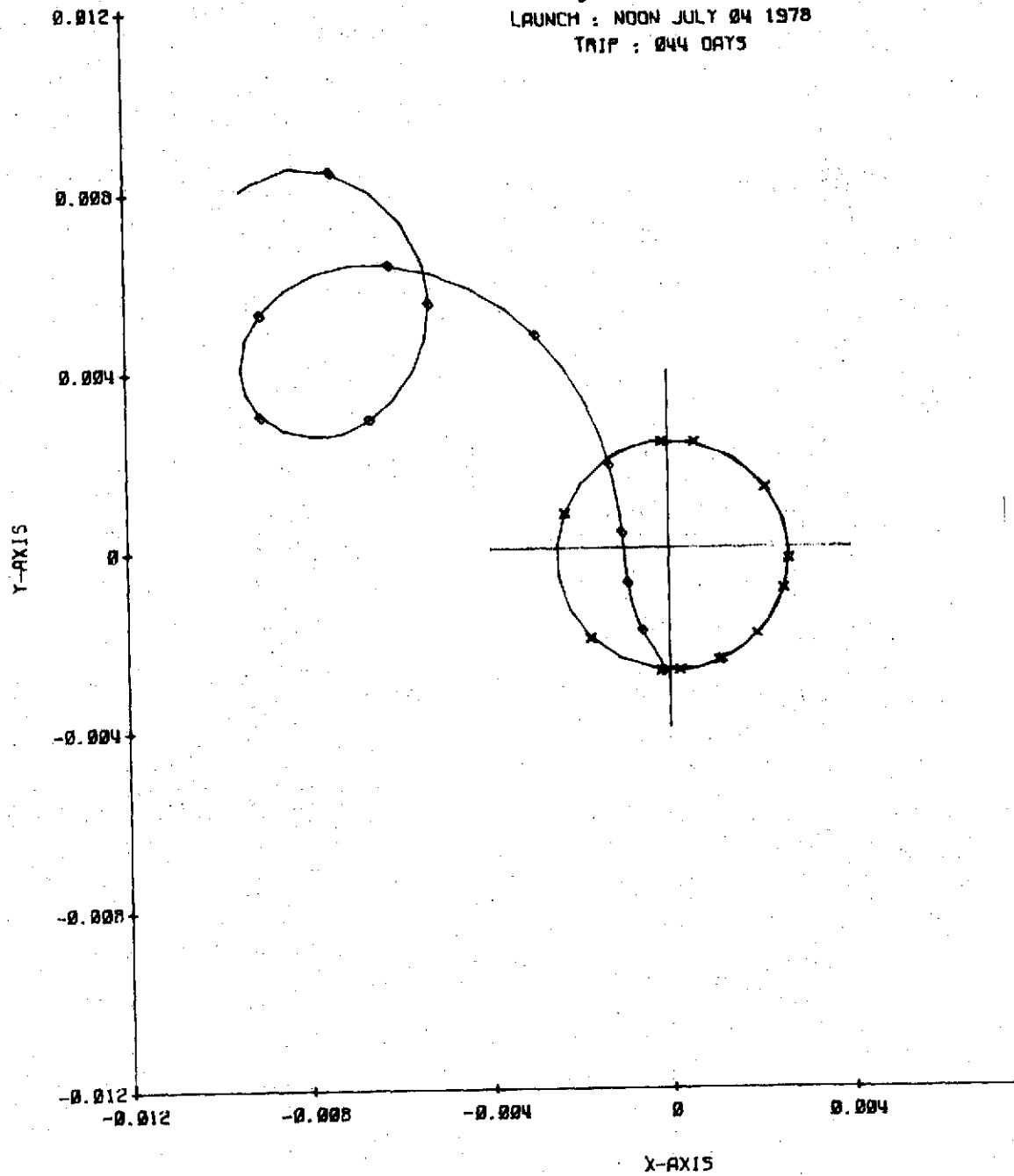


Figure 4b 3-Impulse Transfer (Example 4)

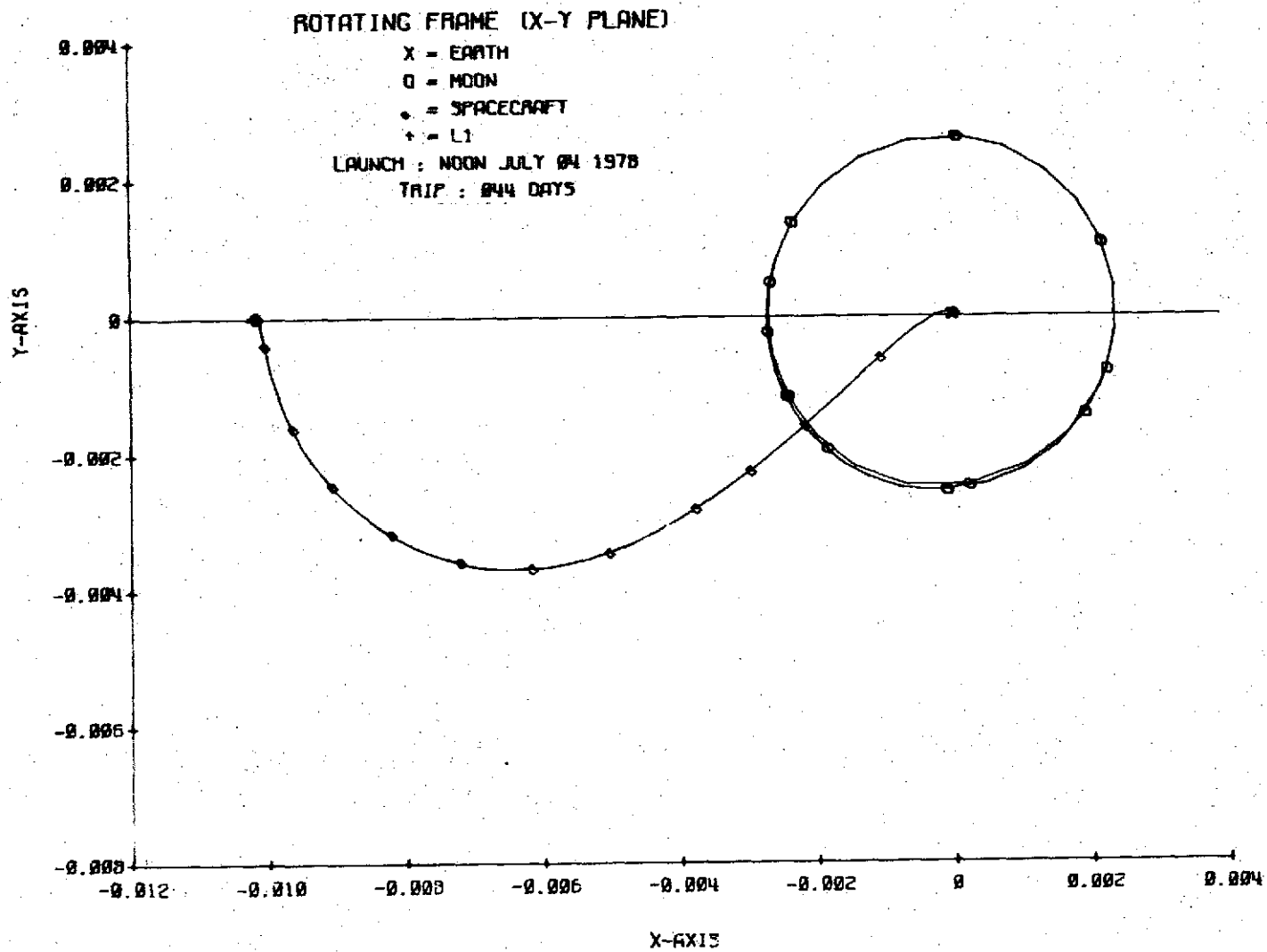


Figure 4c 3-Impulse Transfer (Example 4)



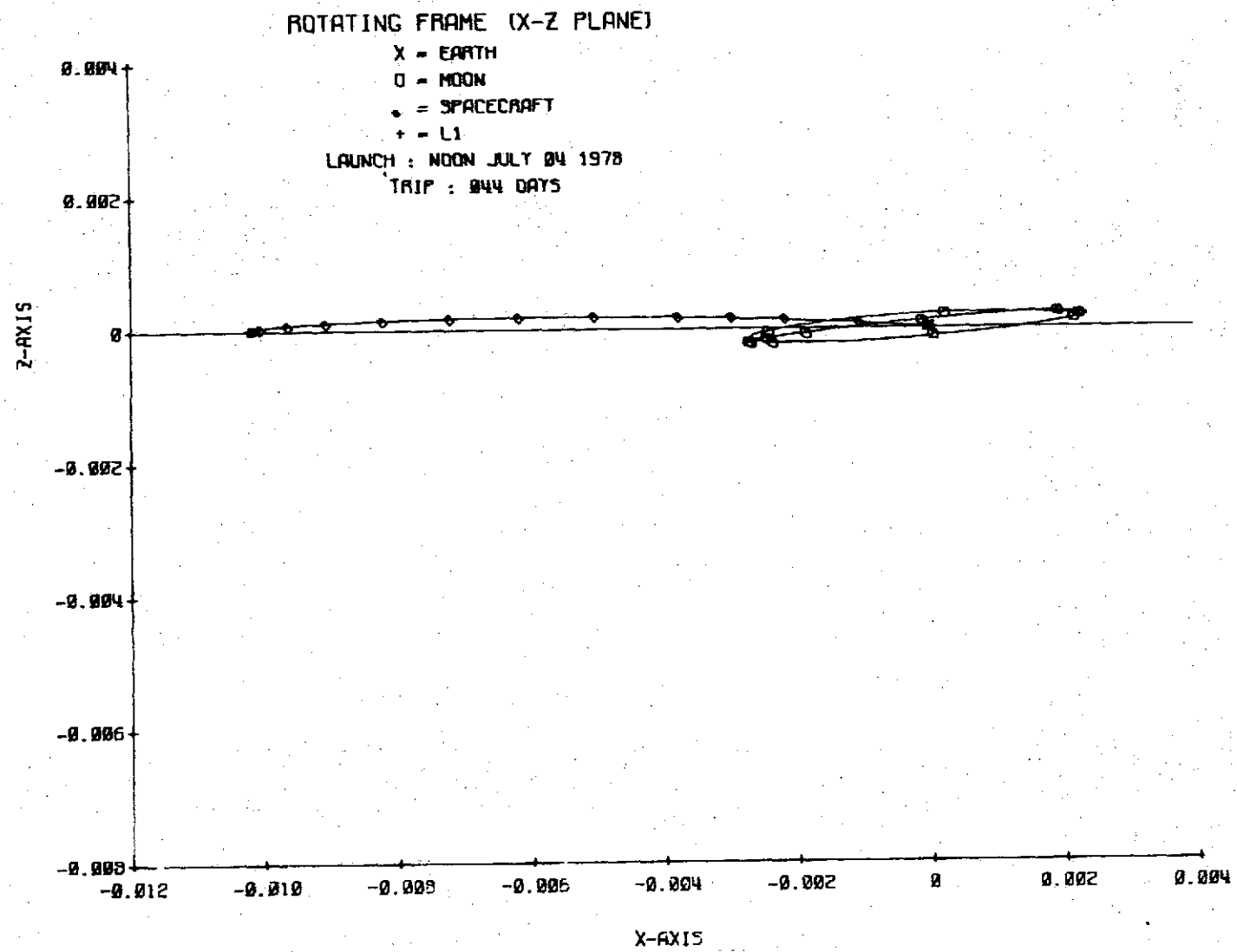


Figure 4d 3-Impulse Transfer (Example 4)

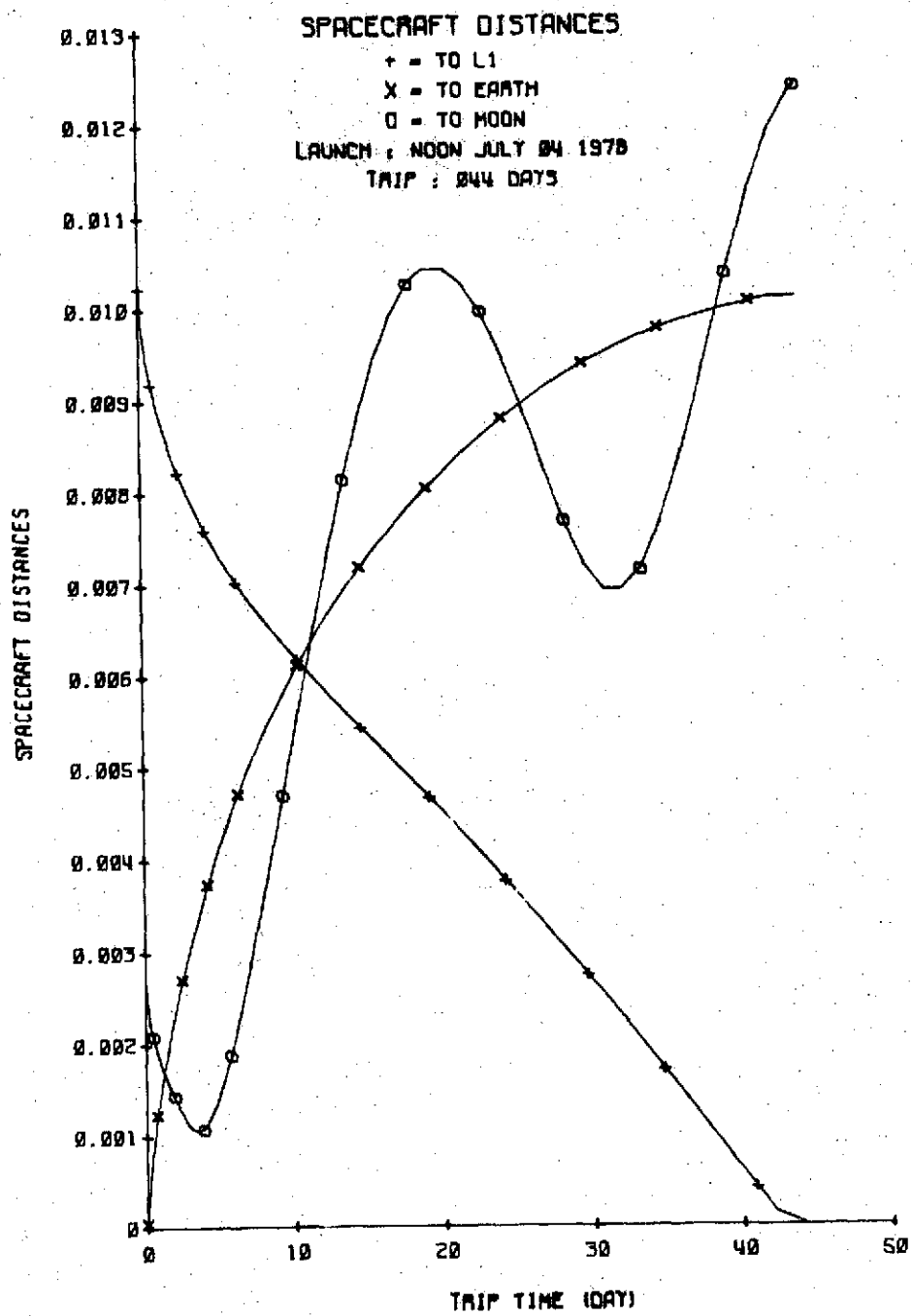


Figure 4e 3-Impulse Transfer (Example 4)

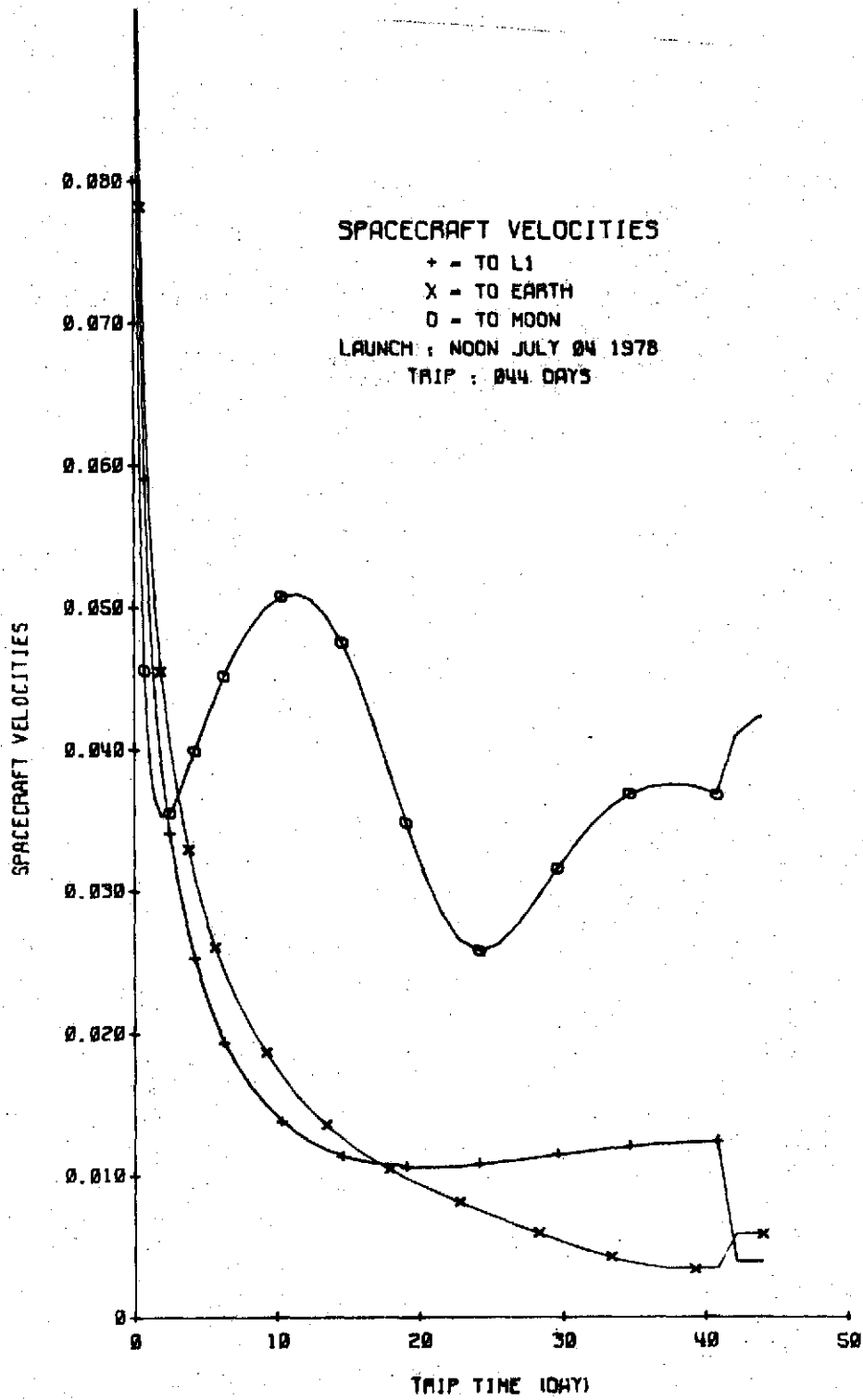


Figure 4f 3-Impulse Transfer (Example 4)

PRIMER VECTOR MAGNITUDE

LAUNCH : NOON JULY 04 1978

TRIP : 044 DAYS

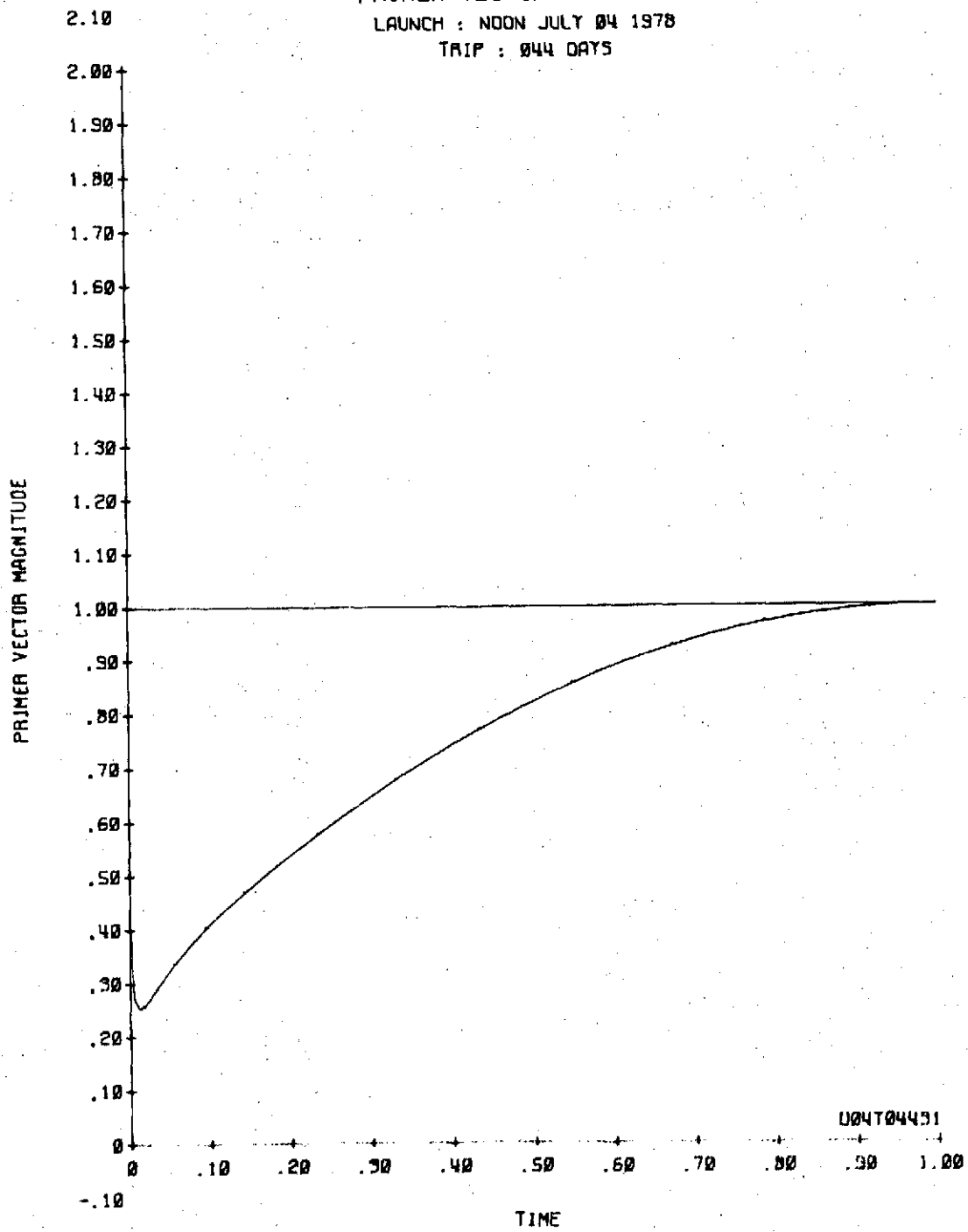


Figure 4g 3-Impulse Transfer (Example 4)

PROGRAM AND SUBROUTINE LISTING

TRAJ	LAMB
TRAJ3	LAMB3
EXLAM	DISP
EXLAM3	DISP3
ETP21	PTRAJ
ETP213	PTRAJ3
PTP31	FDATA
PTP313	FDATA3
FOURBY	COMAUG
THRBDY	COMDX
TWOBDY	UPX
CSTEP	PVEC
CSTEP3	RVEMV
DELRV	MXV
DELRV3	VVT
COMIC	DOT
COMIC3	VMAG
COMFG	INVERT
COMFG3	UNITV
COMF	VXV
COMF3	MTRANS
COMG	MXM
CTAR	
CTAR3	

```

C PROGRAM TRAJ
C PROGRAM PROPAGATES GIVEN STATES FROM TSTART TO TEND
  IMPLICIT REAL*8(A-H,L-M,O-Z)
  DIMENSION RSEO(3),VSEO(3),RSMO(3),VSMO(3),REVO(3),VEVO(3),
  1RSVO(3),VSVO(3),VSVI(3),VEVOP(3),PVO(6),RSVF(3),VSVF(3),RSEF(3),
  2VSEF(3),RSMF(3),VSMF(3),S11(3,3),S12(3,3),S21(3,3),S22(3,3)
  COMMON/CONST/MS,ME,MM,GL1,AUM,UTIME,UVELM,ERRMAX,DTR
  COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
  READ(5,100)GL1,AUM,UTIME,UVELM
  READ(5,100)MS,ME,MM
  READ(5,101)IMODE,IPV,IPTRAJ,IFILE
  READ(5,100)RSEO,VSEO,RSMO,VSMO
  READ(5,100)TDAY0,TRIPD,ERRMXM
  DTR=1.7453292519943296D-2
  ERRMAX=ERRMXM/AUM
  TDAYF=TDAY0+TRIPD
  TSTART=TDAY0/UTIME
  TEND=TDAYF/UTIME
  IMTX=0
  IPVTM=0
  ITER=0
  WRITE(6,102)GL1,AUM,UTIME,UVELM,ERRMXM,ERRMAX,MS,ME,MM
  WRITE(6,103)TDAY0,TRIPD,TDAYF,TSTART,TEND
  WRITE(6,105)RSEO,VSEO,RSMO,VSMO
  GO TO(1,2,3),IMODE
1 READ(5,100)REVMAG,VEVMAG,OINCD,OBLD,LOND,THED
  OINC=DTR*OINCD
  OBL=DTR*OBLD
  LON=DTR*LOND
  THE=DTR*THED
  CALL COMIC(REVMAG,VEVMAG,LON,THE,OINC,OBL,RSEO,VSEO,
  1REVO,VEVO,RSVO,VSVO,VSVI)
  WRITE(6,106)REVMAG,VEVMAG,OINCD,OBLD,LOND,THED
  GO TO 6
2 READ(5,100)REVO,VEVOP
  DO 201 I=1,3
  RSVO(I)=RSEO(I)+REVO(I)
201 VSVI(I)=VSEO(I)+VEVOP(I)
  WRITE(6,107)REVO,VEVOP
  GO TO 6
3 READ(5,100)RSVO,VSVI
6 WRITE(6,108)RSVO,VSVI
  IF(IPV.EQ.0) GO TO 7
  READ(5,100)PVO
  IMTX=1
  IPVTM=1
  WRITE(6,109)PVO
7 WRITE(6,104)IMODE,IPTRAJ,IFILE,IMTX,IPV,IPVTM
  CALL FOURBY(TSTART,TEND,RSVO,VSVI,RSEO,VSEO,RSMO,VSMO,
  1PVO,RSVF,VSVF,RSEF,VSEF,RSMF,VSMF,S11,S12,S21,S22)
  WRITE(6,110)RSVF,VSVF,RSEF,VSEF,RSMF,VSMF
100 FORMAT(4D20.11)
101 FORMAT(4I5)
102 FORMAT(1H ,40X,'FOUR-BODY TRAJECTORY'/1H0,T8,'GL1',T28,'AUM',
  1T48,'UTIME',T68,'UVELM',T88,'ERRMXM',T108,'ERRMAX'/1H ,
  21P6D20.11/1H ,T8,'MS',T28,'ME',T48,'MM'/1H ,1P3D20.11)
103 FORMAT(1H0,T8,'TDAY0',T28,'TRIPD',T48,'TDAYF',T68,'TSTART',

```

```
1T88,'TEND'/1H ,1P5D20.11)
104 FORMAT(1H0,T8,'IMODE',T28,'IPTRAJ',T48,'IFILE',T68,'IMTX',T88,
1'IPV',T108,'IPVTM'/1H ,110,5I20)
5 FORMAT(1H0,T8,'RSEO',T68,'VSEO'/1H ,1P6D20.11/1H ,T8,'RSMO',
1T68,'VSMO'/1H ,1P6D20.11)
106 FORMAT(1H0,T8,'REVMAG',T28,'VEVMAG',T48,'OINCD',T68,'OBLD',
1T88,'LOND',T108,'THED'/1H ,1P6D20.11)
107 FORMAT(1H0,T8,'REVO',T68,'VEVOP'/1H ,1P6D20.11)
108 FORMAT(1H0,T8,'RSVO',T68,'VSVI'/1H ,1P6D20.11)
109 FORMAT(1H0,T8,'PVO'/1H ,1P6D20.11)
110 FORMAT(1H0,T8,'RSVF',T68,'VSVF'/1H ,1P6D20.11/1H ,T8,'RSEF',
1T68,'VSEF'/1H ,1P6D20.11/1H ,T8,'RSMF',T68,'VSMF'/1H ,1P6D20.11)
RETURN
END
```

```

C PROGRAM TRAJ3
C PROGRAM PROPAGATES GIVEN STATES FROM TSTART TO TEND IN
C EARTH-MOON-VEHICLE SPACE.
  IMPLICIT REAL*8(A-H,L-M,O-Z)
  DIMENSION REMO(3),VEMO(3),REVO(3),VEVO(3),VEVOP(3),REVF(3),
1 VEVF(3),REMF(3),VEMF(3),S11(3,3),S12(3,3),S21(3,3),S22(3,3)
  COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER
  COMMON/CONST3/ME,MM,GAMMA,UDM,UTIME,UVELM,ERRMAX,DTR
  READ(5,100)GAMMA,UDM,UTIME,UVELM
  READ(5,100)ME,MM
  READ(5,101)IMODE,IPV,IPTRAJ,IFILE
  READ(5,100)REMO,VEMO
  READ(5,100)TDAY0,TRIPD,ERRMXM
  DTR=1.7453292519943296D-2
  ERRMAX=ERRMXM/UDM
  TDAYF=TDAY0+TRIPD
  TSTART=TDAY0/UTIME
  TEND=TDAYF/UTIME
  IMTX=0
  IPVTM=0
  ITER=0
  WRITE(6,102)GAMMA,UDM,UTIME,UVELM,ERRMXM,ERRMAX,ME,MM
  WRITE(6,103)TDAY0,TRIPD,TDAYF,TSTART,TEND
  WRITE(6,105)REMO,VEMO
  GO TO (1,2),IMODE
1 READ(5,100)REVMAG,VEVMAG,OINCD,LOND,THED
  OINC=DTR*OINCD
  LON=DTR*LOND
  THE=DTR*THED
  CALL COMIC3(REVMAG,VEVMAG,LON,THE,OINC,REVO,VEVO,VEVOP)
  WRITE(6,106)REVMAG,VEVMAG,OINCD,LOND,THED
  GO TO 3
2 READ(5,100)REVO,VEVOP
3 WRITE(6,107)REVO,VEVOP
  IF(IPV.EQ.0) GO TO 4
  READ(5,100)PVO
  IMTX=1
  IPVTM=1
  WRITE(6,108)PVO
4 WRITE(6,104)IMODE,IPTRAJ,IFILE,IMTX,IPV,IPVTM
  CALL THRBODY(TSTART,TEND,REVO,VEVOP,REMO,VEMO,PVO,REVF,VEVF,
1 REMF,VEMF,S11,S12,S21,S22)
  WRITE(6,109)REVF,VEVF,REMF,VEMF
100 FORMAT(4D20.11)
101 FORMAT(4I5)
102 FORMAT(1H ,40X,'THREE-BODY TRAJECTORY'/1H0,T8,'GAMMA',T28,
1 'UDM',T48,'UTIME',T68,'UVELM',T88,'ERRMXM',T108,'ERRMAX'/1H ,
21P6D20.11/1H ,T8,'ME',T28,'MM'/1H ,1P2D20.11)
103 FORMAT(1H0,T8,'TDAY0',T28,'TRIPD',T48,'TDAYF',T68,'TSTART',
1T88,'TEND'/1H ,1P5D20.11)
104 FORMAT(1H0,T8,'IMODE',T28,'IPTRAJ',T48,'IFILE',T68,'IMTX',
1T88,'IPV',T108,'IPVTM'/1H ,1I0,5I20)
105 FORMAT(1H0,T8,'REMO',T68,'VEMO'/1H ,1P6D20.11)
106 FORMAT(1H0,T8,'REVMAG',T28,'VEVMAG',T48,'OINCD',T68,'LOND',
1T88,'THED'/1H ,1P5D20.11)
107 FORMAT(1H0,T8,'REVO',T68,'VEVOP'/1H ,1P6D20.11)
108 FORMAT(1H0,T8,'PVO'/1H ,1P6D20.11)

```



```
109. FORMAT(1H0,T8,'REVF',T68,'VEVF'/1H ,1P6D20.11/1H0,T8,'REMF',  
1T68,'VEMF'/1H ,1P6D20.11)  
RETURN  
END
```

```

C PROGRAM EXLAM
C PROGRAM SOLVES 2-POINT BOUNDARY VALUE PROBLEM OF TRANSFER FROM
C A GIVEN INITIAL POSITION TO A GIVEN FINAL POSITION IN FIXED TIME.
  IMPLICIT REAL*8(A-H,K-M,O-Z)
  DIMENSION RSEO(3),VSEO(3),RSMO(3),VSMO(3),REVO(3),VEVO(3),
  1RSVO(3),VSVO(3),VSVI(3),VEVOP(3),RSVF(3),VSVF(3),RSEF(3),
  2VSEF(3),RSMF(3),VSMF(3),S11(3,3),S12(3,3),S21(3,3),S22(3,3),
  3RSVTAR(3),PVO(6)
  COMMON/CONST/MS,ME,MM,GL1,AUM,UTIME,UVELM,ERRMAX,DTR
  COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
  READ(5,100)GL1,AUM,UTIME,UVELM
  READ(5,100)MS,ME,MM
  READ(5,101)IMODE,ITLMAX,IPTJX,IFILEX
  READ(5,100)RSEO,VSEO,RSMO,VSMO
  READ(5,100)TDAYO,TRIPD,ERRMXM
  READ(5,100)RSVTAR,ERRMIN
  READ(5,100)KNR
  DTR=1.7453292519943296D-2
  ERRMAX=ERRMXM/AUM
  TDAYF=TDAYO+TRIPD
  TSTART=TDAYO/UTIME
  TEND=TDAYF/UTIME
  IMTX=1
  IPV=0
  IPVTM=0
  IPTRAJ=0
  IFILE=0
  WRITE(6,102)GL1 AUM,UTIME,UVELM,ERRMXM,ERRMAX,MS,ME,MM,ERRMIN,
  1KNR
  WRITE(6,103)TDAYO,TRIPD,TDAYF,TSTART,TEND
  WRITE(6,104)IMODE,ITLMAX,IPTJX,IFILEX,IPTRAJ,IFILE
  WRITE(6,105)RSEO,VSEO,RSMO,VSMO,RSVTAR
  GO TO (1,2,3),IMODE
  1 READ(5,100)REVMAG,VEVMAG,OINCD,OBLD,LOND,THED
  OINC=DTR*OINCD
  OBL=DTR*OBLD
  LON=DTR*LOND
  THE=DTR*THED
  CALL COMIC(REVMAG,VEVMAG,LON,THE,OINC,OBL,VCIR,RSEO,VSEO,
  1REVO,VEVO,RSVO,VSVO,VSVI)
  WRITE(6,106)REVMAG,VEVMAG,OINCD,OBLD,LOND,THED
  GO TO 6
  2 READ(5,100)REVO,VEVOP
  DO 201 I=1,3
  RSVO(I)=RSEO(I)+REVO(I)
  201 VSVI(I)=VSEO(I)+VEVOP(I)
  WRITE(6,107)REVO,VEVOP
  GO TO 6
  3 READ(5,100)RSVO,VSVI
  6 WRITE(6,108)RSVO,VSVI
  CALL LAMB(TSTART,TEND,RSVO,VSVI,RSEO,VSEO,RSMO,VSMO,KNR,ITLMAX,
  1ERRMIN,RSVTAR,RSVF,VSVF,RSEF,VSEF,RSMF,VSMF,S11,S12,S21,S22)
  IPTRAJ=IPTJX
  IFILE=IFILEX
  IMTX=0
  IF(IPTRAJ.GT.0) GO TO 7
  IF(IFILE.GT.0) GO TO 7

```

```

GO TO 8
7 CALL FOURBY(TSTART,TEND,RSVO,VSVI,RSEO,VSEO,RSMO,VSMO,PVO,
1RSVF,VSVF,RSEF,VSEF,RSMF,VSMF,S11,S12,S21,S22)
8 CONTINUE
100 FORMAT(4D20.11)
101 FORMAT(4I5)
102 FORMAT(1H ,40X,'FOUR-BODY LAMBERT PROBLEM'/1H0,T8,'GL1',T28,
1'AUM',T48,'UTIME',T68,'UVELM',T88,'ERRMXM',T108,'ERRMAX'/1H ,
21P6D20.11/1H ,T8,'MS',T28,'ME',T48,'MM',T68,'ERRMIN',T88,
3'KNR'/1H ,1P5D20.11)
103 FORMAT(1H0,T8,'TDAY0',T28,'TRIP0',T48,'TDAYF',T68,'TSTART',
1T88,'TEND'/1H ,1P5D20.11)
104 FORMAT(1H ,T8,'IMODE',T28,'ITLMAX',T48,'IPTJX',T68,'IFILEX',
1T88,'IPTRAJ',T108,'IFILE'/1H ,110,5I20)
105 FORMAT(1H0,T8,'RSEO',T68,'VSEO'/1H ,1P6D20.11/1H ,T8,'RSMO',
1T68,'VSMO'/1H ,1P6D20.11/1H ,T8,'RSVTAR'/1H ,1P3D20.11)
106 FORMAT(1H0,T8,'REVMAG',T28,'VEVMAG',T48,'OINCD',T68,'OBLD',
1T88,'LOND',T108,'THED'/1H ,1P6D20.11)
107 FORMAT(1H0,T8,'REVO',T68,'VEVOP'/1H ,1P6D20.11)
108 FORMAT(1H0,T8,'RSVO',T68,'VSVI'/1H ,1P6D20.11)
RETURN
END

```

```

PROGRAM EXLAM3
PROGRAM SOLVES 2-POINT BOUNDARY VALUE PROBLEM OF TRANSFER FROM
A GIVEN INITIAL POSITION TO A GIVEN FINAL POSITION IN FIXED
TIME.
IMPLICIT REAL*8(A-H,K-M,O-Z)
DIMENSION REMO(3),VEMO(3),REVTAR(3),REVO(3),VEVO(3),VEVOP(3),
1REVF(3),VEVF(3),REMF(3),VEMF(3),S11(3,3),S12(3,3),S21(3,3),
2S22(3,3),PVO(6)
COMMON/CONST3/ME,MM,GAMMA,UDM,UTIME,UVELM,ERRMAX,DTR
COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
READ(5,100)GAMMA,UDM,UTIME,UVELM
READ(5,100)ME,MM
READ(5,101)IMODE,ITLMAX,IPTJX,IFILEX
READ(5,100)REMO,VEMO
READ(5,100)TDAYO,TRIPD,ERRMXM
READ(5,100)REVTAR,ERRMIN
READ(5,100)KNR
DTR=1.7453292519943296D-2
ERRMAX=ERRMXM/UDM
TDAYF=TDAYO+TRIPD
TSTART=TDAYO/UTIME
TEND=TDAYF/UTIME
IMTX=1
IPV=0
IPVTM=0
IFILE=0
WRITE(6,102)GAMMA,UDM,UTIME,UVELM,ERRMXM,ERRMAX,ME,MM,ERRMIN,KNR
WRITE(6,103)TDAYO,TRIPD,TDAYF,TSTART,TEND
WRITE(6,104)IMODE,ITLMAX,IPTJX,IFILEX
WRITE(6,105)REMO,VEMO,REVTAR
GO TO (1,2),IMODE
1 READ(5,100)REVMAG,VEVMAG,OINCD,LOND,THED
OINC=DTR*OINCD
LON=DTR*LOND
THE=DTR*THED
CALL COMIC3(REVMAG,VEVMAG,LON,THE,OINC,REVO,VEVO,VEVOP)
WRITE(6,106)REVMAG,VEVMAG,OINCD,LOND,THED
GO TO 3
2 READ(5,100)REVO,VEVOP
3 WRITE(6,107)REVO,VEVOP
CALL LAMB3(TSTART,TEND,REVO,VEVOP,REMO,VEMO,KNR,ITLMAX,ERRMIN,
1REVTAR,REVF,VEVF,REMF,VEMF,S11,S12,S21,S22)
WRITE(6,108)REVF,VEVF,REMF,VEMF
IPTRAJ=IPTJX
IFILE=IFILEX
IMTX=0
IF(IPTRAJ.EQ.0) GO TO 4
IF(IFILE.EQ.0) GO TO 4
GO TO 5
4 CALL THRBODY(TSTART,TEND,REVO,VEVOP,REMO,VEMO,PVO,REVF,VEVF,
1REMF,VEMF,S11,S12,S21,S22)
5 CONTINUE
100 FORMAT(4D20.11)
101 FORMAT(4I5)
102 FORMAT(1H ,40X,'3-BODY LAMBERT PROBLEM'/1H0,T8,'GAMMA',T28,
1'UDM',T48,'UTIME',T68,'UVELM',T88,'ERRMXM',T108,'ERRMAX'/1H ,
21P6D20.11/1H ,T8,'ME',T28,'MM',T48,'ERRMIN',T68,'KNR'/1H ,

```

```
31P4D20.11)
103 FORMAT(1H0,T8,'TDAY0',T28,'TRIPD',T48,'TDAYF',T68,'TSTART',
1T88,'TEND'/1H,1P5D20.11)
p4 FORMAT(1H,T8,'IMODE',T28,'ITLMAX',T48,'IPTJX',T68,'IFILEX'/
11H,110,3120)
105 FORMAT(1H0,T8,'REMO',T68,'VEMO'/1H,1P6D20.11/1H,T8,'REVTAR'/
11H,1P3D20.11)
106 FORMAT(1H0,T8,'REVMAG',T28,'VEVMAG',T48,'OINCD',T68,'THED'/1H,
11P4D20.11)
107 FORMAT(1H0,T8,'REVO',T68,'VEVOP'/1H,1P6D20.11)
108 FORMAT(1H0,T8,'REVF',T68,'VEVF'/1H,1P6D20.11/1H,T8,'REMF',
1T68,'VEMF'/1H,1P6D20.11)
RETURN
END
```

```

PROGRAM ETP21
PROGRAM COMPUTES FUEL OPTIMAL 2-IMPULSE TRANSFER FROM A PARKING
ORBIT OF GIVEN INCLINATION TO A GIVEN FINAL POSITION AND VELOCITY
IN FIXED TIME.
IMPLICIT REAL*8(A-H,K-M,O-Z)
DIMENSION RSEO(3),VSEO(3),RSMO(3),VSMO(3),XD(3),X(3),XS(3),
1RSVTAR(3),VSVTAR(3),V(3,3),GS(3),TSIS(3),LTS(3,3),LS(3,3),
2S11S(3,3),S12S(3,3),S21S(3,3),S22S(3,3),GGS(3),VSAV(3,3),G(3),
3TSI(3),LT(3,3),L(3,3),GG(3),S11(3,3),S12(3,3),S21(3,3),
4S22(3,3),DX(3),DELX(3),UVI(3),UVF(3),TEMP(3,3),DUM(3),UVID(3)
DIMENSION R(3),RMVO(3),VMVO(3),RSVO(3),VSVI(3),UVIS(3),UVFS(3),
1GD(3),TSID(3),LTD(3,3),LD(3,3),S11D(3,3),S12D(3,3),S21D(3,3),
2S22D(3,3),RSVF(3),VSVF(3),RSEF(3),VSEF(3),RSMF(3),VSMF(3),
3DUVI(3),REVO(3),VEVO(3),UVFD(3),PVO(6),GGD(3),VSVO(3)
COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
COMMON/CONST/MS,ME,MM,GL1,AUM,UTIME,UVELM,ERRMAX,DTR
COMMON/TARG/AY,AZ,ATAR,RSVTAR,VSVTAR
READ(5,100)GL1,AUM,UTIME,UVELM
READ(5,100)MS,ME,MM
READ(5,100)REVMAG,OINCD,OBLD,ERRMXM
READ(5,100)TDAYO,TTRIPD
READ(5,100)VEVMAG,LOND,THED
READ(5,100)RSEO,VSEO,RSMO,VSMO
READ(5,100)EPS,EPSTSI,KDX,EPSTV
READ(5,101)ICOMV,ITERMX,IFILEX,ITAR
IF(ITAR.GT.0) GO TO 21
READ(5,100)AYM,AZM,ATARD
GO TO 22
21 READ(5,100)RSVTAR,VSVTAR
22 IF(ICOMV.GT.0) GO TO 19
OPTION TO INPUT VARIANCE MATRIX (ICOMV=0)
READ(5,100)((V(I,J),J=1,3),I=1,3)
GO TO 1
19 DO 20 I=1,3
DO 20 J=1,3
V(I,J)=0.
IF(I.EQ.J) V(I,J)=1.0
20 CONTINUE
1 IPV=0
IPVTM=0
IPTRAJ=0
IFILE=0
IMTX=1
ITER=0
ITERD=0
DTR=1.7453292519943296D-2
ERRMAX=ERRMXM/AUM
IF(ITAR.GT.0) GO TO 23
AY=AYM/AUM
AZ=AZM/AUM
ATAR=DTR*ATARD
23 TDAYF=TDAYO+TTRIPD
TTRIP=TTRIPD/UTIME
TSTART=TDAYO/UTIME
TEND=TDAYF/UTIME
ALPHA=1.0D-3
BETA=10.

```

```

ALPHAM=ALPHA/(1.-ALPHA)
BETAM=BETA/(BETA-1.)
WRITE(6,102)
WRITE(6,103)GL1,AUM,UTIME,UVELM,ERRMXM,ERRMAX
WRITE(6,104)MS,ME,MM,OINCD,OBLD
WRITE(6,105)TDAYO,TTRIPD,TDAYF,TSTART,TTRIP,TEND
WRITE(6,106)RSEO,VSEO
WRITE(6,107)RSMO,VSMO
WRITE(6,108)VEVMAG,LOND,THED,REVMAG
WRITE(6,109)ALPHA,BETA,EPS,EPSTSI,KDX,EPSV
WRITE(6,110)ICOMV,ITERMX,IFILEX,ITAR
IF(ITAR.GT.0) GO TO 24
WRITE(6,124)AYM,AZM,ATARD
GO TO 25
24 WRITE(6,125)RSVTAR,VSVTAR
25 LON=DTR*LOND
THE=DTR*THED
OINC=DTR*OINCD
OBL=DTR*OBLD
XS(1)=VEVMAG
XS(2)=LON
XS(3)=THE
C COMPUTE INITIAL NOMINAL TRAJECTORY
CALL COMFG(TSTART,TEND,REVMAG,XS,OINC,OBL,RSEO,VSEO,RSMO,VSMO,
IFS,TESTRS,GS,TSIS,LTS,S11S,S12S,S21S,S22S,UVIS,UVFS)
ITER=1
CALL COMAUG(FS,GS,TSIS,LTS,V,LS,FGS,GGS)
IF(ICOMV.EQ.0) GO TO 200
C OPTION TO COMPUTE TRIAL VARIANCE MATRIX (ICOMV=1)
GGSMAG=VMAG(GGS)
DO 2 I=1,3
DO 2 J=1,3
V(I,J)=V(I,J)*EPSV/GGSMAG
2 CONTINUE
200 WRITE(6,111)((V(I,J),J=1,3),I=1,3)
C SAVE VARIANCE MATRIX
DO 3 I=1,3
DO 3 J=1,3
3 VSAV(I,J)=V(I,J)
KV=1.
KDXSAV=KDX
IF(TESTRS.GT.EPSTSI)GO TO 8
C UPDATE VARIABLES
CALL UPX(XS,FS,TESTRS,GS,TSIS,LTS,LS,FGS,GGS,S11S,S12S,S21S,
1S22S,UVIS,UVFS,X,F,TESTR,G,TSI,LT,L,FG,GG,S11,S12,S21,S22,
2UVI,UVF)
WRITE(6,112)X,F,FG,TESTR
WRITE(6,113)TSI,G
WRITE(6,114)LT
WRITE(6,115)GG
C ACCELERATED GRADIENT PROJECTION ITERATION LOOP
4 ITERD=ITERD+1
5 DO 501 I=1,3
DO 501 J=1,3
501 V(I,J)=KV*VSAV(I,J)
6 CALL MXV(V,GG,R,3,3)
P=DOT(GG,R,3)

```

```

IF((P-2.*FG).LE.EPS) GO TO 7
LAM=2.*FG/P
CALL VVT(R,TEMP,3)
DO 601 I=1,3
DO 601 J=1,3
601 VSAV(I,J)=V(I,J)+(LAM-1.)*TEMP(I,J)/P
GO TO 5
7 CALL MXV(V,GG,DX,3,3)
FORM TRIAL NEW INDEPENDENT VARIABLES
DO 701 I=1,3
701 XS(I)=X(I)-DX(I)
WRITE(6,116)ITERD,DX
WRITE(6,117)XS,KV
WRITE(6,111)((V(I,J),J=1,3),I=1,3)
CALL COMFG(TSTART,TEND,REVMAG,XS,OINC,OBL,RSEO,VSEO,RSMO,VSMO,
IFS,TESTRS,GS,TSIS,LTS,S11S,S12S,S21S,S22S,UVIS,UVFS)
CALL COMAUG(FS,GS,TSIS,LTS,V,LS,FGS,GGs)
IF(TESTRS.LT.EPSTSI) GO TO 11
CONSTRAINT RESTORATION
8 WRITE(6,118)
801 CALL COMDX(LTS,LS,TSIS,DX)
WRITE(6,119)DX
9 DO 901 I=1,3
DELX(I)=KDX*DX(I)
901 XD(I)=XS(I)+DELX(I)
WRITE(6,120)KDX,DELX
CALL COMFG(TSTART,TEND,REVMAG,XD,OINC,OBL,RSEO,VSEO,RSMO,VSMO,
IFD,TESTRD,GD,TSID,LTD,S11D,S12D,S21D,S22D,UVID,UVFD)
IF(TESTRD.LT.TESTRS) GO TO 10
KDX=KDX/10.
WRITE(6,126)
GO TO 9
10 CALL COMAUG(FD,GD,TSID,LTD,V,LD,FGD,GGD)
CALL UPX(XD,FD,TESTRD,GD,TSID,LTD,LD,FGD,GGD,S11D,S12D,S21D,
1S22D,UVID,UVFD,XS,FS,TESTRS,GS,TSIS,LTS,LS,FGS,GGs,S11S,S12S,
2S21S,S22S,UVIS,UVFS)
KDX=2.*KDX
IF(KDX.GT.1.0)KDX=1.0
IF(TESTRS.GT.EPSTSI) GO TO 801
KDX=KDXSAV
IF(ITERD.EQ.0) GO TO 11
IF(FS.LT.F) GO TO 11
KV=KV/2.
GO TO 5
CONSTRAINT RESTORED
11 IF(ITERD.LT.ITERMX)GO TO 300
WRITE(6,127)
GO TO 17
300 IF(ITERD.EQ.0)GO TO 16
UPDATE VARIANCE MATRIX
CALL MXV(V,GGs,R,3,3)
P=DOT(GGS,R,3)
WRITE(6,121)FS,TESTRS,FGS,TSIS
WRITE(6,122)GS,GGs
WRITE(6,123)P
IF(P.LT.EPS) GO TO 17
GAMMA=-DOT(GG,R,3)/P

```



```

IF(GAMMA.EQ.(-1.)) GO TO 12
IF(GAMMA.GE.ALPHAM)GO TO 13
LAM=ALPHA
GO TO 14
12 IF(GAMMA.LE.(-BETAM)) GO TO 13
LAM=BETA
GO TO 14
13 LAM=GAMMA/(GAMMA+1.)
14 CALL VVT(R,TEMP,3)
DO 15 I=1,3
DO 15 J=1,3
15 V(I,J)=V(I,J)+(LAM-1.)*TEMP(I,J)/P
IF(FGS.LT.FG) GO TO 16
GO TO 6
16 KV=1.
CALL UPX(XS,FS,TESTRS,GS,TSIS,LTS,LS,FGS,GG,S11S,S12S,S21S,
1S22S,UVIS,UVFS,X,F,TESTR,G,TSI,LT,L,FG,GG,S11,S12,S21,S22,
2UVI,UVF)
IF(ITERD.EQ.0) GO TO 4
DO 161 I=1,3
DO 161 J=1,3
161 VSAV(I,J)=V(I,J)
GO TO 4
17 CALL UPX(XS,FS,TESTRS,GS,TSIS,LTS,LS,FGS,GG,S11S,S12S,S21S,
1S22S,UVIS,UVFS,X,F,TESTR,G,TSI,LT,L,FG,GG,S11,S12,S21,S22,
2 UVI,UVF)
C COMPUTE PRIMER VECTOR DERIVATIVE
CALL MXV(S11,UVI,DUM,3,3)
CALL INVERT(S12,TEMP)
DO 18 I=1,3
18 DUM(I)=UVF(I)-DUM(I)
CALL MXV(TEMP,DUM,DUVI,3,3)
DO 400 I=1,3
PVO(I)=UVI(I)
400 PVO(I+3)=DUVI(I)
IPV=1
IPVTM=1
IPTRAJ=1
IFILE=IFILEX
C GENERATE OPTIMAL TRAJECTORY AND PRIMER VECTOR HISTORY
VEVMAG=X(1)
LON=X(2)
THE=X(3)
CALL COMIC(REVMAG,VEVMAG,LON,THE,OINC,OBL,RSEO,VSEO,REVO,VEVO,
1RSVO,VSVO,VSVI)
CALL FOURBY(TSTART,TEND,RSVO,VSVI,RSEO,VSEO,RSMO,
1VSMO,PVO,RSVF,VSVF,RSEF,VSEF,RSMF,VSMF,S11,S12,S21,S22)
100 FORMAT(4D20.11)
101 FORMAT(4I5)
102 FORMAT(1H ,T8,'4-BODY 2-IMPULSE OPTIMAL TRANSFER FROM EARTH' /
11H ,T8,'TO HALO TARGET AT L1. ITERATE ON VELOCITY MAGNITUDE' /
21H ,T8,'LONGITUDE OF NODE AND ORBITAL ANGLE FROM NODE USING' /
31H ,T8,'ACCELERATED GRADIENT PROJECTION AND DAVIDON VARIANCE' /
41H ,T8,'METHOD')
103 FORMAT(1H0,T8,'GL1',T28,'AUM',T48,'UTIME',T68,'UVELM',T88,
1'ERRMXM',T108,'ERRMAX'/1H ,1P6D20.11)
104 FORMAT(1H ,T8,'MS',T28,'ME',T48,'MM',T68,'OINC',T88,'OBLD' /

```

```

11H ,1P5D20.11)
105 FORMAT(1H ,T8,'TDAYO',T28,'TTRIPD',T48,'TDAYF',T68,'TSTART',
1T88,'TTRIP',T108,'TEND'/1H ,1P6D20.11)
6 FORMAT(1H ,T8,'RSEO',T68,'VSEO'/1H ,1P6D20.11)
107 FORMAT(1H ,T8,'RSMO',T68,'VSMO'/1H ,1P6D20.11)
108 FORMAT(1H ,T8,'VEVMAG',T28,'LOND',T48,'THED',T68,'REVMAG'/
11H ,1P4D20.11)
109 FORMAT(1H ,T8,'ALPHA',T28,'BETA',T48,'EPS',T68,'EPSTSI',T88,
1'KDX',T108,'EPSV'/1H ,1P6D20.11)
110 FORMAT(1H ,T8,'ICOMV',T28,'ITERMX',T48,'IFILEX',T68,'ITAR'/
11H ,1I10,3I20)
111 FORMAT(1H0,T8,'V'/1H ,1P6D20.11/1H ,1P3D20.11)
112 FORMAT(1H0,T8,'X',T68,'F',T88,'FG',T108,'TESTR'/1H ,1P6D20.11)
113 FORMAT(1H ,T8,'TSI',T68,'G'/1H ,1P6D20.11)
114 FORMAT(1H ,T8,'LT'/1H ,1P6D20.11/1H ,1P3D20.11)
115 FORMAT(1H ,T8,'GG'/1H ,1P3D20.11)
116 FORMAT(1H1,T8,'ITERD',T28,'DX'/1H ,1I10,10X,1P3D20.11)
117 FORMAT(1H ,T8,'XS',T68,'KV'/1H ,1P4D20.11)
118 FORMAT(1H0,T8,'CONSTRAINT RESTROATION')
119 FORMAT(1H0,T8,'DX'/1H ,1P3D20.11)
120 FORMAT(1H ,T8,'KDX',T28,'DELX'/1H ,1P4D20.11)
121 FORMAT(1H ,T8,'FS',T28,'TESTRS',T48,'FGS',T68,'TSIS'/1H ,
11P6D20.11)
122 FORMAT(1H ,T8,'GS',T68,'GGS'/1H ,1P6D20.11)
123 FORMAT(1H ,T8,'P'/1H ,1PD20.11)
124 FORMAT(1H0,T8,'AYM',T28,'AZM',T48,'ATARD'/1H ,1P3D20.11)
125 FORMAT(1H0,T8,'INPUT TARGET'/1H0,T8,'RSVTAR',T68,'VSVTAR'/
11H ,1P6D20.11)
126 FORMAT(1H0,T8,'-----')
127 FORMAT(1H0,T8,'NO. OF DAVIDON ITERATIONS HAS REACHED MAXIMUM')
RETURN
END

```

```

C PROGRAM ETP213
C PROGRAM COMPUTES FUEL OPTIMAL 2-IMPULSE TRANSFER FROM A PARKING
C ORBIT OF GIVEN INCLINATION TO A GIVEN FINAL POSITION AND VELOCITY
C IN FIXED TIME IN EARTH-MOON-VEHICLE SPACE.
  IMPLICIT REAL*8(A-H,K-M,O-Z)
  DIMENSION REMO(3),VEMO(3),XD(3),X(3),XS(3),REVTAR(3),VEVTAR(3),
1V(3,3),GS(3),TSIS(3),LTS(3,3),LS(3,3),S11S(3,3),S12S(3,3),
2S21S(3,3),GGS(3),VSAV(3,3),G(3),TSI(3),LT(3,3),L(3,3),GG(3),
3S11(3,3),S12(3,3),S21(3,3),S22(3,3),DX(3),DELX(3),UVI(3),UVF(3),
4TEMP(3,3),DUM(3),UVID(3),S22S(3,3),PVO(6),GGD(3)
  DIMENSION R(3),REVO(3),VEVO(3),VEVOP(3),UVIS(3),UVFS(3),UVFD(3),
1GD(3),TSID(3),LTD(3,3),LD(3,3),S11D(3,3),S12D(3,3),S21D(3,3),
2S22D(3,3),REVF(3),VEVF(3),REMF(3),VEMF(3),DUVI(3)
  COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
  COMMON/CONST3/ME,MM,GAMMA,UDM,UTIME,UVELM,ERRMAX,DTR
  COMMON/TARG/AY,AZ,ATAR,REVTAR,VEVTAR
  READ(5,100)GAMMA,UDM,UTIME,UVELM
  READ(5,100)ME,MM
  READ(5,100)REVMAG,OINCD,ERRMXM
  READ(5,100)TDAYO,TTRIPD
  READ(5,100)VEVMAG,LOND,THED
  READ(5,100)REMO,VEMO
  READ(5,100)EPS,EPSTS1,KDX,EPSV
  READ(5,101)ICOMV,ITERMX,IFILEX,ITAR
  IF(ITAR.GT.0) GO TO 21
  READ(5,100)AYM,AZM,ATARD
  GO TO 22
21 READ(5,100)REVTAR,VEVTAR
22 IF(ICOMV.GT.0) GO TO 19
  OPTION TO INPUT VARIANCE MATRIX (ICOMV=0)
  READ(5,100) ((V(I,J),J=1,3),I=1,3)
  GO TO 1
19 DO 20 I=1,3
  DO 20 J=1,3
  V(I,J)=0.
  IF(I.EQ.J) V(I,J)=1.0
20 CONTINUE
  1 IPV=0
  IPVTM=0
  IPTRAJ=0
  IFILE=0
  IMTX=1
  ITER=0
  ITERD=0
  DTR=1.7453292519943296D-2
  ERRMAX=ERRMXM/UDM
  IF(ITAR.GT.0) GO TO 23
  AY=AYM/AUM
  AZ=AZM/AUM
  ATAR=DTR*ATARD
23 TDAYF=TDAYO+TTRIPD
  TTRIP=TTRIPD/UTIME
  TSTART=TDAYO/UTIME
  TEND=TDAYF/UTIME
  ALPHA=1.0D-3
  BETA=10.
  ALPHAM=ALPHA/(1.-ALPHA)

```

```

BETAM=BETA/(BETA-1.)
WRITE(6,102)
WRITE(6,103)GAMMA,UDM,UTIME,UVELM,ERRMXM,ERRMAX
WRITE(6,104)ME,MM,OINCD
WRITE(6,105)TDAYO,TTRIPD,TDAYF,TSTART,TTRIP,TEND
WRITE(6,106)REMO,VEMO
WRITE(6,108)VEVMAG,LOND,THED,REVMAG
WRITE(6,109)ALPHA,BETA,EPS,EPSTSI,KDX,EPSTSI
WRITE(6,110)ICOMV,ITERMX,IFILEX,ITAR
IF(ITAR.GT.0) GO TO 24
WRITE(6,124)AYM,AZM,ATARD
GO TO 25
24 WRITE(6,125)REVTAR,VEVTAR
25 LON=DTR*LOND
THE=DTR*THED
OINC=DTR*OINCD
XS(1)=VEVMAG
XS(2)=LON
XS(3)=THE
C COMPUTE INITIAL NOMINAL TRAJECTORY
CALL COMFG3(TSTART,TEND,REVMAG,XS,OINC,REMO,VEMO,FS,TESTRS,GS,
1TSIS,LTS,S11S,S12S,S21S,S22S,UVIS,UVFS)
ITER=1
CALL COMAUG(FS,GS,TSIS,LTS,V,LS,FGS,GG)
C IF(ICOMV.EQ.0) GO TO 200
OPTION TO COMPUTE TRIAL VARIANCE MATRIX (ICOMV=1)
GGSMAG=VMAG(GGS)
DO 2 I=1,3
DO 2 J=1,3
V(I,J)=V(I,J)*EPSTSI/GGSMAG
2 CONTINUE
200 WRITE(6,111) ((V(I,J),J=1,3),I=1,3)
C SAVE VARIANCE MATRIX
DO 3 I=1,3
DO 3 J=1,3
3 VSAV(I,J)=V(I,J)
KV=1.
KDXSAV=KDX
IF(TESTRS.GT.EPSTSI)GO TO 8
C UPDATE VARIABLES
CALL UPX(XS,FS,TESTRS,GS,TSIS,LTS,LS,FGS,GG,S11S,S12S,S21S,
1S22S,UVIS,UVFS,X,F,TESTR,G,TSI,LT,L,FG,GG,S11,S12,S21,S22,
2UVI,UVF)
WRITE(6,112)X,F,FG,TESTR
WRITE(6,113)TSI,G
WRITE(6,114)LT
WRITE(6,115)GG
C ACCELERATED GRADIENT PROJECTION ITERATION LOOP
4 ITERD=ITERD+1
5 DO 501 I=1,3
DO 501 J=1,3
501 V(I,J)=KV*VSAV(I,J)
6 CALL MXV(V,GG,R,3,3)
P=DOT(GG,R,3)
IF((P-2.*FG).LE.EPS) GO TO 7
LAM=2.*FG/P
CALL VVT(R,TEMP,3)

```

```

DO 601 I=1,3
DO 601 J=1,3
601 VSAV(I,J)=V(I,J)+(LAM-1.)*TEMP(I,J)/P
GO TO 5
7 CALL MXV(V,GG,DX,3,3)
C FORM TRIAL NEW INDEPENDENT VARIABLES
DO 701 I=1,3
701 XS(I)=X(I)-DX(I)
WRITE(6,116)ITERD,DX
WRITE(6,117)XS,KV
WRITE(6,111)((V(I,J),J=1,3),I=1,3)
CALL COMFG3(TSTART,TEND,REVMAG,XS,OINC,REMO,VEMO,FS,TESTRS,GS,
1TSIS,LTS,S11S,S12S,S21S,S22S,UVIS,UVFS)
CALL COMAUG(FS,GS,TSIS,LTS,V,LS,FGS,GGS)
IF(TESTRS.LT.EPSTSI) GO TO 11
C CONSTRAINT RESTORATION
8 WRITE(6,118)
801 CALL COMDX(LTS,LS,TSIS,DX)
WRITE(6,119)DX
9 DO 901 I=1,3
DELX(I)=KDX*DX(I)
901 XD(I)=XS(I)+DELX(I)
WRITE(6,120)KDX,DELX
CALL COMFG3(TSTART,TEND,REVMAG,XD,OINC,REMO,VEMO,FD,TESTRD,GD,
1TSID,LTD,S11D,S12D,S21D,S22D,UVID,UVFD)
IF(TESTRD.LT.TESTRS) GO TO 10
KDX=KDX/10.
GO TO 9
10 CALL COMAUG(FD,GD,TSID,LTD,V,LD,FGD,GGD)
CALL UPX(XD,FD,TESTRD,GD,TSID,LTD,LD,FGD,GGD,S11D,S12D,S21D,
1S22D,UVID,UVFD,XS,FS,TESTRS,GS,TSIS,LTS,LS,FGS,GGS,S11S,S12S,
2S21S,S22S,UVIS,UVFS)
KDX=2.*KDX
IF(KDX.GT.1.) KDX=1.
IF(TESTRS.GT.EPSTSI) GO TO 801
KDX=KDXSAV
IF(ITER.EQ.0) GO TO 11
IF(FS.LT.F) GO TO 11
KV=KV/2.
GO TO 5
C CONSTRAINT RESTORED
11 IF(ITERD.LT.ITERMX)GO TO 300
WRITE(6,126)
GO TO 17
300 IF(ITERD.EQ.0)GO TO 16
C UPDATE VARIANCE MATRIX
CALL MXV(V,GG,R,3,3)
P=DOT(GGS,R,3)
WRITE(6,121)FS,TESTRS,FGS,TSIS
WRITE(6,122)GS,GGS
WRITE(6,123)P
IF(P.LT.EPS) GO TO 17
GAMMA=-DOT(GG,R,3)/P
IF(GAMMA.EQ.(-1.)) GO TO 12
IF(GAMMA.GE.ALPHAM)GO TO 13
LAM=ALPHA
GO TO 14

```

```

12 IF(GAMMA.LE.(-BETAM)) GO TO 13
   LAM=BETA
   GO TO 14
13 LAM=GAMMA/(GAMMA+1.)
14 CALL VVT(R,TEMP,3)
   DO 15 I=1,3
   DO 15 J=1,3
15 V(I,J)=V(I,J)+(LAM-1.)*TEMP(I,J)/P
   IF(FGS.LT.FG) GO TO 16
   GO TO 6
16 KV=1.
   CALL UPX(XS,FS,TESTRS,GS,TSIS,LTS,LS,FGS,GG,S11S,S12S,S21S,
1S22S,UVIS,UVFS,X,F,TESTR,G,TSI,LT,L,FG,GG,S11,S12,S21,S22,
2UVI,UVF)
   IF(ITERD.EQ.0) GO TO 4
   DO 161 I=1,3
   DO 161 J=1,3
161 VSAV(I,J)=V(I,J)
   GO TO 4
17 CALL UPX(XS,FS,TESTRS,GS,TSIS,LTS,LS,FGS,GG,S11S,S12S,S21S,
1S22S,UVIS,UVFS,X,F,TESTR,G,TSI,LT,L,FG,GG,S11,S12,S21,S22,
2 UVI,UVF)
C   COMPUTE PRIMER VECTOR DERIVATIVE
   CALL MXV(S11,UVI,DUM,3,3)
   CALL INVERT(S12,TEMP)
   DO 18 I=1,3
18 DUM(I)=UVF(I)-DUM(I)
   CALL MXV(TEMP,DUM,DUVI,3,3)
   DO 400 I=1,3
   PVO(I)=UVI(I)
400 PVO(I+3)=DUVI(I)
   IPV=1
   IPVTM=1
   IPTRAJ=1
   IFILE=IFILEX
C   GENERATE OPTIMAL TRAJECTORY AND PRIMER VECTOR HISTORY
   VEV MAG=X(1)
   LON=X(2)
   THE=X(3)
   CALL COMIC3(REVMAG,VEVMAG,LON,THE,OINC,REVO,VEVO,VEVOP)
   CALL THRBODY(TSTART,TEND,REVO,VEVOP,REMO,VEMO,PVO,REVF,VEVF,
1REMF,VEMF,S11,S12,S21,S22)
100 FORMAT(4D20.11)
101 FORMAT(4I5)
102 FORMAT(1H ,T8,'3-BODY 2-IMPULSE OPTIMAL TRANSFER FROM EARTH'//
11H ,T8,'TO HALO ORBIT AT L1 OR L2. ITERATE ON VELOCITY'//
21H ,T8,'MAGNITUDE, LONGITUDE OF NODE AND ORBITAL ANGLE FROM'//
31H ,T8,'NODE USING ACCELERATED GRADIENT PROJECTION AND '//
41H ,T8,'DAVIDON VARIANCE METHOD')
103 FORMAT(1H0,T8,'GAMMA',T28,'UDM',T48,'UTIME',T68,'UVELM',T88,
1'ERRMXM',T108,'ERRMAX'/1H ,1P6D20.11)
104 FORMAT(1H ,T8,'ME',T28,'MM',T48,'OINCD'//
11H ,1P5D20.11)
105 FORMAT(1H ,T8,'TDAY0',T28,'TTRIPD',T48,'TDAYF',T68,'TSTART',
1T88,'TTRIP',T108,'TEND'/1H ,1P6D20.11)
106 FORMAT(1H ,T8,'REMO',T68,'VEMO'/1H ,1P6D20.11)
108 FORMAT(1H ,T8,'VEVMAG',T28,'LOND',T48,'THED',T68,'REVMAG'/

```

```

11H ,1P4D20.11)
109 FORMAT(1H ,T8,'ALPHA',T28,'BETA',T48,'EPS',T68,'EPSTSI',T88,
1'KDX',T108,'EPSV'/1H ,1P6D20.11)
10 FORMAT(1H ,T8,'ICOMV',T28,'ITERMX',T48,'IFILEX',T68,'ITAR'/
11H ,I10,3I20)
111 FORMAT(1H0,T8,'V'/1H ,1P6D20.11/1H ,1P3D20.11)
112 FORMAT(1H0,T8,'X',T68,'F',T88,'FG',T108,'TESTR'/1H ,1P6D20.11)
113 FORMAT(1H ,T8,'TSI',T68,'G'/1H ,1P6D20.11)
114 FORMAT(1H ,T8,'LT'/1H ,1P6D20.11/1H ,1P3D20.11)
115 FORMAT(1H ,T8,'GG'/1H ,1P3D20.11)
116 FORMAT(1H1,T8,'ITERD',T28,'DX'/1H ,I10,10X,1P3D20.11)
117 FORMAT(1H ,T8,'XS',T68,'KV'/1H ,1P4D20.11)
118 FORMAT(1H0,T8,'CONSTRAINT RESTROATION')
119 FORMAT(1H0,T8,'DX'/1H ,1P3D20.11)
120 FORMAT(1H ,T8,'KDX',T28,'DELX'/1H ,1P4D20.11)
121 FORMAT(1H ,T8,'FS',T28,'TESTRS',T48,'FGS',T68,'TSIS'/1H ,
11P6D20.11)
122 FORMAT(1H ,T8,'GS',T68,'GGS'/1H ,1P6D20.11)
123 FORMAT(1H ,T8,'P'/1H ,1PD20.11)
124 FORMAT(1H0,T8,'AYM',T28,'AZM',T48,'ATARD'/1H ,1P3D20.11)
125 FORMAT(1H0,T8,'INPUT TARGET'/1H0,T8,'REVTAR',T68,'VEVTAR'/1H ,
11P6D20.11)
126 FORMAT(1H0,T8,'NO. OF DAVIDON ITERATIONS HAS REACHED MAXIMUM')
RETURN
END

```

```

C PROGRAM PTP31
C PROGRAM COMPUTES FUEL OPTIMAL 3-IMPULSE TRANSFER FROM A GIVEN
C INITIAL POSITION TO A GIVEN FINAL POSITION AND VELOCITY IN
C FIXED TIME.
  IMPLICIT REAL*8(A-H,K-M,O-Z)
  DIMENSION X(4),G(4),V(4,4),XS(4),GS(4),R(4),DX(4),RSVO(3),
1VSV0(3),RSE0(3),VSE0(3),RSM0(3),VSM0(3),VSVI(3),
2TEMP1(4,4),SFM11S(3,3),SFM12S(3,3),SFM21S(3,3),SFM22S(3,3),
3SMI11S(3,3),SMI12S(3,3),SMI21S(3,3),SMI22S(3,3),RSVM(3),
4VSVMP(3),RSVMS(3),VSVMP(3),DUVIS(3),DUVMS(3),
5DUVMP(3),DG(4),VS(4,4),DXC(4),XC(4),RSVMC(3),VSVMPC(3),
6UVIC(3),UVMC(3),UVFC(3),DUVIC(3),DUVMC(3),DUVMPC(3),RC(4),
7SMI11C(3,3),SMI12C(3,3),SMI21C(3,3),SMI22C(3,3),SFM11C(3,3),
8SFM12C(3,3),SFM21C(3,3),SFM22C(3,3),RSVMD(3),VSVMPD(3),UVIS(3),
9UVMS(3),UVFS(3),S(4),SFM11(3,3),SFM12(3,3),SFM21(3,3),
1SFM22(3,3),UVI(3),UVM(3),UVF(3),DUVI(3),DUVMM(3),DUVMP(3),
2SMI11(3,3),SMI12(3,3),SMI21(3,3),SMI22(3,3),PVO(6),PVM(6),
3RSVTAR(3),VSVTAR(3),SFM11D(3,3),SFM12D(3,3)
  DIMENSION VSVMM(3),RSEM(3),VSEM(3),RSMM(3),VSMM(3),
1RSVF(3),VSVF(3),RSEF(3),VSEF(3),RSMF(3),VSMF(3)
  DIMENSION DGC(4),VC(4,4),GC(4),VSVMMS(3),VSVMMC(3)
  COMMON/CONST/MS,ME,MM,GL1,AUM,UTIME,UVELM,ERRMAX,DTR
  COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
  READ(5,100)GL1,AUM,UTIME,UVELM
  READ(5,100)MS,ME,MM
  READ(5,100)TSTART,TM,TEND
  READ(5,100)RSE0,VSE0,RSM0,VSM0
  READ(5,100)RSVO,VSVO
  READ(5,100)VSVI,VSVMP
  READ(5,100)RSVTAR,VSVTAR
  READ(5,100)KNR,ERRMIN,ERRMXM
  READ(5,100)FMINM,EPS,EPSV
  READ(5,101)ICOMV,ITLMAX,ILINC,ITDMAX,IFILEX
  IF(ICOMV.GT.0)GO TO 190
C OPTION TO INPUT VARIANCE MATRIX(ICOMV=0)
  READ(5,100) ((V(I,J),J=1,4),I=1,4)
  GO TO 1
190 DO 200 I=1,4
  DO 200 J=1,4
  V(I,J)=0.
  IF(I.EQ.J) V(I,J)=1.0
200 CONTINUE
1 DTR=1.7453292519943296D-2
  ERRMAX=ERRMXM/AUM
  FMIN=FMINM*UVELM
  KNRSV=KNR
  ALPMIN=1.0D-8
  WRITE(6,102)GL1,AUM,UTIME,UVELM,ERRMXM,ERRMAX,MS,ME,MM
  WRITE(6,103)TSTART,TM,TEND,RSVO,VSVO,RSE0,VSE0,RSM0,VSM0
  WRITE(6,104)VSVI,VSVMP,RSVTAR,VSVTAR
  WRITE(6,105)KNR,ERRMIN,FMINM,EPS,EPSV
  WRITE(6,106)ICOMV,ITLMAX,ILINC,ITDMAX,IFILEX
  IPV=0
  IPVTM=0
  IFILE=0
  IPTRAJ=0
  IMTX=1

```



```

ITERD=0
DO 2 I=1,3
2 X(I)=VSVI(I)
X(4)=TM
C COMPUTE INITIAL NOMINAL TRAJECTORY
CALL COMF(TSTART, TM, TEND, RSV0, VSVO, VSVI, RSEO, VSEO, RSMO, VSMO,
1ERRMIN, ILINC, KNRSV, ITERD, ITLMAX, RSVMD, VSVMPD, RSVTAR, VSVTAR,
2SFM11D, SFM12D, RSVM, VSVMM, VSVMP, F, UVI, UVM, UVF, SMI11,
3SMI12, SMI21, SMI22, SFM11, SFM12, SFM21, SFM22)
CALL COMG(SMI11, SMI12, SMI21, SMI22, SFM11, SFM12, VSVMP, VSVMM, UVI,
1UVM, UVF, DUVI, DUVMM, DUVMP, G)
GMAG=DSQRT(DOT(G, G, 4))
WRITE(6, 108) F, G, GMAG
DO 3 I=1,3
RSVMD(I)=RSVM(I)
VSVMPD(I)=VSVMP(I)
DO 3 J=1,3
SFM11D(I, J)=SFM11(I, J)
3 SFM12D(I, J)=SFM12(I, J)
IF(ICOMV.EQ.0) GO TO 40
C OPTION TO COMPUTE TRIAL VARIANCE MATRIX (ICOMV=1)
DO 4 I=1,4
DO 4 J=1,4
V(I, J)=V(I, J)*EPSV/GMAG
4 CONTINUE
40 WRITE(6, 107) ((V(I, J), J=1, 4), I=1, 4)
C ACCELERATED GRADIENT ITERATION LOOP
5 ITERD=ITERD+1
WRITE(6, 109) ITERD
CALL MXV(V, G, S, 4, 4)
DO 50 I=1,4
50 S(I)=-S(I)
SG=DOT(S, G, 4)
WRITE(6, 110) S, SG
IF(SG.LT.0.) GO TO 51
DO 52 I=1,4
52 S(I)=-S(I)
SG=-SG
WRITE(6, 111) S, SG
51 ALP=(FMIN-F)/SG
ALPHA=DMIN1(1.00+0, ALP)
DO 61 I=1,4
DX(I)=ALPHA*S(I)
61 XS(I)=X(I)+DX(I)
DO 62 I=1,3
62 VSVI(I)=XS(I)
WRITE(6, 112) DX, ALPHA, XS
CALL COMF(TSTART, XS(4), TEND, RSV0, VSVO, VSVI, RSEO, VSEO, RSMO, VSMO,
1ERRMIN, ILINC, KNRSV, ITERD, ITLMAX, RSVMD, VSVMPD, RSVTAR, VSVTAR,
2SFM11D, SFM12D, RSVM, VSVMM, VSVMP, F, UVI, UVM, UVF, SMI11S,
3SMI12S, SMI21S, SMI22S, SFM11S, SFM12S, SFM21S, SFM22S)
CALL COMG(SMI11S, SMI12S, SMI21S, SMI22S, SFM11S, SFM12S, VSVMP,
1VSVMM, UVI, UVM, UVF, DUVI, DUVMM, DUVMP, GS)
GSMAG=DSQRT(DOT(GS, GS, 4))
WRITE(6, 113) FS, GS, GSMAG
DO 701 I=1,4
701 DG(I)=GS(I)-G(I)

```

```

CALL MXV(V,DG,R,4,4)
DO 71 I=1,4
71 R(I)=R(I)-DX(I)
   RMAG=DSQRT(DOT(R,R,4))
   P=DOT(DG,R,4)
   PMAG=DABS(P)
   SGS=DOT(S,GS,4)
   WRITE(6,114)DG,P,SGS,R,RMAG
8 CALL VVT(R,TEMP1,4)
DO 81 I=1,4
DO 81 J=1,4
81 V(I,J)=V(I,J)-TEMP1(I,J)/P
82 IF(SGS.GT.0.)GO TO 83
   IF(FS.LT.F)GO TO 14
   ALPHA=ALPHA/10.0
   IF(ALPHA.LT.ALPMIN)GO TO 84
   GO TO 6
84 DO 840 I=1,4
DO 840 J=1,4
V(I,J)=0.
IF(I.EQ.J)V(I,J)=1.
840 V(I,J)=V(I,J)*EPSV/GMAG
WRITE(6,122)
GO TO 40
83 CONTINUE
C CUBIC INTERPOLATION BETWEEN X AND XS
Z=3.*(F-FS)/ALPHA+SG+SGS
W2=Z*Z-SG*SGS
IF(W2.LE.0.)GO TO 13
W=DSQRT(W2)
ALPHAC=ALPHA*(1.-(SGS+W-Z)/(SGS-SG+2.*W))
DO 9 I=1,4
DXC(I)=ALPHAC*S(I)
9 XC(I)=X(I)+DXC(I)
DO 901 I=1,3
901 VSVI(I)=XC(I)
WRITE(6,115)DXC,ALPHAC,XC
CALL COMF(TSTART,XC(4),TEND,RSVO,VSVO,VSVI,RSEO,VSEO,RSMO,VSMO,
IERRMIN,ILINC,KNRSV,ITERD,ITLMAX,RSVMD,VSVMPD,RSVTAR,VSVTAR,
2SFM11D,SFM12D,RSVMC,VSVMMC,VSVMPD,FC,UVIC,UVMC,UVFC,SMI11C,
3SMI12C,SMI21C,SMI22C,SFM11C,SFM12C,SFM21C,SFM22C)
CALL COMG(SMI11C,SMI12C,SMI21C,SMI22C,SFM11C,SFM12C,VSVMPD,
1VSVMMC,UVIC,UVMC,UVFC,DUVIC,DUVMMC,DUVMPC,GC)
GCMAG=DSQRT(DOT(GC,GC,4))
WRITE(6,116)FC,GC,GCMAG
DO 902 I=1,4
902 DGC(I)=GC(I)-G(I)
DO 91 I=1,3
RSVMD(I)=RSVMC(I)
VSVMPD(I)=VSVMPD(I)
DO 91 J=1,3
SFM11D(I,J)=SFM11C(I,J)
91 SFM12D(I,J)=SFM12C(I,J)
CALL MXV(V,GC,RC,4,4)
DO 92 I=1,4
92 RC(I)=RC(I)-DXC(I)
RCMAG=DSQRT(DOT(RC,RC,4))

```

```

PC=DOT(DGC,RC,4)
CALL VVT(RC,TEMP1,4)
DO 93 I=1,4
DO 93 J=1,4
93 V(I,J)=V(I,J)-TEMP1(I,J)/PC
WRITE(6,117)DGC,PC,RC,RCMAG
IF(FS.GT.F)GO TO 11
IF(FC.GT.FS)GO TO 14
10 DO 20 I=1,4
X(I)=XC(I)
20 G(I)=GC(I)
DO 201 I=1,3
UVI(I)=UVIC(I)
UVM(I)=UVMC(I)
DUVI(I)=DUVIC(I)
DUVMM(I)=DUVMC(I)
DUVMP(I)=DUVMPC(I)
RSVMD(I)=RSVMC(I)
VSVMPD(I)=VSVMP(I)
DO 201 J=1,3
SFM11D(I,J)=SFM11C(I,J)
201 SFM12D(I,J)=SFM12C(I,J)
F=FC
GMAG=GCMAG
GO TO 16
11 IF(FC.LT.F)GO TO 10
C FC GRTHN F. REPEAT INTERPOLATION IN REDUCED INTERVAL.
FS=FC
ALPHA=ALPHAC
DO 250 I=1,3
XS(I)=XC(I)
GS(I)=GC(I)
RSVMS(I)=RSVMC(I)
VSVMPS(I)=VSVMP(I)
UVIS(I)=UVIC(I)
UVMS(I)=UVMC(I)
DUVIS(I)=DUVIC(I)
DUVMMS(I)=DUVMC(I)
DUVMPS(I)=DUVMPC(I)
DO 250 J=1,3
SFM11S(I,J)=SFM11C(I,J)
250 SFM12S(I,J)=SFM12C(I,J)
SGS=DOT(S,GS,4)
WRITE(6,121)SGS
GO TO 82
13 IF(FS.GT.F)GO TO 15
14 F=FS
GMAG=GSMAG
DO 140 I=1,4
X(I)=XS(I)
140 G(I)=GS(I)
DO 141 I=1,3
UVI(I)=UVIS(I)
UVM(I)=UVMS(I)
DUVI(I)=DUVIS(I)
DUVMM(I)=DUVMMS(I)
DUVMP(I)=DUVMPS(I)

```

```

RSVMD(I)=RSVMS(I)
VSVMPD(I)=VSVMP(S(I)
DO 141 J=1,3
SFM11D(I,J)=SFM11S(I,J)
41 SFM12D(I,J)=SFM12S(I,J)
15 CONTINUE
16 DO 161 I=1,3
RSVM(I)=RSVMD(I)
VSVMP(I)=VSVMPD(I)
DO 161 J=1,3
SFM11(I,J)=SFM11D(I,J)
161 SFM12(I,J)=SFM12D(I,J)
WRITE(6,118)X,F,GMAG,G,RSVMD,VSVMPD
WRITE(6,107)((V(I,J),J=1,4),I=1,4)
IF(GMAG.LT.EPS)GO TO 18
IF(ITERD.LT.ITDMAX)GO TO 5
WRITE(6,119)
18 IPV=1
IPTRAJ=1
IFILE=IFILEX
WRITE(6,120)
C GENERATE OPTIMAL TRAJECTORY AND PRIMER VECTOR HISTORY
DO 30 I=1,3
PVO(I)=UVI(I)
PVO(I+3)=DUVI(I)
PVM(I)=UVM(I)
PVM(I+3)=DUVMP(I)
30 VSVI(I)=X(I)
CALL FOURBY(TSTART,X(4),RSVO,VSVI,RSEO,VSEO,
1RSMO,VSMO,PVO,RSVM,VSVM,RSEM,VSEM,RSMM,VSMM,SMI11,SMI12,
2SMI21,SMI22)
CALL FOURBY(X(4),TEND,RSVM,VSVMP,RSEM,VSEM,RSMM,VSMM,
1PVM,RSVF,VSVF,RSEF,VSEF,RSMF,VS MF,SFM11,SFM12,SFM21,SFM22)
100 FORMAT(4D20.11)
101 FORMAT(6I5)
102 FORMAT(1H0,40X,'4-BODY 3-IMPULSE BOUNDARY VALUE PROBLEM'/
11H0,T8,'GL1',T28,'AUM',T48,'UTIME',T68,'UVELM',T88,'ERRMXM',
2T108,'ERRMAX'/1H,1P6D20.11/1H,T8,'MS',T28,'ME',T48,'MM'/1H,
31P3D20.11)
103 FORMAT(1H,T8,'TSTART',T28,'TM',T48,'TEND'/1H,1P3D20.11/1H,
1T8,'RSVO',T68,'VSVO'/1H,1P6D20.11/1H,T8,'RSEO',T68,'VSEO'/
21H,1P6D20.11/1H,T8,'RSMO',T68,'VSMO'/1H,1P6D20.11)
104 FORMAT(1H0,T8,'VSVI',T68,'VSVMP'/1H,1P6D20.11/1H,T8,'RSVTAR',
1T68,'SVTAR'/1H,1P6D20.11)
105 FORMAT(1H0,T8,'KNR',T28,'ERRMIN',T48,'FMINM',T68,'EPS',T88,
1'EPSV'/1H,1P5D20.11)
106 FORMAT(1H0,T8,'ICOMV',T28,'ITLMAX',T48,'ILINC',T68,'ITDMAX',T88,
1'IFILEX'/1H,110,4120)
107 FORMAT(1H0,T8,'V'/1H,1P6D20.11/1H,1P6D20.11/1H,1P4D20.11)
108 FORMAT(1H,T8,'F',T28,'G',T108,'GMAG'/1H,1P6D20.11)
109 FORMAT(1H1,T8,'ITERD'/1H,110)
110 FORMAT(1H0,T8,'S',T88,'SG'/1H,1P5D20.11)
111 FORMAT(1H0,T8,'SIGNS OF S AND SG REVERSED'/1H,T8,'S',T88,'SG'/
11H,1P5D20.11)
112 FORMAT(1H0,T8,'DX',T88,'ALPHA'/1H,1P5D20.11/1H,T8,'XS'/1H,
11H,1P4D20.11)
113 FORMAT(1H,T8,'FS',T28,'GS',T108,'GSMAG'/1H,1P6D20.11)

```

```
114 FORMAT(1H ,T8,'DG',T88,'P',T108,'SGS'/1H ,1P6D20.11/1H ,T8,  
1'R',T88,'RMAG'/1H ,1P5D20.11)  
115 FORMAT(1H0,T8,'CUBIC INTERPOLATION BETWEEN X AND XS'/1H0,T8,  
1'DXC',T88,'ALPHAC'/1H ,1P5D20.11/1H ,T8,'XC'/1H ,1P4D20.11)  
116 FORMAT(1H ,T8,'FC',T28,'GC',T108,'GCMAG'/1H ,1P6D20.11)  
117 FORMAT(1H ,T8,'DGC',T88,'PC'/1H ,1P5D20.11/1H ,T8,'RC',T88,  
1'RCMAG'/1H ,1P5D20.11)  
118 FORMAT(1H0,T8,'OUTPUT OF THIS ITERATION'/1H0,T8,'X',T88,'F',  
1T108,'GMAG'/1H ,1P6D20.11/1H ,T8,'G'/1H ,1P4D20.11/1H ,T8,  
2'RSVMD',T68,'VSMPD'/1H ,1P6D20.11)  
119 FORMAT(1H0,T8,'NO. OF DAVIDON ITERATIONS HAS REACHED MAXIMUM')  
120 FORMAT(1H1)  
121 FORMAT(1H0,T8,'SGS'/1H ,1P1D20.11)  
122 FORMAT(1H0,T8,'RESTART WITH IDENTITY V MATRIX SCALED BY EPSV')  
RETURN  
END
```

C PROGRAM PTP3I3

C PROGRAM COMPUTES FUEL OPTIMAL 3-IMPULSE TRANSFER FROM A GIVEN  
C INITIAL POSITION TO A GIVEN FINAL POSITION AND VELOCITY IN  
Q FIXED TIME.

IMPLICIT REAL\*8(A-H,K-M,O-Z)

DIMENSION X(4),G(4),V(4,4),XS(4),GS(4),R(4),DX(4),REVO(3),  
1VEVO(3),REMO(3),VEMO(3),VEVOP(3),TEMP1(4,4),SFM11S(3,3),  
2SFM12S(3,3),SFM21S(3,3),SFM22S(3,3),SMI11S(3,3),SMI12S(3,3),  
3SMI21S(3,3),SMI22S(3,3),REVM(3),VEVMP(3),REVMS(3),VEVMPS(3),  
4DUVIS(3),DUVMMS(3),DUVMPS(3),DG(4),VS(4,4),DXC(4),XC(4),  
5REVMC(3),VEVMPC(3),UVIC(3),UVMC(3),UVFC(3),DUVIC(3),DUVMC(3),  
6DUVMPC(3),RC(4),SMI11C(3,3),SMI12C(3,3),SMI21C(3,3),SMI22C(3,3),  
7SFM11C(3,3),SFM12C(3,3),SFM21C(3,3),SFM22C(3,3),REVMD(3),  
8VEVMPD(3),UVIS(3),UVMS(3),UVFS(3),S(4),SFM11(3,3),SFM12(3,3),  
9SFM21(3,3),SFM22(3,3),UVI(3),UVM(3),UVF(3),DUVI(3),DUVMM(3),  
1DUVMP(3),SMI11(3,3),SMI12(3,3),SMI21(3,3),SMI22(3,3),PVO(6),  
2PVM(3),REVTAR(3),VEVTAR(3),SFM11D(3,3),SFM12D(3,3),DGC(4),  
3VC(4,4),GC(4),VEVMM(3),REMM(3),VEMM(3),REVF(3),VEVF(3),REMF(3),  
4VEMF(3),VEVMMS(3),VEVMC(3)

COMMON/CONST3/ME,MM,GAMMA,UDM,UTIME,UVELM,ERRMAX,DTR

COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR

READ(5,100)GAMMA,UDM,UTIME,UVELM

READ(5,100)ME,MM

READ(5,100)TSTART,TM,TEND

READ(5,100)REMO,VEMO

READ(5,100)REVO,VEVO

READ(5,100)VEVOP,VEVMP

READ(5,100)REVTAR,VEVTAR

READ(5,100)KNR,ERRMIN,ERRMXM

READ(5,100)FMINM,EPS,EPSV

READ(5,101)ICOMV,ITLMAX,ILINC,ITDMAX,IFILEX

IF(ICOMV.GT.0)GO TO 190

C OPTION TO INPUT VARIANCE MATRIX(ICOMV=0)

READ(5,100)((V(I,J),J=1,4),I=1,4)

GO TO 1

190 DO 200 I=1,4

DO 200 J=1,4

V(I,J)=0.

IF(I.EQ.J)V(I,J)=1.0

200 CONTINUE

1 DTR=1.7453292519943296D-2

ERRMAX=ERRMXM/UDM

FMIN=FMINM\*UVELM

KNRSV=KNR

ALPMIN=1.0D-8

WRITE(6,102)GAMMA,UDM,UTIME,UVELM,ERRMXM,ERRMAX,ME,MM

WRITE(6,103)TSTART,TM,TEND,REVO,VEVO,REMO,VEMO

WRITE(6,104)VEVOP,VEVMP,REVTAR,VEVTAR

WRITE(6,105)KNR,ERRMIN,FMINM,EPS,EPSV

WRITE(6,106)ICOMV,ITLMAX,ILINC,ITDMAX,IFILEX

IPV=0

IPVTM=0

IFILE=0

IPTRAJ=0

IMTX=1

ITERD=0

DO 2 I=1,3

```

2 X(I)=VEVOP(I)
  X(4)=TM
C COMPUTE INITIAL NOMINAL TRAJECTORY
  CALL COMF3(TSTART, TM, TEND, REVO, VEVO, VEVOP, REMO, VEMO, ERRMIN,
1 I LINC, KNRSV, ITERD, ITLMAX, REVMD, VEVMPD, REV TAR, VEV TAR, SFM11D,
2 SFM12D, REVM, VEVMM, VEVMP, F, UVI, UVM, UVF, SMI11, SMI12, SMI21, SMI22,
3 SFM11, SFM12, SFM21, SFM22)
  CALL COMG(SMI11, SMI12, SMI21, SMI22, SFM11, SFM12, VEVMP, VEVMM, UVI,
1 UVM, UVF, DUVI, DUVMM, DUVMP, G)
  GMAG=DSQRT(DOT(G, G, 4))
  WRITE(6, 108) F, G, GMAG
  DO 3 I=1, 3
    REVMD(I)=REVM(I)
    VEVMPD(I)=VEVMP(I)
  DO 3 J=1, 3
    SFM11D(I, J)=SFM11(I, J)
3 SFM12D(I, J)=SFM12(I, J)
  IF(ICOMV.EQ.0)GO TO 40
C OPTION TO COMPUTE TRIAL VARIANCE MATRIX (ICOMV=1)
  DO 4 I=1, 4
    DO 4 J=1, 4
      V(I, J)=V(I, J)*EPSV/GMAG
4 CONTINUE
40 WRITE(6, 107)((V(I, J), J=1, 4), I=1, 4)
C ACCELERATED GRADIENT ITERATION LOOP
5 ITERD=ITERD+1
  WRITE(6, 109) ITERD
  CALL MXV(V, G, S, 4, 4)
  DO 50 I=1, 4
50 S(I)=-S(I)
  SG=DOT(S, G, 4)
  WRITE(6, 110) S, SG
  IF(SG.LT.0.)GO TO 51
  DO 52 I=1, 4
52 S(I)=-S(I)
  SG=-SG
  WRITE(6, 111) S, SG
51 ALP=(FMIN-F)/SG
  ALPHA=DMIN1(1.0D+0, ALP)
6 DO 61 I=1, 4
  DX(I)=ALPHA*S(I)
61 XS(I)=X(I)+DX(I)
  DO 62 I=1, 3
62 VEVOP(I)=XS(I)
  WRITE(6, 112) DX, ALPHA, XS
  CALL COMF3(TSTART, XS(4), TEND, REVO, VEVO, VEVOP, REMO, VEMO, ERRMIN,
1 I LINC, KNRSV, ITERD, ITLMAX, REVMD, VEVMPD, REV TAR, VEV TAR, SFM11D,
2 SFM12D, REVM, VEVMM, VEVMP, F, UVI, UVM, UVF, SMI11S, SMI12S,
3 SMI21S, SMI22S, SFM11S, SFM12S, SFM21S, SFM22S)
  CALL COMG(SMI11S, SMI12S, SMI21S, SMI22S, SFM11S, SFM12S, VEVMP,
1 VEVMM, UVI, UVM, UVF, DUVI, DUVMM, DUVMP, GS)
  GSMAG=DSQRT(DOT(GS, GS, 4))
  WRITE(6, 113) F, GS, GSMAG
  DO 701 I=1, 4
701 DG(I)=GS(I)-G(I)
  DO 7 I=1, 3
    REVMD(I)=REVM(S(I))

```

```

VEVMPD(I)=VEVMPS(I)
DO 7 J=1,3
SFM11D(I,J)=SFM11S(I,J)
7 SFM12D(I,J)=SFM12S(I,J)
CALL MXV(V,DG,R,4,4)
DO 71 I=1,4
71 R(I)=R(I)-DX(I)
RMAG=DSQRT(DOT(R,R,4))
P=DOT(DG,R,4)
PMAG=DABS(P)
SGS=DOT(S,GS,4)
WRITE(6,114)DG,P,SGS,R,RMAG
8 CALL VVT(R,TEMP1,4)
DO 81 I=1,4
DO 81 J=1,4
81 V(I,J)=V(I,J)-TEMP1(I,J)/P
82 IF(SGS.GT.0.)GO TO 83
IF(FS.LT.F)GO TO 14
ALPHA=ALPHA/10.0
IF(ALPHA.LT.ALPMIN)GO TO 84
GO TO 6
84 DO 840 I=1,4
DO 840 J=1,4
V(I,J)=0.
IF(I.EQ.J)V(I,J)=1.
840 V(I,J)=V(I,J)*EPSV/GMAG
WRITE(6,122)
GO TO 40
83 CONTINUE
CUBIC INTERPOLATION BETWEEN X AND XS
Z=3.*(F-FS)/ALPHA+SG+SGS
W2=Z*Z-SG*SGS
IF(W2.LE.0.)GO TO 13
W=DSQRT(W2)
ALPHAC=ALPHA*(1.-(SGS+W-Z)/(SGS-SG+2.*W))
DO 9 I=1,4
DXC(I)=ALPHAC*S(I)
9 XC(I)=X(I)+DXC(I)
DO 901 I=1,3
901 VEVOP(I)=XC(I)
WRITE(6,115)DXC,ALPHAC,XC
CALL COMF3(TSTART,XC(4),TEND,REVO,VEVO,VEVOP,REMO,VEMO,ERRMIN,
1 IINC,KNRSV,ITERD,ITLMAX,REVMD,VEVMPD,REVTAR,VEVTAR,SFM11D,
2 SFM12D,REVMC,VEVMC,VEVMPC,FC,UVIC,UVMC,UVFC,SMI11C,SMI12C,
3 SMI21C,SMI22C,SFM11C,SFM12C,SFM21C,SFM22C)
CALL COMG(SMI11C,SMI12C,SMI21C,SMI22C,SFM11C,SFM12C,VEVMPC,
1 VEVMMC,UVIC,UVMC,UVFC,DUVIC,DUVMC,DUVMPC,GC)
GCMAG=DSQRT(DOT(GC,GC,4))
WRITE(6,116)FC,GC,GCMAG
DO 902 I=1,4
902 DGC(I)=GC(I)-G(I)
DO 91 I=1,3
REVMC(I)=REVMC(I)
VEVMPD(I)=VEVMPC(I)
DO 91 J=1,3
SFM11D(I,J)=SFM11C(I,J)
91 SFM12D(I,J)=SFM12C(I,J)

```



```

CALL MXV(V,GC,RC,4,4)
DO 92 I=1,4
92 RC(I)=RC(I)-DXC(I)
RCMAG=DSQRT(DOT(RC,RC,4))
PC=DOT(DGC,RC,4)
CALL VVT(RC,TEMP1,4)
DO 93 I=1,4
DO 93 J=1,4
93 V(I,J)=V(I,J)-TEMP1(I,J)/PC
WRITE(6,117)DGC,PC,RC,RCMAG
IF(FS.GT.F)GO TO 11
IF(FC.GT.FS)GO TO 14
10 DO 20 I=1,4
X(I)=XC(I)
20 G(I)=GC(I)
DO 201 I=1,3
UVI(I)=UVIC(I)
UVM(I)=UVMC(I)
DUVI(I)=DUVIC(I)
DUVMM(I)=DUVMC(I)
DUVMP(I)=DUVMPC(I)
REVM(I)=REVMC(I)
VEVMP(I)=VEVMPC(I)
DO 201 J=1,3
SFM11D(I,J)=SFM11C(I,J)
201 SFM12D(I,J)=SFM12C(I,J)
F=FC
GMAG=GCMAG
GO TO 16
11 IF(FC.LT.F)GO TO 10
FC GRTHN F. REPEAT INTERPOLATION IN REDUCED INTERVAL.
FS=FC
ALPHA=ALPHAC
DO 250 I=1,3
XS(I)=XC(I)
GS(I)=GC(I)
REVMS(I)=REVMC(I)
VEVMPS(I)=VEVMPC(I)
UVIS(I)=UVIC(I)
UVMS(I)=UVMC(I)
DUVIS(I)=DUVIC(I)
DUVMMS(I)=DUVMC(I)
DUVMPS(I)=DUVMPC(I)
DO 250 J=1,3
SFM11S(I,J)=SFM11C(I,J)
250 SFM12S(I,J)=SFM12C(I,J)
SGS=DOT(S,GS,4)
WRITE(6,121)SGS
GO TO 82
13 IF(FS.GT.F)GO TO 15
14 F=FS
GMAG=GSMAG
DO 140 I=1,4
X(I)=XS(I)
40 G(I)=GS(I)
DO 141 I=1,3
UVI(I)=UVIS(I)

```

```

UVM(I)=UVMS(I)
DUVI(I)=DUVIS(I)
DUVMM(I)=DUVMMS(I)
DUVMP(I)=DUVMPS(I)
REVMD(I)=REVMS(I)
VEVMPD(I)=VEVMPS(I)
DO 141 J=1,3
SFM11D(I,J)=SFM11S(I,J)
141 SFM12D(I,J)=SFM12S(I,J)
15 CONTINUE
16 DO 161 I=1,3
REVM(I)=REVMD(I)
VEVMP(I)=VEVMPD(I)
DO 161 J=1,3
SFM11(I,J)=SFM11D(I,J)
161 SFM12(I,J)=SFM12D(I,J)
WRITE(6,118)X,F,G,GMAG,REVMD,VEVMPD
WRITE(6,107)((V(I,J),J=1,4),I=1,4)
IF(GMAG.LT.EPS)GO TO 18
IF(ITERD.LT.ITDMAX)GO TO 5
WRITE(6,119)
18 IPV=1
IPTRAJ=1
IFILE=IFILEX
WRITE(6,120)
GENERATE OPTIMAL TRAJECTORY AND PRIMER VECTOR HISTORY
DO 30 I=1,3
PVO(I)=UVI(I)
PVO(I+3)=DUVI(I)
PVM(I)=UVM(I)
PVM(I+3)=DUVMP(I)
30 VEVOP(I)=X(I)
CALL THRBDY(TSTART,X(4),REVO,VEVOP,REMO,VEMO,PVO,REVM,VEVMM,
1REMM,VEMM,SMI11,SMI12,SMI21,SMI22)
CALL THRBDY(X(4),TEND,REVM,VEVMP,REMM,VEMM,PVM,REVF,VEVF,REMF,
1VEMF,SFM11,SFM12,SFM21,SFM22)
100 FORMAT(4D20.11)
101 FORMAT(6I5)
102 FORMAT(1H0,40X,'3-BODY 3-IMPULSE BOUNDARY VALUE PROBLEM'/
11H0,T8,'GAMMA',T28,'UDM',T48,'UTIME',T68,'UVELM',T88,'ERRMXM',
2T108,'ERRMAX'/1H,1P6D20.11/1H,T8,'ME',T28,'MM'/1H,1P2D20.11)
103 FORMAT(1H,T8,'TSTART',T28,'TM',T48,'TEND'/1H,1P3D20.11/1H,
1T8,'REVO',T68,'VEVO'/1H,1P6D20.11/1H,T8,'REMO',T68,'VEMO'/
21H,1P6D20.11)
104 FORMAT(1H0,T8,'VEVOP',T68,'VEVMP'/1H,1P6D20.11/1H,T8,'REVTAR',
1T68,'VEVTAR'/1H,1P6D20.11)
105 FORMAT(1H0,T8,'KNR',T28,'ERRMIN',T48,'FMINM',T68,'EPS',T88,
1'EPSV'/1H,1P5D20.11)
106 FORMAT(1H0,T8,'ICOMV',T28,'ITLMAX',T48,'ILINC',T68,'ITDMAX',T88,
1'IFILEX'/1H,I10,4I20)
107 FORMAT(1H0,T8,'V'/1H,1P6D20.11/1H,1P6D20.11/1H,1P4D20.11)
108 FORMAT(1H,T8,'F',T28,'G',T108,'GMAG'/1H,1P6D20.11)
109 FORMAT(1H1,T8,'ITERD'/1H,I10)
110 FORMAT(1H0,T8,'S',T88,'SG'/1H,1P5D20.11)
111 FORMAT(1H0,T8,'SIGNS OF S AND SG REVERSED'/1H,T8,'S',T88,'SG'/
11H,1P5D20.11)
112 FORMAT(1H0,T8,'DX',T88,'ALPHA'/1H,1P5D20.11/1H,T8,'XS'/1H,

```

```

11H ,1P4D20.11)
113 FORMAT(1H ,T8,'FS',T28,'GS',T108,'GSMAG'/1H ,1P6D20.11)
114 FORMAT(1H ,T8,'DG',T88,'P',T108,'SGS'/1H ,1P6D20.11/1H ,T8,
1'R',T88,'RMAG'/1H ,1P5D20.11)
115 FORMAT(1H0,T8,'CUBIC INTERPOLATION BETWEEN X AND XS'/1H0,T8,
1'DXC',T88,'ALPHAC'/1H ,1P5D20.11/1H ,T8,'XC'/1H ,1P4D20.11)
116 FORMAT(1H ,T8,'FC',T28,'GC',T108,'GCMAG'/1H ,1P6D20.11)
117 FORMAT(1H ,T8,'DGC',T88,'PC'/1H ,1P5D20.11/1H ,T8,'RC',T88,
1'RCMAG'/1H ,1P5D20.11)
118 FORMAT(1H0,T8,'OUTPUT OF THIS ITERATION'/1H0,T8,'X',T88,'F',
1T108,'GMAG'/1H ,1P6D20.11/1H ,T8,'REVM',T68,'VEVMPD'/1H ,
21P6D20.11)
119 FORMAT(1H0,T8,'NO. OF DAVIDON ITERATIONS HAS REACHED MAXIMUM')
120 FORMAT(1H1)
121 FORMAT(1H0,T8,'SGS'/1H ,1P1D20.11)
122 FORMAT(1H0,T8,'RESTART WITH IDENTITY V MATRIX SCALED BY EPSV')
RETURN
END

```

```

SUBROUTINE FOURBY(XTO, XTF, RSVO, VSVO, RSEO, VSEO, RSMO,
1VSMO, PVO, RSVD, VSVD, RSED, VSED, RSMD, VSMD, S11, S12, S21, S22)
  IMPLICIT REAL*8(A-H, L-M, O-Z)
  DIMENSION RSV(3), VSV(3), RSE(3), VSE(3), RSM(3), VSM(3), PVO(6),
1S11(3,3), S12(3,3), S21(3,3), S22(3,3), REV(3), VEV(3), RMV(3), VMV(3),
2REM(3), VEM(3), DR4RSV(3), DR4RSE(3), DR4RSM(3), BIGJ(6,6), STM(6,6),
3STNTM(6,6), MTXSV(6,6), MTXEV(6,6), MTXMV(6,6), MTXSE(6,6),
4MTXSM(6,6), MTXEM(6,6), DRSV(3), DVSV(3), DRSE(3), DVSE(3)
  DIMENSION RSVO(3), VSVO(3), RSEO(3), VSEO(3), RSMO(3), VSMO(3)
  DIMENSION RTM(3), DRSM(3), DVSM(3), DREV(3), DVEV(3), DRMV(3),
1DVMV(3), DREM(3), DVEM(3), CRSV(3), CVSV(3), CRSE(3), CVSE(3), CRSM(3),
2CVSM(3), CREV(3), CVEV(3), CRMV(3), CVMV(3), CREM(3), CVEM(3), PRSV(3)
3, PVSV(3), PRSE(3), PVSE(3), PRSM(3), PVSM(3), RRSV(3), RVSV(3),
4RRSE(3), RVSE(3), RRSM(3), RVSM(3), STMH(6,6), TEMP(6,6)
  DIMENSION VTM(3), PVTM(3), PVDTM(3), PV(6)
  DIMENSION RDV(3), VDV(3), RDS(3), VDS(3), RDE(3), VDE(3), RDM(3),
1VDM(3), RDVD(3), VDVD(3), RDS(3), VDS(3), RDMD(3),
2VDMD(3), RSVD(3), VSVD(3), RSED(3), VSED(3), RSMD(3), VSMD(3), RSL1(3),
3VSL1(3), RDL1(3), VDL1(3), RDL1D(3), VDL1D(3)
  COMMON/TDATA/TO, T, H, RSV, VSV, REV, VEV, RMV, VMV, RSE, VSE, RSM, VSM,
1REM, VEM, RSL1, VSL1, RDVD, VDVD, RDS, VDS, RDMD, VDMD,
2RDL1D, VDL1D, PV, LM, LDM, ANGV, ANGE, ANGS
  COMMON/CONST/MS, ME, MM, GL1, AUM, UTIME, UVELM, ERRMAX, DTR
  COMMON/FLAG/IMTX, IPV, IPTRAJ, IPVTM, IFILE, ITER, ITAR
  COMMON/CTM/TTM, RTM, VTM, LTM, LDTM, PVTM, PVDTM, STNTM
  DO 1 I=1,3
  RSV(I)=RSVO(I)
  VSV(I)=VSVO(I)
  RSE(I)=RSEO(I)
  VSE(I)=VSEO(I)
  RSM(I)=RSMO(I)
1 VSM(I)=VSMO(I)
  TO=XTO
  MSME=MS+ME
  MSMM=MS+MM
  MEMM=ME+MM
  ESE=ME/MSME
  MSM=MM/MSMM
  MEM=MM/MEMM
  EEM=ME/MEMM
  SSE=MS/MSME
  SSM=MS/MSMM
  PSISV=0.
  PSISE=0.
  PSISM=0.
  PSIEV=0.
  PSIMV=0.
  PSIEM=0.
  LM=0.
  LDM=0.
  CALL RVEMV(RSV, VSV, RSE, VSE, RSM, VSM, REV, VEV, RMV, VMV, REM, VEM)
  IF (IMTX.EQ.0) GO TO 2
  DO 2 I=1,6
  PV(I)=0.
  DO 2 J=1,6
  STM(I,J)=0.
  IF (I.EQ.J) STM(I,J)=1.0

```

```

2 CONTINUE
  ISTEP=0
  T=XTO
  TGO=XTF-T
  CALL CSTEP(TGO,RSV,REV,RMV,RSE,RSM,REM,H,DR4RSV,DR4RSE,
1DR4RSM,BIGJ)
  CALL DISP
  IF(IPV.EQ.0) GO TO 21
  LTM=1.
  TTM=XTO
  CALL PVEC(T,RSV,VSV,PVO,STM,PV,LM,LDM)
21 IF(IFILE.EQ.0) GO TO 22
  CALL FDATA
22 IF(IPTRAJ.EQ.0) GO TO 3
  WRITE(6,100)
  IF(IPV.EQ.0) GO TO 23
  WRITE(6,101)
23 CALL PTRAJ
  3 ISTEP=ISTEP+1
  CALL TWOBODY(RSV,VSV,H,MS,PSISV,IMTX,CRSV,CVSV,MTXSV)
  CALL TWOBODY(RSE,VSE,H,MSME,PSISE,0,CRSE,CVSE,MTXSE)
  CALL TWOBODY(RSM,VSM,H,MSMM,PSISM,0,CRSM,CVSM,MTXSM)
  CALL TWOBODY(REV,VEV,H,ME,PSIEV,IMTX,CREV,CVEV,MTXEV)
  CALL TWOBODY(RMV,VMV,H,MM,PSIMV,IMTX,CRMV,CVMV,MTXMV)
  CALL TWOBODY(REM,VEM,H,MEMM,PSIEM,0,CREM,CVEM,MTXEM)
  DO 4 I=1,3
    DRSV(I)=CRSV(I)-RSV(I)-H*VSV(I)
    DVSV(I)=CVSV(I)-VSV(I)
    DRSE(I)=CRSE(I)-RSE(I)-H*VSE(I)
    DVSE(I)=CVSE(I)-VSE(I)
    DRSM(I)=CRSM(I)-RSM(I)-H*VSM(I)
    DVSM(I)=CVSM(I)-VSM(I)
    DREV(I)=CREV(I)-REV(I)-H*VEV(I)
    DVEV(I)=CVEV(I)-VEV(I)
    DRMV(I)=CRMV(I)-RMV(I)-H*VMV(I)
    DVMV(I)=CVMV(I)-VMV(I)
    DREM(I)=CREM(I)-REM(I)-H*VEM(I)
    DVEM(I)=CVEM(I)-VEM(I)
    PRSV(I)=ESE*DRSE(I)+DREV(I)+MSM*DRSM(I)+DRMV(I)
    PVSV(I)=ESE*DVSE(I)+DVEV(I)+MSM*DVSM(I)+DVMV(I)
    PRSE(I)=MSM*DRSM(I)-MEM*DREM(I)
    PVSE(I)=MSM*DVSM(I)-MEM*DVEM(I)
    PRSM(I)=ESE*DRSE(I)+EEM*DREM(I)
    PVSM(I)=ESE*DVSE(I)+EEM*DVEM(I)
    RSV(I)=CRSV(I)+PRSV(I)
    VSV(I)=CVSV(I)+PVSV(I)
    RSE(I)=CRSE(I)+PRSE(I)
    VSE(I)=CVSE(I)+PVSE(I)
    RSM(I)=CRSM(I)+PRSM(I)
  4 VSM(I)=CVSM(I)+PVSM(I)
  CALL RVEMV(RSV,VSV,RSE,VSE,RSM,VSM,REV,VEV,RMV,VMV,REM,VEM)
  T=T+H
  CALL DELRV(H,RSV,REV,RMV,RSE,RSM,REM,CRSV,CREV,CRMV,CRSE,CRSM,
1CREM,DR4RSV,DR4RSE,DR4RSM,RRSV,RVSV,RRSE,RVSE,RRSM,RVSM)
  DO 5 I=1,3
    RSV(I)=RSV(I)+RRSV(I)
    VSV(I)=VSV(I)+RVSV(I)

```

```

RSE(I)=RSE(I)+RRSE(I)
VSE(I)=VSE(I)+RVSE(I)
RSM(I)=RSM(I)+RRSM(I)
5 VSM(I)=VSM(I)+RVSM(I)
CALL RVEMV(RSV,VSV,RSE,VSE,RSM,VSM,REV,VEV,RMV,VMV,REM,VEM)
IF(IMTX.EQ.0) GO TO 70
DO 6 I=1,6
DO 6 J=1,6
6 STMH(I,J)=MTXSV(I,J)+MTXEV(I,J)+MTXMV(I,J)-2.0*BIGJ(I,J)
CALL MXM(STMH,STM,TEMP,6,6,6)
DO 7 J=1,6
DO 7 I=1,6
7 STM(I,J)=TEMP(I,J)
70 CALL DISP
IF(IPV.EQ.0) GO TO 71
CALL PVEC(T,RSV,VSV,PVO,STM,PV,LM,LDM)
71 IF(IFILE.EQ.0) GO TO 72
CALL FDATA
72 IF(IPTRAJ.EQ.0) GO TO 8
CALL PTRAJ
8 IF (T.GE.XTF) GO TO 9
TGO=XTF-T
CALL CSTEP(TGO,RSV,REV,RMV,RSE,RSM,REM,H,DR4RSV,DR4RSE,
1DR4RSM,BIGJ)
GO TO 3
9 DO 10 I=1,3
RSVD(I)=RSV(I)
VSVD(I)=VSV(I)
RSED(I)=RSE(I)
VSED(I)=VSE(I)
RSMD(I)=RSM(I)
VSMD(I)=VSM(I)
DO 10 J=1,3
S11(I,J)=STM(I,J)
S12(I,J)=STM(I,J+3)
S21(I,J)=STM(I+3,J)
10 S22(I,J)=STM(I+3,J+3)
IF(IPV.EQ.0) GO TO 11
IF(IPVTM.EQ.0) GO TO 11
WRITE(6,102)TTM,LTM,LDTM,RTM,VTM
11 IF(IPTRAJ.EQ.0) GO TO 12
WRITE(6,103)ISTEP
12 CONTINUE
100 FORMAT(1H1,40X,'FOUR-BODY TRAJECTORY'/1H0,T8,'T',T28,'TDAY',
1T48,'H',T68,'RSVMAG',T88,'REVMAG',T108,'RMVMAG'/1H ,T8,'RSV',
2T68,'VSV'/1H ,T8,'REV',T68,'VEV'/1H ,T8,'RMV',T68,'VMV'/1H0,T8,
3'RSE',T68,'VSE'/1H ,T8,'RSM',T68,'VSM'/1H ,T8,'REM',T68,'VEM'/
41H ,T8,'RSL1',T68,'VSL1'/1H0,T8,'RDVD',T68,'VDVD'/1H ,T8,'RDSD',
5T68,'VDS1'/1H ,T8,'RDMD',T68,'VDMD'/
61H ,T8,'RDL1D',T68,'VDL1D'/1H0,T8,'ANGV',T28,'ANGE',T48,'ANGS',
61H0,T8,'RDL1D',T68,'VDL1D'/1H ,T8,'ANGV',T28,'ANGE',T48,'ANGS',
7T68,'HDAY')
101 FORMAT(1H ,T8,'PV',T68,'PVD'/1H ,T8,'PVMAG',T28,'PVDMAG')
102 FORMAT(1H0,'TIME OF MAXIMUM PRIMER VECTOR MAGNITUDE'/1H0,T8,
1'TTM',T28,'LTM',T48,'LDTM'/1H ,1P3D20.11/1H ,T8,'RTM',T68,'VTM'
2/1H ,1P6D20.11)
103 FORMAT(1H0,T8,'ISTEP FOR THIS LEG = ',I5)

```

RETURN  
END

```

SUBROUTINE THRBODY(XTO, XTF, REVO, VEVO, REMO, VEMO, PVO, REVD, VEVD,
1 REMD, VEMD, S11, S12, S21, S22)
  IMPLICIT REAL*8(A-H, L-M, O-Z)
  DIMENSION REVO(3), VEVO(3), REMO(3), VEMO(3), PVO(6), REVD(3), VEVD(3)
1, REMD(3), VEMD(3), S11(3,3), S12(3,3), S21(3,3), S22(3,3), REV(3),
2 VEV(3), REM(3), VEM(3), RMV(3), VMV(3), PV(6), STM(6,6), DR4REV(3),
3 BIGJ(6,6), CREV(3), CVEV(3), MTXEV(6,6), CREM(3), CVEM(3), MTXEM(6,6),
4 CRMV(3), CVMV(3), MTXMV(6,6), DRMV(3), DVMV(3),
5 PREV(3), PVEV(3), RREV(3), RVEV(3), STMH(6,6), TEMP(6,6), REL(3),
6 VEL(3), RDVD(3), VDVD(3), RDED(3), RDED(3), RDLD(3), VDLD(3)
  DIMENSION RTM(3), VTM(3), PVTM(3), PVDTM(3), STNTM(6,6)
  COMMON/TDATA3/TO, T, H, REV, VEV, RMV, VMV, REM, VEM, REL, VEL, RDVD,
  IVDVD, RDED, VDED, RDLD, VDLD, PV, LM, LDM, ANGV, ANGE, ANGM
  COMMON/CONST3/ME, MM, GAMMA, UDM, UTIME, UVELM, ERRMAX, DTR
  COMMON/FLAG/IMTX, IPV, IPTRAJ, IPVTM, IFILE, ITER, ITAR
  COMMON/CTM/TTM, RTM, VTM, LTM, LDTM, PVTM, PVDTM, STNTM
  DO 1 I=1,3
    REV(I)=REVO(I)
    VEV(I)=VEVO(I)
    REM(I)=REMO(I)
    VEM(I)=VEMO(I)
    RMV(I)=-REM(I)+REV(I)
1 VMV(I)=-VEM(I)+VEV(I)
  TO=XTO
  MEMM=ME+MM
  MEM=MM/MEMM
  PSIEV=0.
  PSIEM=0.
  PSIMV=0.
  LM=0.
  LDM=0.
  IF(IMTX.EQ.0)GO TO 2
  DO 2 I=1,6
    PV(I)=0.
  DO 2 J=1,6
    STM(I,J)=0.
  IF(I.EQ.J)STM(I,J)=1.0
2 CONTINUE
  ISTEP=0
  T=XTO
  TGO=XTF-T
  CALL CSTEP3(TGO,REV,RMV,REM,H,DR4REV,BIGJ)
  CALL DISP3
  IF(IPV.EQ.0) GO TO 21
  LTM=1.0
  TTM=XTO
  CALL PVEC(T,REV,VEV,PVO,STM,PV,LM,LDM)
21 IF(IFILE.EQ.0) GO TO 22
  CALL FDATA3
22 IF(IPTRAJ.EQ.0) GO TO 3
  WRITE(6,100)
  IF(IPV.EQ.0) GO TO 23
  WRITE(6,101)
23 CALL PTRAJ3
  3 ISTEP=ISTEP+1
  CALL TWOBODY(REV,VEV,H,ME,PSIEV,IMTX,CREV,CVEV,MTXEV)
  CALL TWOBODY(REM,VEM,H,MEMM,PSIEM,0,CREM,CVEM,MTXEM)

```



```

CALL TWOBODY(RMV,VMV,H,MM,PSIMV,IMTX,CRMV,CVMV,MTXMV)
DO 4 I=1,3
DRMV(I)=CRMV(I)-RMV(I)-H*VMV(I)
DVMV(I)=CVMV(I)-VMV(I)
PREV(I)=DRMV(I)
PVEV(I)=DVMV(I)
REV(I)=CREV(I)+PREV(I)
VEV(I)=CVEV(I)+PVEV(I)
REM(I)=CREM(I)
VEM(I)=CVEM(I)
RMV(I)=-REM(I)+REV(I)
4 VMV(I)=-VEM(I)+VEV(I)
T=T+H
CALL DELRV3(H,REV,RMV,CREV,CRMV,DR4REV,RREV,RVEV)
DO 5 I=1,3
REV(I)=REV(I)+RREV(I)
VEV(I)=VEV(I)+RVEV(I)
RMV(I)=-REM(I)+REV(I)
5 VMV(I)=-VEM(I)+VEV(I)
IF(IMTX.EQ.0) GO TO 70
DO 6 I=1,6
DO 6 J=1,6
6 STMH(I,J)=MTXEVI(I,J)+MTXMV(I,J)-2.0*BIGJ(I,J)
CALL MXM(STMH,STM,TEMP,6,6,6)
DO 7 I=1,6
DO 7 J=1,6
7 STM(I,J)=TEMP(I,J)
70 CALL DISP3
IF(IPV.EQ.0) GO TO 71
CALL PVEC(T,REV,VEV,PVO,STM,PV,LM,LDM)
71 IF(IFILE.EQ.0) GO TO 72
CALL FDATA3
72 IF(IPTRAJ.EQ.0) GO TO 8
CALL PTRAJ3
8 IF(T.GE.XTF) GO TO 9
TGO=XTF-T
CALL CSTEP3(TGO,REV,RMV,REM,H,DR4REV,BIGJ)
GO TO 3
9 DO 10 I=1,3
REVD(I)=REV(I)
VEVD(I)=VEV(I)
REMD(I)=REM(I)
VEMD(I)=VEM(I)
DO 10 J=1,3
S11(I,J)=STM(I,J)
S12(I,J)=STM(I,J+3)
S21(I,J)=STM(I+3,J)
10 S22(I,J)=STM(I+3,J+3)
IF(IPV.EQ.0) GO TO 11
IF(IPVTM.EQ.0) GO TO 11
WRITE(6,102)TTM,LTM,LDTM,RTM,VTM
11 IF(IPTRAJ.EQ.0) GO TO 12
WRITE(6,103)ISTEP
12 CONTINUE
100 FORMAT(1H1,40X,'THREE-BODY TRAJECTORY'/1H0,T8,'T',T28,'TDAY',
1T48,'H',T68,'REVMAG',T88,'RMVMAG'/1H ,T8,'REV',T68,'VEV'/
21H ,T8,'RMV',T68,'VMV'/1H ,T8,'REM',T68,'VEM'/1H ,T8,'REL1',

```

```
3T68,'VEL1'/1H0,T8,'RDVD',T68,'VDVD'/1H ,T8,'RDED',T68,'VDED'/  
41H ,T8,'RDL'D',T68,'VDLD'/1H ,T8,'ANGV',T28,'ANGE',T48,'ANGM')  
101 FORMAT(1H0,T8,'PV',T68,'PVD'/1H ,T8,'PVMAG',T28,'PVDMAG')  
102 FORMAT(1H0,T8,'TIME OF MAXIMUM PRIMER VECTOR MAGNITUDE'/1H0,  
1T8,'TTM',T28,'LTM',T48,'LDTM'/1H ,1P3D20.11/1H ,T8,'RTM',T68,  
2'VTM'/1H ,1P6D20.11)  
103 FORMAT(1H0,T8,'ISTEP FOR THIS LEG V ',I5)  
RETURN  
END
```

SUBROUTINE TWOBODY(XR0,XV0,TAU,MU,PSI,IMTX,XRF,XVF,P)

```
C GENERAL SOLUTION OF TWO BODY PROBLEM WITH PARTIAL DERIVATIVES
C   TRAN DOUBLE PRECISION SUBROUTINE
C   DOUBLE PRECISION SO(6),TAU,MU,PSI,S(6),P(6,6),PI(6,6),ACC(3),
C   1ACC0(3),R,RO,SIG0,ALPHA,PSIN,PSIP,A,AP,CO,C1,C2,
C   2C3,C4,C5X3,S1,S2,S3,DTAU,DTAUN,DTAUP,U,FM1,G,FD,GDM1
C   DOUBLE PRECISION XR0(3),XV0(3),XRF(3),XVF(3)
C   INPUTS
C   SO(1),SO(2),SO(3)=X0,Y0,Z0=POSITION COMPONENTS AT REFERENCE TIME TO
C   SO(4),SO(5),SO(6)=XDO,YDO,ZDO=VELOCITY COMPONENTS AT REFERENCE TIME TO
C   TAU=TIME INTERVAL (T-TO) FROM REFERENCE TIME TO TO SOLUTION TIME T
C   MU=CONSTANT IN DIFF. EQS. (XDD,YDD,ZDD)=-MU*(X,Y,Z)/(R**3)
C   PSI=APPROXIMATION FOR FINAL SOLUTION PSI OF KEPLER'S EQUATION
C   OUTPUTS
C   PSI=GENERALIZED ECCENTRIC ANOMALY=SOLUTION OF KEPLER'S EQUATION
C   S(1),S(2),S(3)=X,Y,Z=POSITION COMPONENTS AT SOLUTION TIME T=TO+TAU
C   S(4),S(5),S(6)=XD,YD,ZD=VELOCITY COMPONENTS AT SOLUTION TIME T=TO+TAU
C   P(I,J)=PARTIAL DERIVATIVE DS(I)/DSO(J) OF S(I) WITH RESPECT TO SO(J)
C   PI(I,J)=PARTIAL DSO(I)/DSO(J) WITH ROLES OF TO AND T REVERSED
C   ACC(I)=-MU*S(I)/(R**3)=ACCELERATION COMPONENT AT SOLUTION TIME T
C   ACC0(I)=-MU*SO(I)/(RO**3)=ACCELERATION COMPONENT AT REF TIME TO
C   R=RADIUS AT TIME T=SQUARE ROOT OF (X**2+Y**2+Z**2)
C   RO=RADIUS AT TIME TO
C   START OF INITIAL COMPUTATIONS
C   DO 9 I=1,3
C     SO(I)=XR0(I)
C   9 SO(I+3)=XV0(I)
C   COMPUTE RADIUS RO
C   S1=DMAX1(DABS(SO(1)),DABS(SO(2)),DABS(SO(3)))
C   S2=(SO(1)/S1)**2+(SO(2)/S1)**2+(SO(3)/S1)**2
C   RO=2.
C 10 R=RO
C   RO=(R+S2/R)*.5
C   IF(RO.LT.R) GO TO 10
C   RO=RO*S1
C   COMPUTE OTHER PARAMETERS
C   SIG0=SO(1)*SO(4)+SO(2)*SO(5)+SO(3)*SO(6)
C   ALPHA=SO(4)**2+SO(5)**2+SO(6)**2-2.*MU/RO
C   INITIALIZE SERIES MOD COUNT M TO ZERO
C   M=0
C   INITIALIZE BOUNDS PSIN AND PSIP FOR PSI OR SET PSI=0 IF TAU=0
C   IF(TAU) 20,30,40
C 20 PSIN=-1.D+38
C   PSIP=0.
C   DTAUN=PSIN
C   DTAUP=-TAU
C   GO TO 50
C 30 PSI=0.
C   GO TO 100
C 40 PSIN=0.
C   PSIP=+1.D+38
C   DTAUN=-TAU
C   DTAUP=PSIP
C   USE APPROXIMATION FOR PSI IF IT IS BETWEEN BOUNDS PSIN AND PSIP
```

```

50 IF(PSI.GT.PSIN.AND.PSI.LT.PSIP) GO TO 100
C TRY NEWTON'S METHOD FOR INITIAL PSI SET EQUAL TO ZERO
  PSI=TAU/RO
C) ET PSI=TAU IF NEWTON'S METHOD FAILS
  IF(PSI.LE.PSIN.OR.PSI.GE.PSIP) PSI=TAU
C END OF INITIAL COMPUTATIONS
C
C BEGINNING OF LOOP FOR SOLVING KEPLER'S EQUATION
C BEGINNING OF SERIES SUMMATION
C COMPUTE ARGUMENT A IN REDUCED SERIES OBTAINED BY FACTORING OUT PSI'S
100 A=ALPHA*PSI*PSI
  IF(DABS(A).LE.1.) GO TO 120
C SAVE A IN AP AND MOD A IF IT EXCEEDS UNITY IN MAGNITUDE
  AP=A
110 M=M+1
  A=A*.25
  IF(DABS(A).GT.1.) GO TO 110
C SUM SERIES C5X3=3*S5/PSI**5 AND C4=S4/PSI**4
120 C5X3=(1.+(1.+(1.+(1.+(1.+(1.+(1.+A/342.)*A/272.)*A/210.)*A/156.)
1   *A/110.)*A/72.)*A/42.)/40.
  C4=(1.+(1.+(1.+(1.+(1.+(1.+A/306.)*A/240.)*A/182.)*A/132.)
1   *A/90.)*A/56.)*A/30.)/24.
C COMPUTE SERIES C3=S3/PSI**3,C2=S2/PSI**2,C1=S1/PSI,C0=S0
  C3=(.5+A*C5X3)/3.
  C2=.5+A*C4
  C1=1.+A*C3
  C0=1.+A*C2
  IF(M.LE.0) GO TO 140
C)MOD SERIES C0 AND C1 IF NECESSARY WITH DOUBLE ANGLE FORMULAS
30 C1=C1*C0
  C0=2.*C0*C0-1.
  M=M-1
  IF(M.GT.0) GO TO 130
C DETERMINE C2,C3,C4,C5X3 FROM C0,C1,AP IF DEMOD REQUIRED
  C2=(C0-1.)/AP
  C3=(C1-1.)/AP
  C4=(C2-.5)/AP
  C5X3=(3.*C3-.5)/AP
C COMPUTE SERIES S1,S2,S3 FROM C1,C2,C3
140 S1=C1*PSI
  S2=C2*PSI*PSI
  S3=C3*PSI*PSI*PSI
C END OF SERIES SUMMATION
C COMPUTE RESIDUAL DTAU AND SLOPE R FOR KEPLER'S EQUATION
  G=RO*S1+SIG0*S2
  DTAU=(G+MU*S3)-TAU
  R=DABS(RO*C0+(SIG0*S1+MU*S2))
  IF(DTAU) 200,300,210
C RESET BOUND
200 PSIN=PSI
  DTAUN=DTAU
  GO TO 220
210 PSIP=PSI
  DTAUP=DTAU
)TRY NEWTON'S METHOD AND INITIALIZE SELECTOR N
220 PSI=PSI-DTAU/R
  N=0

```

```

C ACCEPT PSI IF IT IS BETWEEN BOUNDS PSIN AND PSIP
230 IF(PSI.GT.PSIN.AND.PSI.LT.PSIP) GO TO 100
C SELECT ALTERNATE METHOD OF COMPUTING PSI OR STOP ITERATIONS
N=N+1
GO TO (1,2,3,4,300),N
C TRY INCREMENTING BOUND WITH DTAU NEAREST ZERO BY THE RATIO 4*DTAU/TAU
1 IF(DABS(DTAUN).LT.DABS(DTAUP)) PSI=PSIN*(1.-(4.*DTAUN)/TAU)
IF(DABS(DTAUP).LT.DABS(DTAUN)) PSI=PSIP*(1.-(4.*DTAUP)/TAU)
GO TO 230
C TRY DOUBLING BOUND CLOSEST TO ZERO
2 IF(TAU.GT.0.) PSI=PSIN+PSIN
IF(TAU.LT.0.) PSI=PSIP+PSIP
GO TO 230
C TRY INTERPOLATION BETWEEN BOUNDS
3 PSI=PSIN+(PSIP-PSIN)*(-DTAUN/(DTAUP-DTAUN))
GO TO 230
C TRY HALVING BETWEEN BOUNDS
4 PSI=PSIN+(PSIP-PSIN)*.5
GO TO 230
C END OF LOOP FOR SOLVING KEPLER'S EQUATION
C
C COMPUTE REMAINING THREE OF FOUR FUNCTIONS FM1,G,FD,GDM1
300 FM1=-MU*S2/RO
FD=-MU*S1/RO/R
GDM1=-MU*S2/R
C COMPUTE COORDINATES AT SOLUTION TIME T=TO+TAU
DO 310 I=1,3
S(I)=SO(I)+(FM1*SO(I)+G*SO(I+3))
S(I+3)=(FD*SO(I)+GDM1*SO(I+3))+SO(I+3)
XRF(I)=S(I)
10 XVF(I)=S(I+3)
IF (IMTX.EQ. 0) GO TO 500
C COMPUTE ACCELERATIONS
DO 320 I=1,3
ACC(I)=-MU*S(I)/R/R/R
320 ACC0(I)=-MU*SO(I)/RO/RO/RO
C END OF COMPUTATION FOR COORDINATES AND ACCELERATIONS
C
C COMPUTATION OF PARTIAL DERIVATIVES
C COMPUTE COEFFICIENTS FOR STATE PARTIALS
U=S2*TAU+MU*(C4-C5X3)*PSI*PSI*PSI*PSI*PSI
P(1,1)=- (FD*S1+FM1/RO)/RO
P(1,2)=-FD*S2
P(2,1)=FM1*S1/RO
P(2,2)=FM1*S2
P(1,3)=P(1,2)
P(1,4)=-GDM1*S2
P(2,3)=P(2,2)
P(2,4)=G*S2
P(3,1)=-FD*(CO/RO/R+1./R/R+1./RO/RO)
P(3,2)=- (FD*S1+GDM1/R)/R
P(4,1)=-P(1,1)
P(4,2)=-P(1,2)
P(3,3)=P(3,2)
P(3,4)=-GDM1*S1/R
P(4,3)=-P(1,2)
P(4,4)=-P(1,4)

```

C COMPUTE COEFFICIENTS FOR MU PARTIALS

P(1,5)=-S1/R0/R

P(2,5)=S2/R0

) P(3,5)=U/R0-S3

P(1,6)=-P(1,5)

P(2,6)=S2/R

P(3,6)=-U/R+S3

DO 400 I=1,3

C MATRIX ACCUMULATIONS FOR STATE PARTIALS

DO 400 J=1,4

PI(J,1)=P(J,1)\*S0(I)+P(J,2)\*S0(I+3)

400 PI(J,I+3)=P(J,3)\*S0(I)+P(J,4)\*S0(I+3)

DO 410 I=1,3

DO 420 J=1,3

P(I,J)=S(I)\*PI(1,J)+S(I+3)\*PI(2,J)+U\*S(I+3)\*ACCO(J)

P(I,J+3)=S(I)\*PI(1,J+3)+S(I+3)\*PI(2,J+3)-U\*S(I+3)\*S0(J+3)

P(I+3,J)=S(I)\*PI(3,J)+S(I+3)\*PI(4,J)+U\*ACC(I)\*ACCO(J)

420 P(I+3,J+3)=S(I)\*PI(3,J+3)+S(I+3)\*PI(4,J+3)-U\*ACC(I)\*S0(J+3)

P(I,I)=P(I,I)+FM1+1.

P(I,I+3)=P(I,I+3)+G

P(I+3,I)=P(I+3,I)+FD

410 P(I+3,I+3)=P(I+3,I+3)+GDM1+1.

C END OF COMPUTATION FOR PARTIAL DERIVATIVES

C

C END OF PROGRAM - ALL OUTPUTS HAVE BEEN COMPUTED

500 CONTINUE

RETURN

END

```

SUBROUTINE CSTEP(TGO,RSV,REV,RMV,RSE,RSM,REM,H,
1DR4RSV,DR4RSE,DR4RSM,BIGJ)
  IMPLICIT REAL*8(A-H,M,O-Z)
  DIMENSION RSV(3),VSV(3),REV(3),VEV(3),RMV(3),VMV(3),RSE(3),
1VSE(3),RSM(3),VSM(3),REM(3),VEM(3),DR4RSV(3),DR4RSE(3),
2DR4RSM(3),URSV(3),UREV(3),URMV(3),URSE(3),URSM(3),UREM(3),
3UMTX(3,3),SVSV(3,3),EVEV(3,3),VMVM(3,3),SESE(3,3),SMSM(3,3),
4EMEM(3,3),BIGJ(6,6),TEMP1(3),TEMP2(3),TEMP3(3),TEMP4(3),
5TEMP5(3),TEMP6(3),TEMP7(3),TEMP8(3),TEMP9(3)
  DIMENSION RSVRSV(3),REVREV(3),RMVRMV(3),RSERSE(3),RSMRSM(3),
1REMREM(3),DEVSE(3),DSVSE(3),DMVSM(3),DSVSM(3),DMVEM(3),
2DEMSE(3),DEVEM(3),DEMSE(3),DSMSE(3)
  COMMON/CONST/MS,ME,MM,GL1,AUM,UTIME,UVELM,ERRMAX,DTR
  RSV3=VMAG(RSV)**3
  REV3=VMAG(REV)**3
  RMV3=VMAG(RMV)**3
  RSE3=VMAG(RSE)**3
  RSM3=VMAG(RSM)**3
  REM3=VMAG(REM)**3
  CALL UNITV(RSV,URSV)
  CALL UNITV(REV,UREV)
  CALL UNITV(RMV,URMV)
  CALL UNITV(RSE,URSE)
  CALL UNITV(RSM,URSM)
  CALL UNITV(REM,UREM)
  DO 1 I=1,3
  RSVRSV(I)=RSV(I)/RSV3
  REVREV(I)=REV(I)/REV3
  RMVRMV(I)=RMV(I)/RMV3
  RSERSE(I)=RSE(I)/RSE3
  RSMRSM(I)=RSM(I)/RSM3
  REMREM(I)=REM(I)/REM3
  DEVSE(I)=REVREV(I)+RSERSE(I)
  DSVSE(I)=RSVRSV(I)-RSERSE(I)
  DMVSM(I)=RMVRMV(I)+RSMRSM(I)
  DSVSM(I)=RSVRSV(I)-RSMRSM(I)
  DMVEM(I)=RMVRMV(I)+REMREM(I)
  DEMSM(I)=REMREM(I)-RSMRSM(I)
  DEVEM(I)=REVREV(I)-REMREM(I)
  DEMSE(I)=REMREM(I)+RSERSE(I)
  DSMSE(I)=RSMRSM(I)-RSERSE(I)
  DO 1 J=1,3
  UMTX(I,J)=0.0
  IF(I.EQ.J) UMTX(I,J)=1.0
1 CONTINUE
  CALL VVT(URSV,SVSV,3)
  CALL VVT(UREV,EVEV,3)
  CALL VVT(URMV,VMVM,3)
  CALL VVT(URSE,SESE,3)
  CALL VVT(URSM,SMSM,3)
  CALL VVT(UREM,EMEM,3)
  DO 2 I=1,3
  DO 2 J=1,3
  SVSV(I,J)=(UMTX(I,J)-3.0*SVSV(I,J))/RSV3
  EVEV(I,J)=(UMTX(I,J)-3.0*EVEV(I,J))/REV3
  VMVM(I,J)=(UMTX(I,J)-3.0*VMVM(I,J))/RMV3
  SESE(I,J)=(UMTX(I,J)-3.0*SESE(I,J))/RSE3

```

```

SMSM(I,J)=(UMTX(I,J)-3.0*SMSM(I,J))/RSM3
EMEM(I,J)=(UMTX(I,J)-3.0*EMEM(I,J))/REM3
2 CONTINUE
CALL MXV(SVSV,DEVSE,TEMP1,3,3)
CALL MXV(EVEV,DSVSE,TEMP2,3,3)
CALL MXV(SVSV,DMVSM,TEMP3,3,3)
CALL MXV(VMVM,DSVSM,TEMP4,3,3)
CALL MXV(EVEV,DMVEM,TEMP5,3,3)
CALL MXV(SESE,DEMSE,TEMP6,3,3)
CALL MXV(VMVM,DEVEM,TEMP7,3,3)
CALL MXV(SMSM,DEMSE,TEMP8,3,3)
CALL MXV(EMEM,DSMSE,TEMP9,3,3)
DO 3 I=1,3
DR4RSV(I)=MS*ME*(TEMP1(I)+TEMP2(I))+MS*MM*(TEMP3(I)+TEMP4(I))
1+ME*MM*(TEMP5(I)-TEMP6(I)+TEMP7(I)+TEMP8(I))
DR4RSE(I)=- (MS+ME)*MM*TEMP6(I)-MM*MS*TEMP9(I)+MM*ME*TEMP8(I)
DR4RSM(I)=(MS+MM)*ME*TEMP8(I)+ME*MS*TEMP9(I)-ME*MM*TEMP6(I)
3 CONTINUE
CPERR=VMAG(DR4RSV)/24.0
H=DSORT(DSORT(ERRMAX/CPERR))
IF(TGO.LT.H) H=TGO
DO 4 I=1,6
DO 4 J=1,6
BIGJ(I,J)=0.0
IF(I.EQ.J) BIGJ(I,J)=1.0
IF((J-I).EQ.3) BIGJ(I,J)=H
4 CONTINUE
RETURN
END

```



```

SUBROUTINE CSTEP3(TGO,REV,RMV,REM,H,DR4REV,BIGJ)
  IMPLICIT REAL*8(A-H,M,O-Z)
  DIMENSION REV(3),RMV(3),REM(3),DR4REV(3),BIGJ(6,6),UREV(3),
1 URMV(3),RMVRMV(3),REMREM(3),REVREV(3),DMVEM(3),DEVEM(3),
2 UMTX(3,3),EVEV(3,3),MVMV(3,3),TEMP1(3),TEMP2(3)
  COMMON/CONST3/ME,MM,GAMMA,UDM,UTIME,UVELM,ERRMAX,DTR
  REV3=VMAG(REV)**3
  RMV3=VMAG(RMV)**3
  REM3=VMAG(REM)**3
  CALL UNITV(REV,UREV)
  CALL UNITV(RMV,URMV)
  DO 1 I=1,3
  RMVRMV(I)=RMV(I)/RMV3
  REMREM(I)=REM(I)/REM3
  REVREV(I)=REV(I)/REV3
  DMVEM(I)=RMVRMV(I)+REMREM(I)
  DEVEM(I)=REVREV(I)-REMREM(I)
  DO 1 J=1,3
  UMTX(I,J)=0.
  IF(I.EQ.J) UMTX(I,J)=1.0
1 CONTINUE
  CALL VVT(UREV,EVEV,3)
  CALL VVT(URMV,MVMV,3)
  DO 2 I=1,3
  DO 2 J=1,3
  EVEV(I,J)=(UMTX(I,J)-3.0*EVEV(I,J))/REV3
2 MVMV(I,J)=(UMTX(I,J)-3.0*MVMV(I,J))/RMV3
  CALL MXV(EVEV,DMVEM,TEMP1,3,3)
  CALL MXV(MVMV,DEVEM,TEMP2,3,3)
  DO 3 I=1,3
3 DR4REV(I)=ME*MM*(TEMP1(I)+TEMP2(I))
  CPERR=VMAG(DR4REV)/24.0
  H=DSQRT(DSQRT(ERRMAX/CPERR))
  IF(TGO.LT.H) H=TGO
  DO 4 I=1,6
  DO 4 J=1,6
  BIGJ(I,J)=0.
  IF(I.EQ.J) BIGJ(I,J)=1.0
  IF((J-I).EQ.3) BIGJ(I,J)=H
4 CONTINUE
  RETURN
  END

```

```
SUBROUTINE DELRV(H,RSV,REV,RMV,RSE,RSM,REM,CRSV,CREV,CRMV,  
1CRSE,CRSM,CREM,DR4RSV,DR4RSE,DR4RSM,RRSV,RVSV,RRSE,RVSE,  
2RRSM,RVSM)
```

```
IMPLICIT REAL*8(A-H,M,O-Z)
```

```
DIMENSION RSV(3),REV(3),RMV(3),RSE(3),RSM(3),REM(3),CRSV(3),  
1CREV(3),CRMV(3),CRSE(3),CRSM(3),CREM(3),DR4RSV(3),DR4RSE(3),  
2DR4RSM(3),RRSV(3),RVSV(3),RRSE(3),RVSE(3),RRSM(3),RVSM(3)
```

```
DIMENSION SSV(3),SEV(3),SMV(3),SSE(3),SSM(3),SEM(3),DDRSV(3),  
1DDRSE(3),DDRSM(3)
```

```
COMMON/CONST/MS,ME,MM,GL1,AUM,UTIME,UVELM,ERRMAX,DTR
```

```
H2=H*H
```

```
H26=H2/6.0
```

```
H23=H2/3.0
```

```
RSV3=VMAG(RSV)**3
```

```
REV3=VMAG(REV)**3
```

```
RMV3=VMAG(RMV)**3
```

```
RSE3=VMAG(RSE)**3
```

```
RSM3=VMAG(RSM)**3
```

```
REM3=VMAG(REM)**3
```

```
CRSV3=VMAG(CRSV)**3
```

```
CREV3=VMAG(CREV)**3
```

```
CRMV3=VMAG(CRMV)**3
```

```
CRSE3=VMAG(CRSE)**3
```

```
CRSM3=VMAG(CRSM)**3
```

```
CREM3=VMAG(CREM)**3
```

```
DO 1 I=1,3
```

```
SSV(I)=CRSV(I)/CRSV3-RSV(I)/RSV3
```

```
SEV(I)=CREV(I)/CREV3-REV(I)/REV3
```

```
SMV(I)=CRMV(I)/CRMV3-RMV(I)/RMV3
```

```
SSE(I)=CRSE(I)/CRSE3-RSE(I)/RSE3
```

```
SSM(I)=CRSM(I)/CRSM3-RSM(I)/RSM3
```

```
SEM(I)=CREM(I)/CREM3-REM(I)/REM3
```

```
DDRSV(I)=MS*SSV(I)+ME*(SSE(I)+SEV(I))+MM*(SSM(I)+SMV(I))
```

```
DDRSE(I)=(MS+ME)*SSE(I)+MM*(SSM(I)-SEM(I))
```

```
DDRSM(I)=(MS+MM)*SSM(I)+ME*(SSE(I)+SEM(I))
```

```
RRSV(I)=H2*(DR4RSV(I)*H23+DDRSV(I))/20.0
```

```
RVSV(I)=H*(DR4RSV(I)*H26+DDRSV(I))/4.0
```

```
RRSE(I)=H2*(DR4RSE(I)*H23+DDRSE(I))/20.0
```

```
RVSE(I)=H*(DR4RSE(I)*H26+DDRSE(I))/4.0
```

```
RRSM(I)=H2*(DR4RSM(I)*H23+DDRSM(I))/20.0
```

```
RVSM(I)=H*(DR4RSM(I)*H26+DDRSM(I))/4.0
```

```
1 CONTINUE
```

```
RETURN
```

```
END
```

```

SUBROUTINE DELRV3(H,REV,RMV,CREV,CRMV,DR4REV,RREV,RVEV)
  IMPLICIT REAL*8(A-H,M,O-Z)
  DIMENSION REV(3),RMV(3),CREV(3),CRMV(3),DR4REV(3),RREV(3),
1 RVEV(3),SEV(3),SMV(3),DDREV(3)
  COMMON/CONST3/ME,MM,GAMMA,UDM,UTIME,UVELM,ERRMAX,DTR
  H2=H*H
  H26=H2/6.0
  H23=H2/3.0
  REV3=VMAG(REV)**3
  CREV3=VMAG(CREV)**3
  RMV3=VMAG(RMV)**3
  CRMV3=VMAG(CRMV)**3
  DO 1 I=1,3
  SEV(I)=CREV(I)/CREV3-REV(I)/REV3
  SMV(I)=CRMV(I)/CRMV3-RMV(I)/RMV3
  DDREV(I)=ME*SEV(I)+MM*SMV(I)
  RREV(I)=H2*(DR4REV(I)*H23+DDREV(I))/20.0
1 RVEV(I)=H*(DR4REV(I)*H26+DDREV(I))/4.0
  RETURN
  END

```

```

SUBROUTINE COMIC(REVMAG,VEVMAG,LON,THE,OINC,OBL,RSEO,VSEO,
IREVO,VEVO,RSVO,VSVO,VSVI)
  IMPLICIT REAL*8(A-H,L-M,O-Z)
  DIMENSION RSEO(3),VSEO(3),REVO(3),VEVO(3),RSVO(3),VSVO(3),
1VSVI(3),TEMP(3),VEVOP(3)
  COMMON/CONST/MS,ME,MM,GL1,AUM,UTIME,UVELM,ERRMAX,DTR
  COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
  COMMON/TRIG/CLON,SLON,CTHE,STHE,CINC,SINC,COBL,SOBL
  IF(ITER.GT.0) GO TO 1
  VCIR=DSQRT(ME/REVMAG)
  CINC=DCOS(OINC)
  SINC=DSIN(OINC)
  COBL=DCOS(OBL)
  SOBL=DSIN(OBL)
1 CLON=DCOS(LON)
  SLON=DSIN(LON)
  CTHE=DCOS(THE)
  STHE=DSIN(THE)
  REVO(1)=REVMAG*(CLON*CTHE-SLON*CINC*STHE)
  REVO(2)=REVMAG*(COBL*(SLON*CTHE+CLON*CINC*STHE)
1   +SOBL*SINC*STHE)
  REVO(3)=REVMAG*(-SOBL*(SLON*CTHE+CLON*CINC*STHE)
1   +COBL*SINC*STHE)
  VEVOP(1)=VEVMAG*(-CLON*STHE-SLON*CINC*CTHE)
  VEVOP(2)=VEVMAG*(COBL*(-SLON*STHE+CLON*CINC*CTHE)
1   +SOBL*SINC*CTHE)
  VEVOP(3)=VEVMAG*(-SOBL*(-SLON*STHE+CLON*CINC*CTHE)
1   +COBL*SINC*CTHE)
  CALL UNITV(VEVOP,TEMP)
  DO 2 I=1,3
  VEVO(I)=VCIR*TEMP(I)
  VSVO(I)=VSEO(I)+VEVO(I)
  RSVO(I)=RSEO(I)+REVO(I)
2 VSVI(I)=VSEO(I)+VEVOP(I)
  RETURN
  END

```

```

SUBROUTINE COMIC3(REVMAG,VEVMAG,LON,THE,OINC,REVO,VEVO,VEVOP)
IMPLICIT REAL*8(A-H,L-M,O-Z)
DIMENSION REVO(3),VEVO(3),VEVOP(3),TEMP(3)
COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
COMMON/CONST3/ME,MM,GAMMA,UDM,UTIME,UVELM,ERRMAX,DTR
COMMON/TRIG3/CLON,SLON,CTHE,STHE,CINC,SINC
IF(ITER.GT.0) GO TO 1
VCIR=DSQRT(ME/REVMAG)
CINC=DCOS(OINC)
SINC=DSIN(OINC)
1 CLON=DCOS(LON)
  SLON=DSIN(LON)
  CTHE=DCOS(THE)
  STHE=DSIN(THE)
  REVO(1)=REVMAG*(CLON*CTHE-SLON*CINC*STHE)
  REVO(2)=REVMAG*(SLON*CTHE+CLON*CINC*STHE)
  REVO(3)=REVMAG*SINC*STHE
  VEVOP(1)=VEVMAG*(-CLON*STHE-SLON*CINC*CTHE)
  VEVOP(2)=VEVMAG*(-SLON*STHE+CLON*CINC*CTHE)
  VEVOP(3)=VEVMAG*SINC*CTHE
  CALL UNITV(VEVOP,TEMP)
  DO 2 I=1,3
2 VEVO(I)=VCIR*TEMP(I)
  WRITE(6,100)REVO,VEVO
  WRITE(6,101)VEVOP
100 FORMAT(1H0,T8,'REVO',T68,'VEVO'/1H ,1P6D20.11)
101 FORMAT(1H ,T8,'VEVOP'/1H ,1P3D20.11)
  RETURN
  END

```

```

SUBROUTINE COMFG(TSTART,TEND,REVMAG,XD,OINC,OBL,
1RSEO,VSEO,RSMO,VSMO,FD,TESTRD,GD,TSID,LTD,S11,S12,S21,S22,
2UVI,UVF)
  IMPLICIT REAL*8(A-H,L-M,O-Z)
  DIMENSION RSEO(3),VSEO(3),RSMO(3),VSMO(3),REVO(3),VEVO(3),
1RSVO(3),VSVO(3),VSVI(3),RSVTAR(3),VSVTAR(3),REMF(3),VEMF(3),
2PVO(6),RSVF(3),VSVF(3),RSEF(3),VSEF(3),RSMF(3),VSMF(3),
3S11(3,3),S12(3,3),S21(3,3),S22(3,3),TSID(3),DVI(3),UVI(3),
4DVF(3),UVF(3),DRODX(3,3),DVODX(3,3),TEMP1(3,3),TEMP2(3,3),
5GD(3),LTD(3,3),XD(3),REVF(3),VEVF(3),RMVF(3),VMVF(3)
  DIMENSION DUVI(3),DUM(3),S12I(3,3)
  COMMON/CONST/MS,ME,MM,GL1,AUM,UTIME,UVELM,ERRMAX,DTR
  COMMON/TRIG/CLON,SLON,CTHE,STHE,CINC,SINC,COBL,SOBL
  COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
  COMMON/TARG/AY,AZ,ATAR,RSVTAR,VSVTAR
  VEVMAG=XD(1)
  LON=XD(2)
  THE=XD(3)
  VEVMP5=VEVMAG/UVELM
  LOND=LON/DTR
  THED=THE/DTR
  WRITE(6,108)
  WRITE(6,101)XD,VEVMP5,LOND,THED
  CALL COMIC(REVMAG,VEVMAG,LON,THE,OINC,OBL,RSEO,VSEO,
1REVO,VEVO,RSVO,VSVO,VSVI)
  CALL FOURBY(TSTART,TEND,RSVO,VSVI,RSEO,VSEO,RSMO,VSMO,
1PVO,RSVF,VSVF,RSEF,VSEF,RSMF,VSMF,S11,S12,S21,S22)
  CALL RVEMV(RSVF,VSVF,RSEF,VSEF,RSMF,VSMF,REVF,VEVF,RMVF,VMVF,
1REMF,VEMF)
  WRITE(6,102)RSVF,VSVF
  IF(ITER.GT.0) GO TO 11
  IF(ITAR.GT.0) GO TO 11
  CALL CTAR(RSEF,VSEF,RSMF,VSMF,REMF,VEMF,AY,AZ,ATAR,RSVTAR,
1VSVTAR)
11 DO 2 I=1,3
  TSID(I)=RSVF(I)-RSVTAR(I)
  DVI(I)=VSVI(I)-VSVO(I)
  2 DVF(I)=VSVTAR(I)-VSVF(I)
  TESTRD=VMAG(TSID)
  DVIMAG=VMAG(DVI)
  DVFMAG=VMAG(DVF)
  DVIMPS=DVIMAG/UVELM
  DVFMPS=DVFMAG/UVELM
  CALL UNITV(DVI,UVI)
  CALL UNITV(DVF,UVF)
  FD=DVFMAG
  CALL MXV(S11,UVI,DUM,3,3)
  DO 21 I=1,3
21 DUM(I)=UVF(I)-DUM(I)
  CALL INVERT(S12,S12I)
  CALL MXV(S12I,DUM,DUVI,3,3)
  WRITE(6,103)DVI,UVI
  WRITE(6,104)DVF,UVF
  WRITE(6,105)DUVI,TSID
  WRITE(6,106)DVIMAG,DVFMAG,DVIMPS,DVFMPS,FD,TESTRD
  ORODX(1,1)=0.
  DRODX(1,2)=REVMAG*(-SLON*CTHE-CLON*CINC*STHE)

```

```

DRODX(1,3)=REVMAG*(-CLON*STHE-SLON*CINC*CTHE)
DRODX(2,1)=0.
DRODX(2,2)=REVMAG*COBL*(CLON*CTHE-SLON*CINC*STHE)
DRODX(2,3)=REVMAG*(COBL*(-SLON*STHE+CLON*CINC*CTHE)
1      +SOBL*SINC*CTHE)
DRODX(3,1)=0.
DRODX(3,2)=-REVMAG*SOBL*(CLON*CTHE-SLON*CINC*STHE)
DRODX(3,3)=REVMAG*(-SOBL*(-SLON*STHE+CLON*CINC*CTHE)
1      +COBL*SINC*CTHE)
DVODX(1,1)=-CLON*STHE-SLON*CINC*CTHE
DVODX(1,2)=VEVMAG*(SLON*STHE-CLON*CINC*CTHE)
DVODX(1,3)=VEVMAG*(-CLON*CTHE+SLON*CINC*STHE)
DVODX(2,1)=COBL*(-SLON*STHE+CLON*CINC*CTHE)
1      +SOBL*SINC*CTHE
DVODX(2,2)=VEVMAG*COBL*(-CLON*STHE-SLON*CINC*CTHE)
DVODX(2,3)=VEVMAG*(COBL*(-SLON*CTHE-CLON*CINC*STHE)
1      -SOBL*SINC*STHE)
DVODX(3,1)=-SOBL*(-SLON*STHE+CLON*CINC*CTHE)
1      +COBL*SINC*CTHE
DVODX(3,2)=-VEVMAG*SOBL*(-CLON*STHE-SLON*CINC*CTHE)
DVODX(3,3)=VEVMAG*(-SOBL*(-SLON*CTHE-CLON*CINC*STHE)
1      -COBL*SINC*STHE)
CALL MXM(S21,DRODX,TEMP1,3,3,3)
CALL MXM(S22,DVODX,TEMP2,3,3,3)
DO 3 I=1,3
DO 3 J=1,3
3 TEMP1(I,J)=TEMP1(I,J)+TEMP2(I,J)
CALL MTRANS(TEMP1,TEMP2,3,3)
CALL MXV(TEMP2,UVF,GD,3,3)
DO 4 I=1,3
4 GD(I)=-GD(I)
WRITE(6,107)GD
CALL MXM(S11,DRODX,TEMP1,3,3,3)
CALL MXM(S12,DVODX,TEMP2,3,3,3)
DO 5 I=1,3
DO 5 J=1,3
LTD(I,J)=TEMP1(I,J)+TEMP2(I,J)
5 CONTINUE
101 FORMAT(1H0,T8,'XD',T68,'VEVMPS',T88,'LOND',T108,'THED'/1H ,
11P6D20.11)
102 FORMAT(1H ,T8,'RSVF',T68,'VSVF'/1H ,1P6D20.11)
103 FORMAT(1H ,T8,'DVI',T68,'UVI'/1H ,1P6D20.11)
104 FORMAT(1H ,T8,'DVF',T68,'UVF'/1H ,1P6D20.11)
105 FORMAT(1H ,T8,'DUV1',T68,'TSID'/1H ,1P6D20.11)
106 FORMAT(1H ,T8,'DVIMAG',T28,'DVFMAG',T48,'DVIMPS',T68,'DVFMPS',
1T88,'FD',T108,'TESTRD'/1H ,1P6D20.11)
107 FORMAT(1H ,T8,'GD'/1H ,1P3D20.11)
108 FORMAT(1H0,T8,'-----')
RETURN
END

```

```

SUBROUTINE COMFG3(TSTART,TEND,REVMAG,XD,OINC,REMO,
1VEMO,FD,TESTRD,GD,TSID,LTD,S11,S12,S21,S22,UVI,UVF)
  IMPLICIT REAL*8(A-H,L-M,O-Z)
  DIMENSION REMO(3),VEMO(3),REVO(3),VEVO(3),VEVOP(3),REVTAR(3),
1VEVTAR(3),PVO(6),REVF(3),VEVF(3),REMF(3),VEMF(3),S11(3,3),XD(3),
2S12(3,3),S21(3,3),S22(3,3),TSID(3),DVI(2),UVI(3),DVF(3),UVF(3),
3DRODX(3,3),DVODX(3,3),TEMP1(3,3),TEMP2(3,3),GD(3),LTD(3,3)
  DIMENSION DUVI(3),DUM(3),S121(3,3)
  COMMON/CONST3/ME,MM,GAMMA,UDM,UTIME,UVELM,ERRMAX,DTR
  COMMON/TRIG3/CLON,SLON,CTHE,STHE,CINC,SINC
  COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
  COMMON/TARG/AY,AZ,ATAR,REVTAR,VEVTAR
  VEVMAG=XD(1)
  LON=XD(2)
  THE=XD(3)
  VEVMP5=VEVMAG/UVELM
  LOND=LON/DTR
  THED=THE/DTR
  WRITE(6,108)
  WRITE(6,101)XD,VEVMP5,LOND,THED
  CALL COMIC3(REVMAG,VEVMAG,LON,THE,OINC,REVO,VEVO,VEVOP)
  CALL THRBDY(TSTART,TEND,REVO,VEVOP,REMO,VEMO,PVO,REVF,VEVF,
1REMF,VEMF,S11,S12,S21,S22)
  WRITE(6,102)REVF,VEVF
  IF(ITER.GT.0) GO TO 11
  IF(ITAR.GT.0) GO TO 11
  CALL CTAR3(REMF,VEMF,AY,AZ,ATAR,REVTAR,VEVTAR)
11 DO 2 I=1,3
  TSID(I)=REVF(I)-REVTAR(I)
  DVI(I)=VEVOP(I)-VEVO(I)
  2 DVF(I)=VEVTAR(I)-VEVF(I)
  TESTRD=VMAG(TSID)
  DVIMAG=VMAG(DVI)
  DVFMAG=VMAG(DVF)
  DVIMPS=DVIMAG/UVELM
  DVFMPS=DVFMAG/UVELM
  CALL UNITV(DVI,UVI)
  CALL UNITV(DVF,UVF)
  FD=DVFMAG
  CALL MXV(S11,UVI,DUM,3,3)
  DO 21 I=1,3
  21 DUM(I)=UVF(I)-DUM(I)
  CALL INVERT(S12,S121)
  CALL MXV(S121,DUM,DUVI,3,3)
  WRITE(6,103)DVI,UVI
  WRITE(6,104)DVF,UVF
  WRITE(6,105)DUVI,TSID
  WRITE(6,106)DVIMAG,DVFMAG,DVIMPS,DVFMPS,FD,TESTRD
  DRODX(1,1)=0.
  DRODX(1,2)=REVMAG*(-SLON*CTHE-CLON*CINC*STHE)
  DRODX(1,3)=REVMAG*(-CLON*STHE-SLON*CINC*CTHE)
  DRODX(2,1)=0.
  DRODX(2,2)=REVMAG*(CLON*CTHE-SLON*CINC*STHE)
  DRODX(2,3)=REVMAG*(-SLON*STHE+CLON*CINC*CTHE)
  DRODX(3,1)=0.
  DRODX(3,2)=0.
  DRODX(3,3)=REVMAG*SINC*CTHE

```



```

DVODX(1,1)=-CLON*STHE-SLON*CINC*CTHE
DVODX(1,2)=VEVMAG*(SLON*STHE-CLON*CINC*CTHE)
DVODX(1,3)=VEVMAG*(-CLON*CTHE+SLON*CINC*STHE)
DVODX(2,1)=-SLON*STHE+CLON*CINC*CTHE
DVODX(2,2)=VEVMAG*(-CLON*STHE-SLON*CINC*CTHE)
DVODX(2,3)=VEVMAG*(-SLON*CTHE-CLON*CINC*STHE)
DVODX(3,1)=SINC*CTHE
DVODX(3,2)=0.
DVODX(3,3)=-VEVMAG*SINC*STHE
CALL MXM(S21,DRODX,TEMP1,3,3,3)
CALL MXM(S22,DVODX,TEMP2,3,3,3)
DO 3 I=1,3
DO 3 J=1,3
3 TEMP1(I,J)=TEMP1(I,J)+TEMP2(I,J)
CALL MTRANS(TEMP1,TEMP2,3,3)
CALL MXV(TEMP2,UVF,GD,3,3)
DO 4 I=1,3
4 GD(I)=-GD(I)
WRITE(6,107)GD
CALL MXM(S11,DRODX,TEMP1,3,3,3)
CALL MXM(S12,DVODX,TEMP2,3,3,3)
DO 5 I=1,3
DO 5 J=1,3
LTD(I,J)=TEMP1(I,J)+TEMP2(I,J)
5 CONTINUE
100 FORMAT(1H0,T8,'COMPUTED TARGET'/1H0,T8,'REVTAR',T68,'VEVTAR'/
11H ,1P6D20.11)
101 FORMAT(1H0,T8,'XD',T68,'VEVMPS',T88,'LOND',T108,'THED'/1H ,
11P6D20.11)
102 FORMAT(1H ,T8,'REVF',T68,'VEVF'/1H ,1P6D20.11)
103 FORMAT(1H ,T8,'DVI',T68,'UVI'/1H ,1P6D20.11)
104 FORMAT(1H ,T8,'DVF',T68,'UVF'/1H ,1P6D20.11)
105 FORMAT(1H ,T8,'DUVI',T68,'TSID'/1H ,1P6D20.11)
106 FORMAT(1H ,T8,'DVIMAG',T28,'DVFMAG',T48,'DVIMPS',T68,'DVFMPS',
1T88,'FD',T108,'TESTRD'/1H ,1P6D20.11)
107 FORMAT(1H ,T8,'GD'/1H ,1P3D20.11)
108 FORMAT(1H0,T8,'-----',)
RETURN
END

```

```

SUBROUTINE COMF(TSTART, TM, TEND, RSVO, VSVO, VSVI, RSEO, VSEO, RSMO,
1VSMO, ERRMIN, ILINC, KNRSV, ITERD, ITLMAX, RSVMD, VSVMPD, RSVTAR,
2VSVTAR, S11D, S12D, RSVM, VSVMM, VSVMP, DV, UVI, UVM, UVF, SMI11, SMI12,
3SMI21, SMI22, SFM11, SFM12, SFM21, SFM22)

```

```

IMPLICIT REAL*8(A-H, K-M, O-Z)

```

```

C   EXTRAPOLATE STATES TO TM AND SOLVE LAMBERT PROBLEM FROM TM TO
C   TEND

```

```

DIMENSION RSVO(3), VSVO(3), VSVI(3), RSEO(3), VSEO(3), RSMO(3),
1VSMO(3), PVO(6), RSVM(3), VSVMM(3), RSEM(3), VSEM(3), RSMM(3),
2VSMM(3), SMI11(3,3), SMI12(3,3), SMI21(3,3), SMI22(3,3), SFM11(3,3),
3SFM12(3,3), SFM21(3,3), SFM22(3,3), S11D(3,3), S12D(3,3),
4RSVMX(3), VSVMPX(3), DRM(3), TEMP1(3,3), DUM(3),
5RSVF(3), VSVF(3), RSEF(3), VSEF(3), RSMF(3), VSMF(3), VSVMP(3),
6DVI(3), DVM(3), DVF(3), UVI(3), UVM(3), UVF(3), DVMPX(3)

```

```

DIMENSION RSVTAR(3), VSVTAR(3), RSVMD(3), VSVMPD(3)

```

```

COMMON/CONST/MS, ME, MM, GL1, AUM, UTIME, UVELM, ERRMAX, DTR

```

```

KNR=KNRSV

```

```

WRITE(6,100)

```

```

WRITE(6,101)VSVI, TM

```

```

C   ADVANCE STATES FROM TSTART TO TM

```

```

CALL FOURBY(TSTART, TM, RSVO, VSVI, RSEO, VSEO, RSMO, VSMO,
1PVO, RSVM, VSVMM, RSEM, VSEM, RSMM, VSMM, SMI11, SMI12, SMI21, SMI22)

```

```

IF(ITERD.GT.0) GO TO 3

```

```

DO 2 I=1,3

```

```

VSVMPX(I)=VSVMP(I)

```

```

2 RSVMX(I)=RSVM(I)

```

```

KNR=KNRSV

```

```

CALL LAMB(TM, TEND, RSVMX, VSVMPX, RSEM, VSEM, RSMM, VSMM, KNR, ITLMAX,
1ERRMIN, RSVTAR, RSVF, VSVF, RSEF, VSEF, RSMF, VSMF, SFM11, SFM12, SFM21,
2SFM22)

```

```

GO TO 71

```

```

C   INCREMENTAL SOLUTION OF SECOND LEG LAMBERT PROBLEM

```

```

3 DO 4 I=1,3

```

```

VSVMPX(I)=VSVMPD(I)

```

```

RSVMX(I)=RSVMD(I)

```

```

DRM(I)=(RSVM(I)-RSVMD(I))/ILINC

```

```

DO 4 J=1,3

```

```

SFM11(I,J)=S11D(I,J)

```

```

4 SFM12(I,J)=S12D(I,J)

```

```

WRITE(6,102)RSVMD, RSVM, DRM, ILINC

```

```

WRITE(6,106)VSVMPD

```

```

DO 7 ITERL=1, ILINC

```

```

CALL INVERT(SFM12, TEMP1)

```

```

CALL MXV(SFM11, DRM, DUM, 3, 3)

```

```

CALL MXV(TEMP1, DUM, DVMPX, 3, 3)

```

```

DO 5 I=1,3

```

```

DVMPX(I)=-DVMPX(I)

```

```

RSVMX(I)=RSVMX(I)+DRM(I)

```

```

5 VSVMPX(I)=VSVMPX(I)+DVMPX(I)

```

```

WRITE(6,103)ITERL, DVMPX, RSVMX, VSVMPX

```

```

KNR=KNRSV

```

```

6 CALL LAMB(TM, TEND, RSVMX, VSVMPX, RSEM, VSEM, RSMM, VSMM,
1KNR, ITLMAX, ERRMIN, RSVTAR, RSVF, VSVF, RSEF, VSEF, RSMF, VSMF, SFM11,
2SFM12, SFM21, SFM22)

```

```

7 CONTINUE

```

```

SAVE VSVMP AND COMPUTE DV

```

```

71 DO 8 I=1,3

```

```

VSVMP(I)=VSVMPX(I)
DVI(I)=VSVI(I)-VSV0(I)
DVM(I)=VSVMP(I)-VSVMM(I)
8 DVF(I)=VSVTAR(I)-VSVF(I)
DVIMAG=VMAG(DVI)
DVMMAG=VMAG(DVM)
DVF MAG=VMAG(DVF)
DV=DVIMAG+DVMMAG+DVF MAG
DVIMPS=DVIMAG/UELM
DVMMPS=DVMMAG/UELM
DVF MPS=DVF MAG/UELM
DV MPS=DV/UELM
CALL UNITV(DVI,UVI)
CALL UNITV(DVM,UVM)
CALL UNITV(DVF,UVF)
WRITE(6,104)DVI,UVI,DVM,UVM,DVF,UVF,DVIMAG,DVMMAG,DVF MAG,
1DV,DVIMPS,DVMMPS,DVF MPS,DV MPS
100 FORMAT(1H0,40X,'FIRST LEG TRAJECTORY')
101 FORMAT(1H0,T8,'VSVI',T68,'TM'/1H ,1P4D20.11)
102 FORMAT(1H0,T8,'SOLVE SECOND LEG LAMBERT PROBLEM IN INCREMENTS'
1/1H0,T8,'RSVMD',T68,'RSVM'/1H ,1P6D20.11/1H ,T8,'DRM',T68,
2'ILINC'/1H ,1P3D20.11,110)
103 FORMAT(1H1,T8,'ITERL',T28,'DVMPX'/1H ,110,10X,1P3D20.11/1H ,T8,
1'RSVMX',T68,'VSVMPX'/1H ,1P6D20.11)
104 FORMAT(1H0,T8,'DVI',T68,'UVI'/1H ,1P6D20.11/1H ,T8,'DVM',T68,
1'UVM'/1H ,1P6D20.11/1H ,T8,'DVF',T68,'UVF'/1H ,1P6D20.11/1H ,
2T8,'DVIMAG',T28,'DVMMAG',T48,'DVF MAG',T68,'DV'/1H ,1P4D20.11/
31H ,T8,'DVIMPS',T28,'DVMMPS',T48,'DVF MPS',T68,'DV MPS'/1H ,
41P4D20.11)
106 FORMAT(1H ,T8,'VSVMPD'/1H ,1P3D20.11)
RETURN
END

```

```

SUBROUTINE COMF3(TSTART, TM, TEND, REVO, VEVO, VEVOP, REMO, VEMO,
1ERRMIN, ILINC, KNRSV, ITERD, ITLMAX, REVMD, VEVMPD, REV TAR, VEVTAR,
2S11D, S12D, REVM, VEVMM, VEVMP, DV, UVI, UVM, UVF, SMI11, SMI12, SMI21,
3SMI22, SFM11, SFM12, SFM21, SFM22)

```

```

IMPLICIT REAL*8(A-H, K-M, O-Z)

```

```

EXTRAPOLATE STATES TO TM AND SOLVE LAMBERT PROBLEM FROM TM TO
TEND

```

```

DIMENSION REVO(3), VEVO(3), VEVOP(3), REMO(3), VEMO(3), PVO(6),
1REVM(3), VEVMM(3), VEVMP(3), REMM(3), VEMM(3), SMI11(3,3), SMI12(3,3),
2SMI21(3,3), SMI22(3,3), SFM11(3,3), SFM12(3,3), SFM21(3,3),
3SFM22(3,3), S11D(3,3), S12D(3,3), REVMX(3), VEVMPX(3), DRM(3),
4REVF(3), VEVF(3), REMF(3), VEMF(3), DVI(3), DVM(3), DVF(3),
5UVI(3), UVM(3), UVF(3), DVMPX(3), REV TAR(3), VEVTAR(3), REVMD(3),
6VEVMPD(3), TEMP1(3,3), DUM(3)

```

```

COMMON/CONST3/ME, MM, GAMMA, UDM, UTIME, UVELM, ERRMAX, DTR
WRITE(6,100)

```

```

WRITE(6,101)VEVOP, TM

```

```

ADVANCE STATES FROM TSTART TO TM

```

```

CALL THRBDY(TSTART, TM, REVO, VEVOP, REMO, VEMO, PVO, REVM, VEVMM,
1REMM, VEMM, SMI11, SMI12, SMI21, SMI22)

```

```

IF(ITERD.GT.0)GO TO 3

```

```

DO 2 I=1,3

```

```

VEVMPX(I)=VEVMP(I)

```

```

2 REVMX(I)=REVM(I)

```

```

KNR=KNRSV

```

```

CALL LAMB3(TM, TEND, REVMX, VEVMPX, REMM, VEMM, KNR, ITLMAX, ERRMIN,
1REV TAR, REVF, VEVF, REMF, VEMF, SFM11, SFM12, SFM21, SFM22)

```

```

GO TO 71

```

```

INCREMENTAL SOLUTION OF SECOND LEG LAMBERT PROBLEM

```

```

3 DO 4 I=1,3

```

```

VEVMPX(I)=VEVMPD(I)

```

```

REVMX(I)=REVMD(I)

```

```

DRM(I)=(REVM(I)-REVMD(I))/ILINC

```

```

DO 4 J=1,3

```

```

SFM11(I,J)=S11D(I,J)

```

```

4 SFM12(I,J)=S12D(I,J)

```

```

WRITE(6,102)REVMD, REVM, DRM, ILINC

```

```

WRITE(6,106)VEVMPD

```

```

DO 7 ITERL=1, ILINC

```

```

CALL INVERT(SFM12, TEMP1)

```

```

CALL MXV(SFM11, DRM, DUM, 3, 3)

```

```

CALL MXV(TEMP1, DUM, DVMPX, 3, 3)

```

```

DO 5 I=1,3

```

```

DVMPX(I)=-DVMPX(I)

```

```

REVMX(I)=REVMX(I)+DRM(I)

```

```

5 VEVMPX(I)=VEVMPX(I)+DVMPX(I)

```

```

WRITE(6,103)ITERL, DVMPX, REVMX, VEVMPX

```

```

KNR=KNRSV

```

```

6 CALL LAMB3(TM, TEND, REVMX, VEVMPX, REMM, VEMM, KNR, ITLMAX, ERRMIN,
1REV TAR, REVF, VEVF, REMF, VEMF, SFM11, SFM12, SFM21, SFM22)

```

```

7 CONTINUE

```

```

SAVE VEVMP AND COMPUTE DV

```

```

71 DO 8 I=1,3

```

```

VEVMP(I)=VEVMPX(I)

```

```

DVI(I)=VEVOP(I)-VEVO(I)

```

```

DVM(I)=VEVMP(I)-VEVMM(I)

```

```

8 DVF(I)=VEVTAR(I)-VEVF(I)

```

```

DVIMAG=VMAG(DVI)
DVMMAG=VMAG(DVM)
DVFMAG=VMAG(DVF)
DV=DVIMAG+DVMMAG+DVFMAG
DVIMPS=DVIMAG/UELM
DVMMPS=DVMMAG/UELM
DVFMPS=DVFMAG/UELM
DVMP5=DMAG/UELM
CALL UNITV(DVI,UVI)
CALL UNITV(DVM,UVM)
CALL UNITV(DVF,UVF)
WRITE(6,104)DVI,UVI,DVM,UVM,DVF,UVF,DVIMAG,DVMMAG,DVFMAG,
1DV,DVIMPS,DVMMPS,DVFMPS,DVMP5
100 FORMAT(1H0,40X,'FIRST LEG TRAJECTORY')
101 FORMAT(1H0,T8,'VEVOP',T68,'TM'/1H ,1P4D20.11)
102 FORMAT(1H0,T8,'SOLVE SECOND LEG LAMBERT PROBLEM IN INCREMENTS'
1/1H0,T8,'REVM',T68,'REVM'/1H ,1P6D20.11/1H ,T8,'DRM',T68,
2'ILINC'/1H ,1P3D20.11,110)
103 FORMAT(1H1,T8,'ITERL',T28,'DVMPX'/1H ,110,10X,1P3D20.11/1H ,T8,
1'REVMX',T68,'VEVMPX'/1H ,1P6D20.11)
104 FORMAT(1H0,T8,'DVI',T68,'UVI'/1H ,1P6D20.11/1H ,T8,'DVM',T68,
1'UVM'/1H ,1P6D20.11/1H ,T8,'DVF',T68,'UVF'/1H ,1P6D20.11/1H ,
2T8,'DVIMAG',T28,'DVMMAG',T48,'DVFMAG',T68,'DV'/1H ,1P4D20.11/
31H ,T8,'DVIMPS',T28,'DVMMPS',T48,'DVFMPS',T68,'DVMP5'/1H ,
41P4D20.11)
106 FORMAT(1H ,T8,'VEVMPD'/1H ,1P3D20.11)
RETURN
END

```

```

SUBROUTINE COMG(SMI11,SMI12,SMI21,SMI22,SFM11,SFM12,VTMP,
1VTMM,UVI,UVM,UVF,DUVI,DUVMM,DUVMP,G)
  IMPLICIT REAL*8(A-H,O-Z)
  DIMENSION SMI11(3,3),SMI12(3,3),SMI21(3,3),SMI22(3,3),
1SFM11(3,3),SFM12(3,3),VTMP(3),VTMM(3),UVI(3),UVM(3),UVF(3),
2G(4),SMI12I(3,3),TEMP1(3,3),DUM1(3),DUM2(3),DUM3(3),
3SFM12I(3,3),DUVI(3),DUVMM(3),DUVMP(3),SMI12T(3,3),GVI(3),DVM(3)
  CALL INVERT(SMI12,SMI12I)
  CALL MXV(SMI11,UVI,DUM3,3,3)
  DO 1 I=1,3
1 DUM3(I)=UVM(I)-DUM3(I)
  CALL MXV(SMI12I,DUM3,DUVI,3,3)
  CALL MXV(SMI22,DUVI,DUM2,3,3)
  CALL MXV(SMI21,UVI,DUM1,3,3)
  DO 2 I=1,3
2 DUVMM(I)=DUM1(I)+DUM2(I)
  CALL INVERT(SFM12,SFM12I)
  CALL MXV(SFM11,UVM,DUM1,3,3)
  DO 3 I=1,3
3 DUM1(I)=UVF(I)-DUM1(I)
  CALL MXV(SFM12I,DUM1,DUVMP,3,3)
  PVMMAG=DOT(DUVMM,UVM,3)
  PVPMAG=DOT(DUVMP,UVM,3)
  CALL MTRANS(SMI12,SMI12T,3,3)
  DO 4 I=1,3
4 DUM3(I)=DUVMP(I)-DUVMM(I)
  CALL MXV(SMI12T,DUM3,GVI,3,3)
  DO 5 I=1,3
  G(I)=GVI(I)
5 DVM(I)=VTMP(I)-VTMM(I)
  G(4)=-DOT(DUVMP,DVM,3)
  WRITE(6,100)DUVI,PVMMAG,PVPMAG
  WRITE(6,101)DUVMM,DUVMP
  WRITE(6,102)G
100 FORMAT(1H ,T8,'DUVI',T68,'PVMMAG',T88,'PVPMAG'/1H ,1P5D20.11)
101 FORMAT(1H ,T8,'DUVMM',T68,'DUVMP'/1H ,1P6D20.11)
102 FORMAT(1H ,T8,'G'/1H ,1P4D20.11)
  RETURN
  END

```

```

SUBROUTINE CTAR(RSEF,VSEF,RSMF,VSMF,REMF,VEMF,AY,AZ,ATAR,RSVTAR,
1VSVTAR)
  IMPLICIT REAL*8(A-H,K-M,O-Z)
  DIMENSION RSEF(3),VSEF(3),REMF(3),VEMF(3),RSVTAR(3),VSVTAR(3),
1RSL1(3),VSL1(3),UXLS(3),TEMP(3),UZLS(3),UYLS(3),CLS(3,3),
2CSL(3,3),RLTARL(3),VLTARL(3),RLTAR(3),TEMP1(3),OMGSL(3)
  DIMENSION RSMF(3),VSMF(3)
  COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
  COMMON/CONST/MS,ME,MM,GL1,AUM,UTIME,UVELM,ERRMAX,DTR
  MEM=MM/(ME+MM)
  WRITE(6,100)
  DO 1 I=1,3
    RSL1(I)=(1.-GL1)*(RSEF(I)+MEM*REMF(I))
  1 VSL1(I)=(1.-GL1)*(VSEF(I)+MEM*VEMF(I))
    IF(AY.NE.0.) GO TO 11
    IF(AZ.NE.0.) GO TO 11
    DO 12 I=1,3
      RSVTAR(I)=RSL1(I)
  12 VSVTAR(I)=VSL1(I)
    GO TO 5
  11 CALL UNITV(RSL1,UXLS)
    CALL VXV(RSEF,VSEF,TEMP)
    CALL VXV(RSMF,VSMF,TEMP1)
    DO 13 I=1,3
  13 TEMP(I)=ME*TEMP(I)+MM*TEMP1(I)
    CALL UNITV(TEMP,UZLS)
    CALL VXV(UZLS,UXLS,UYLS)
    DO 2 J=1,3
      CLS(1,J)=UXLS(J)
      CLS(2,J)=UYLS(J)
  2 CLS(3,J)=UZLS(J)
    CALL MTRANS(CLS,CSL,3,3)
    MU=(ME+MM)/(MS+ME+MM)
    BL=(1.-MU)/(1.-GL1)**3+MU/GL1**3
    OMEGAN=DSORT(1.-BL/2.+DSORT((3.*BL/2.)**2-2.*BL))
    KTAR=2.*OMEGAN/(OMEGAN**2+2.*BL+1.)
    CATAR=DCOS(ATAR)
    SATAR=DSIN(ATAR)
    RLTARL(1)=KTAR*AY*SATAR
    RLTARL(2)=AY*CATAR
    RLTARL(3)=AZ*SATAR
    VLTARL(1)=KTAR*OMEGAN*AY*CATAR
    VLTARL(2)=-OMEGAN*AY*SATAR
    VLTARL(3)=OMEGAN*AZ*CATAR
    WRITE(6,101)RLTARL,VLTARL
    CALL MXV(CSL,RLTARL,RLTAR,3,3)
    RMAG=VMAG(RSL1)
    VY=DOT(UYLS,VSL1,3)
    DO 3 I=1,3
  3 OMGSL(I)=VY*UZLS(I)/RMAG
    CALL VXV(OMGSL,RLTAR,TEMP)
    CALL MXV(CSL,VLTARL,TEMP1,3,3)
    DO 4 I=1,3
      RSVTAR(I)=RSL1(I)+RLTAR(I)
  4 VSVTAR(I)=VSL1(I)+TEMP1(I)+TEMP(I)
  5 CONTINUE
  WRITE(6,102)RSVTAR,VSVTAR

```

```
WRITE(6,103)
100 FORMAT(1H0,T8,'COMPUTED TARGET')
101 FORMAT(1H0,T8,'RLTARL',T68,'VLTARL'/1H ,1P6D20.11)
102 FORMAT(1H ,T8,'RSVTAR',T68,'VSVTAR'/1H ,1P6D20.11)
103 FORMAT(1H0)
RETURN
END
```



```

SUBROUTINE CTAR3(REMF,VEMF,AY,AZ,ATAR,REVTAR,VEVTAR)
  IMPLICIT REAL*8(A-H,K-M,O-Z)
  DIMENSION REMF(3),VEMF(3),REVTAR(3),VEVTAR(3),REL(3),VEL(3),
1 UXLE(3),TEMP(3),UZLE(3),UYLE(3),CLE(3,3),CEL(3,3),RLTARL(3),
2 VLTARL(3),RLTAR(3),TEMP1(3),OMGEL(3)
  COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
  COMMON/CONST3/ME,MM,GAMMA,UDM,UTIME,UVELM,ERRMAX,DTR
  WRITE(6,100)
  DO 1 I=1,3
    REL(I)=(1.-GAMMA)*REMF(I)
1  VEL(I)=(1.-GAMMA)*VEMF(I)
    IF(AY.NE.0.) GO TO 11
    IF(AZ.NE.0.) GO TO 11
    DO 12 I=1,3
      REVTAR(I)=REL(I)
12  VEVTAR(I)=VEL(I)
    GO TO 5
11  CALL UNITV(REGM,UXLE)
    CALL VXV(REMF,VEMF,TEMP)
    CALL UNITV(TEMP,UZLE)
    CALL VXV(UZLE,UXLE,LYLE)
    DO 2 J=1,3
      CLE(1,J)=UXLE(J)
      CLE(2,J)=UYLE(J)
2  CLE(3,J)=UZLE(J)
    CALL MTRANS(CLE,CEL,3,3)
    MU=MM/(ME+MM)
    BL=(1.-MU)/(1.-GAMMA)**3+MU/GAMMA**3
    OMEGAN=DSQRT(1.-BL/2.+DSQRT((3.*BL/2.)**2-2.*BL))
    KTAR=2.*OMEGAN/(OMEGAN**2+2.*BL+1.)
    CATAR=DCOS(ATAR)
    SATAR=DSIN(ATAR)
    RLTARL(1)=KTAR*AY*SATAR
    RLTARL(2)=AY*CATAR
    RLTARL(3)=AZ*SATAR
    VLTARL(1)=KTAR*OMEGAN*AY*CATAR
    VLTARL(2)=-OMEGAN*AY*SATAR
    VLTARL(3)=OMEGAN*AZ*CATAR
    WRITE(6,101)RLTARL,VLTARL
    CALL MXV(CSL,RLTARL,RLTAR,3,3)
    RMAG=VMAG(REL)
    VY=DOT(UYLE,VEL,3)
    DO 3 I=1,3
3  OMGEL(I)=VY*UZLE(I)/RMAG
    CALL VXV(OMGEL,RLTAR,TEMP)
    CALL MXV(CEL,VLTARL,TEMP1,3,3)
    DO 4 I=1,3
      REVTAR(I)=REL(I)+RLTAR(I)
4  VEVTAR(I)=VEL(I)+TEMP1(I)+TEMP(I)
5  CONTINUE
    WRITE(6,102)REVTAR,VEVTAR
    WRITE(6,103)
100  FORMAT(1H0,T8,'COMPUTED TARGET')
101  FORMAT(1H ,T8,'RLTARL',T68,'VLTARL'/1H ,1P6D20.11)
102  FORMAT(1H ,T8,'REVTAR',T68,'VEVTAR'/1H ,1P6D20.11)
103  FORMAT(1H0)
  RETURN

```

END

```

SUBROUTINE LAMB(TO,TF,RSV,VSV,RSE,VSE,RSM,VSM,KNR,ITLMAX,
IERRMIN,RSVTAR,RSVX,VSX,RSEX,VSEX,RSMX,VSMX,S11,S12,S21,S22)
C SOLVE LAMBERT PROBLEM BY NEWTON-RAPHSON METHOD ITERATING ON
INITIAL VELOCITY
IMPLICIT REAL*8(A-H,K-M,O-Z)
DIMENSION RSV(3),VSV(3),RSE(3),VSE(3),RSM(3),VSM(3),RSVTAR(3),
1S11(3,3),S12(3,3),S21(3,3),S22(3,3),RSVX(3),VSX(3),RSEX(3),
2VSEX(3),RSMX(3),VSMX(3),REX(3),VEX(3),RMX(3),VMX(3),
3REMX(3),VEMX(3),ERR(3),G2(2,2),G(3,3),GI(3,3),G2I(2,2),VIT(3),
4DELV(3),UDELV(3),KDELV(3),PVO(6)
COMMON/CONST/MS,ME,MM,GL1,AUM,UTIME,UVELM,ERRMAX,DTR
ITER=1
TESTRP=100.0
WRITE(6,108)
WRITE(6,100)TO,TF,RSV,VSV
DO 1 I=1,3
PVO(I)=0.
PVO(I+3)=0.
1 VIT(I)=VSV(I)
2 WRITE(6,101)ITER,VIT
CALL FOURBY(TO,TF,RSV,VIT,RSE,VSE,RSM,VSM,PVO,RSVX,
1VSX,RSEX,VSEX,RSMX,VSMX,S11,S12,S21,S22)
CALL RVEMV(RSVX,VSX,RSEX,VSEX,RSMX,VSMX,REX,VEX,RMX,VMX,
IREMX,VEMX)
DO 3 I=1,3
3 ERR(I)=RSVTAR(I)-RSVX(I)
TESTR=VMAG(ERR)
DO 4 I=1,3
DO 4 J=1,3
4 G(I,J)=-S12(I,J)
WRITE(6,102)RSVX,VSX
WRITE(6,107)ERR,TESTR
IF(ERR(3).NE.0.0) GO TO 5
DO 6 I=1,3
DO 6 J=1,3
6 G2(I,J)=G(I,J)
D2=G2(1,1)*G2(2,2)-G2(1,2)*G2(2,1)
G2I(1,1)=G2(2,2)/D2
G2I(1,2)=-G2(1,2)/D2
G2I(2,1)=-G2(2,1)/D2
G2I(2,2)=G2(1,1)/D2
DO 7 I=1,2
DO 7 J=1,2
GI(I,J)=G2I(I,J)
GI(I,3)=0.0
7 GI(3,J)=0.0
GI(3,3)=0.0
GO TO 8
5 CALL INVERT(G,GI)
8 IF (TESTR.GE.TESTRP) GO TO 9
DO 10 I=1,3
10 VSV(I)=VIT(I)
IF (TESTR.LT.IERRMIN) GO TO 14
CALL MXV(GI,ERR,DELV,3,3)
DO 11 I=1,3
11 DELV(I)=-DELV(I)
TESTRP=TESTR

```

```

DV=VMAG(DELV)
CALL UNITV(DELV,UDELV)
KNR=2.0*KNR
IF(KNR.GT.DV) KNR=DV
GO TO 12
9 KNR=KNR/10.0
12 DO 13 I=1,3
   KDELV(I)=KNR*UDELV(I)
13 VIT(I)=VSV(I)+KDELV(I)
   ITER=ITER+1
   KLV=VMAG(KDELV)
   KLVMIN=10.0-24
   IF(KLV.LT.KLVMIN) GO TO 14
   IF(ITER.LT.ITLMAX) GO TO 2
   WRITE(6,112)
   STOP
14 CONTINUE
   WRITE(6,109)
   WRITE(6,110)VSV
   WRITE(6,111)RSVX,VSVX
   WRITE(6,103)RE VX,VE VX
   WRITE(6,104)RM VX,VM VX
   WRITE(6,105)RSEX,VSEX
   WRITE(6,106)RSMX,VSMX
100 FORMAT(1H0,T8,'T0',T28,'TF'/1H ,1P2D20.11/1H ,T8,'RSV',T68,
1'VSV'/1H ,1P6D20.11)
101 FORMAT(1H0,T8,'ITER',T28,'VIT'/1H ,I10,10X,1P3D20.11)
102 FORMAT(1H ,T8,'RSVX',T68,'VSVX'/1H ,1P6D20.11)
103 FORMAT(1H ,T8,'RE VX',T68,'VE VX'/1H ,1P6D20.11)
104 FORMAT(1H ,T8,'RM VX',T68,'VM VX'/1H ,1P6D20.11)
105 FORMAT(1H ,T8,'RSEX',T68,'VSEX'/1H ,1P6D20.11)
106 FORMAT(1H ,T8,'RSMX',T68,'VSMX'/1H ,1P6D20.11)
107 FORMAT(1H ,T8,'ERR',T68,'TESTR'/1H ,1P4D20.11)
108 FORMAT(1H1)
109 FORMAT(1H0,T8,'LAMBERT PROBLEM HAS CONVERGED')
110 FORMAT(1H0,T8,'CONVERGED VSV'/1H ,1P3D20.11)
111 FORMAT(1H0,T8,'RSVX',T68,'VSVX'/1H ,1P6D20.11)
112 FORMAT(1H0,T8,'NO. OF LAMBERT ITERATIONS HAS REACHED MAXIMUM')
RETURN
END

```

```

SUBROUTINE LAMB3(TO,TF,REV,VEV,REM,VEM,KNR,ITLMAX,ERRMIN,REVTAR,
1REVF,VEVF,REMF,VEMF,S11,S12,S21,S22)
IMPLICIT REAL*8(A-H,K-M,O-Z)
THIS ROUTINE
C SOLVES 3-BODY LAMBERT PROBLEM BY NEWTON-RAPHSON METHOD ITERATING
C ON INITIAL VELOCITY.
DIMENSION REV(3),VEV(3),REM(3),VEM(3),REVTAR(3),REVF(3),VEVF(3),
1REMF(3),VEMF(3),S11(3,3),S12(3,3),S21(3,3),S22(3,3),VIT(3),
2PVO(6),RMVF(3),VMVF(3),ERR(3),G(3,3),G2(2,2),G2I(2,2),GI(3,3),
3DELV(3),UDELV(3),KDELV(3)
COMMON/CONST3/ME,MM,GAMMA,UDM,UTIME,UVELM,ERRMAX,DTR
ITER=1
TESTRP=100.0
WRITE(6,100)TO,TF,REV,VEV
DO 1 I=1,3
PVO(I)=0.
PVO(I+3)=0.
1 VIT(I)=VEV(I)
2 WRITE(6,101)ITER,VIT
CALL THRBODY(TO,TF,REV,VEV,REM,VEM,PVO,REVF,VEVF,REMF,VEMF,S11,
1S12,S21,S22)
DO 21 I=1,3
RMVF(I)=-REMF(I)+REVF(I)
21 VMVF(I)=-VEMF(I)+VEVF(I)
DO 3 I=1,3
3 ERR(I)=REVTAR(I)-REVF(I)
TESTR=VMAG(ERR)
DO 4 I=1,3
DO 4 J=1,3
4 G(I,J)=-S12(I,J)
WRITE(6,102)REVF,VEVF
WRITE(6,105)ERR,TESTR
IF(ERR(3).NE.0.0) GO TO 5
DO 6 I=1,2
DO 6 J=1,2
6 G2(I,J)=G(I,J)
D2=G2(1,1)*G2(2,2)-G2(1,2)*G2(2,1)
G2I(1,1)=G2(2,2)/D2
G2I(1,2)=-G2(1,2)/D2
G2I(2,1)=-G2(2,1)/D2
G2I(2,2)=G2(1,1)/D2
DO 7 I=1,2
DO 7 J=1,2
GI(I,4)=G2I(I,J)
GI(I,3)=0.
7 GI(3,J)=0.
GI(3,3)=0.
GO TO 8
5 CALL INVERT(G,GI)
8 IF(TESTR.GE.TESTRP) GO TO 9
DO 10 I=1,3
10 VEV(I)=VIT(I)
IF(TESTR.LT.ERRMIN) GO TO 14
CALL MXV(GI,ERR,DELV,3,3)
DO 11 I=1,3
11 DELV(I)=-DELV(I)
TESTRP=TESTR

```

```

DV=VMAG(DELV)
CALL UNITV(DELV,UDELV)
KNR=2.0*KNR
IF(KNR.GT.DV) KNR=DV
GO TO 12
9 KNR=KNR/10.0
12 DO 13 I=1,3
KDELV(I)=KNR*UDELV(I)
13 VIT(I)=VEV(I)+KDELV(I)
ITER=ITER+1
KLV=VMAG(KDELV)
KLVMIN=10.D-24
IF(KLV.LT.JLVMIN) GO TO 14
IF(ITER.LT.ITLMAX) GO TO 2
WRITE(6,110)
STOP
14 CONTINUE
WRITE(6,107)
WRITE(6,108)VEV
WRITE(6,109)REVF,VEVF
WRITE(6,103)RMVF,VMVF
WRITE(6,104)REMF,VEMF
100 FORMAT(1H0,T8,'T0',T28,'TF'/1H ,1P2D20.11/1H ,T8,'REV',T68,
1'VEV'/1H ,1P6D20.11)
101 FORMAT(1H0,T8,'ITER',T28,'VIT'/1H ,I10,10X,1P3D20.11)
102 FORMAT(1H ,T8,'REVF',T68,'VEVF'/1H ,1P6D20.11)
103 FORMAT(1H ,T8,'RMVF',T68,'VMVF'/1H ,1P6D20.11)
104 FORMAT(1H ,T8,'REMF',T68,'VEMF'/1H ,1P6D20.11)
105 FORMAT(1H ,T8,'ERR',T68,'TESTR'/1H ,1P4D20.11)
107 FORMAT(1H0,T8,'LAMBERT PROBLEM HAS CONVERGED')
108 FORMAT(1H0,T8,'CONVERGED VEV'/1H ,1P3D20.11)
109 FORMAT(1H0,T8,'REVF',T68,'VEVF'/1H ,1P6D20.11)
110 FORMAT(1H0,T8,'NO. OF LAMBERT ITERATIONS HAS REACHED MAXIMUM')
RETURN
END

```

## SUBROUTINE DISP

```

IMPLICIT REAL*8(A-H,L-M,O-Z)
DIMENSION RSV(3),VSV(3),RSE(3),VSE(3),RSM(3),VSM(3),UXDS(3),
1Q(3),TEMP1(3),TEMP2(3),UZDS(3),UYDS(3),OMGSD(3),RDVD(3),VDVD(3),
2RDM(3),VDM(3),CDS(3,3),RDV(3),VDV(3),RDS(3),
3VDS(3),RDM(3),VDM(3),RDS(3),VDS(3)
DIMENSION RSB(3),VSB(3),RSD(3),VSD(3),REV(3),VEV(3),RMV(3),
1VMV(3),PV(6),RSL1(3),VSL1(3),RDL1(3),VDL1(3),RDL1D(3),VDL1D(3),
2URSV(3),URSE(3),UREV(3),REM(3),VEM(3)
COMMON/CONST/MS,ME,MM,GL1,AUM,UTIME,UVELM,ERRMAX,DTR
COMMON/TDATA/TO,T,H,RSV,VSV,REV,VEV,RMV,VMV,RSE,VSE,RSM,VSM,
1REM,VEM,RSL1,VSL1,RDVD,VDVD,RDS,VDS,RDM,VDM,
2RDL1D,VDL1D,PV,LM,LDM,ANGV,ANGE,ANGS
ESE=ME/(MS+ME)
MEM=MM/(ME+MM)
DO 1 I=1,3
RSB(I)=RSE(I)+MEM*REM(I)
VSB(I)=VSE(I)+MEM*VEM(I)
RSD(I)=RSE(I)
VSD(I)=VSE(I)
RSL1(I)=(1.-GL1)*RSB(I)
1 VSL1(I)=(1.-GL1)*VSB(I)
CALL UNITV(RSE,UXDS)
CALL VXV(RSE,VSE,Q)
CALL UNITV(Q,UZDS)
CALL VXV(UZDS,UXDS,UYDS)
DO 3 J=1,3
CDS(1,J)=UXDS(J)
CDS(2,J)=UYDS(J)
3 CDS(3,J)=UZDS(J)
RSEMAG=VMAG(RSE)
VY=DOT(UYDS,VSE,3)
DO 4 I=1,3
4 OMGSD(I)=UZDS(I)*VY/RSEMAG
DO 5 I=1,3
RDV(I)=RSV(I)-RSD(I)
VDV(I)=VSV(I)-VSD(I)
RDS(I)=-RSD(I)
VDS(I)=-VSD(I)
RDM(I)=RSM(I)-RSD(I)
VDM(I)=VSM(I)-VSD(I)
RDL1(I)=RSL1(I)-RSD(I)
5 VDL1(I)=VSL1(I)-VSD(I)
COMPUTE STATE VECTOR IN ROTATING FRAME
CALL MXV(CDS,RDV,RDVD,3,3)
CALL VXV(OMGSD,RDV,TEMP1)
DO 6 I=1,3
6 TEMP2(I)=VDV(I)-TEMP1(I)
CALL MXV(CDS,TEMP2,VDVD,3,3)
CALL MXV(CDS,RDS,RDS,3,3)
CALL VXV(OMGSD,RDS,TEMP1)
DO 7 I=1,3
7 TEMP2(I)=VDS(I)-TEMP1(I)
CALL MXV(CDS,TEMP2,VDSD,3,3)
CALL MXV(CDS,RDM,RDM,3,3)
CALL VXV(OMGSD,RDM,TEMP1)
DO 9 I=1,3

```

```

9 TEMP2(I)=VDM(I)-TEMP1(I)
  CALL MXV(CDS,TEMP2,VDM,3,3)
  CALL MXV(CDS,RDL1,RDL1D,3,3)
  CALL VXV(OMGSD,RDL1,TEMP1)
  DO 10 I=1,3
10 TEMP2(I)=VDL1(I)-TEMP1(I)
  CALL MXV(CDS,TEMP2,VDL1D,3,3)
C  COMPUTE ANGLES AT SUN,EARTH AND VEHICLE
  CALL UNITV(RSE,URSE)
  CALL UNITV(REV,UREV)
  CALL UNITV(RSV,URSV)
  ANGV=DARCOS(DOT(UREV,URSV,3))/DTR
  ANGE=DARCOS(-DOT(URSE,UREV,3))/DTR
  ANGS=180.0-(ANGV+ANGE)
  RETURN
  END

```



## SUBROUTINE DISP3

```

IMPLICIT REAL*8(A-H,L-M,O-Z)
DIMENSION REV(3),VEV(3),RMV(3),VMV(3),REM(3),VEM(3),REL(3),
1VEL(3),RDVD(3),VDVD(3),RDED(3),VDED(3),
2RDL(3),VLD(3),PV(6),UXDE(3),UYDE(3),UZDE(3),Q(3),CDE(3,3),
3OMGED(3),RDV(3),VDV(3),RDE(3),VDE(3),RDL(3),
4VDL(3),TEMP1(3),TEMP2(3),UREM(3),URMV(3),UREV(3),RED(3),VED(3)
COMMON/CONST3/ME,MM,GAMMA,UDM,UTIME,UVELM,ERRMAX,DTR
COMMON/TDATA3/T0,T,H,REV,VEV,RMV,VMV,REM,VEM,REL,VEL,RDVD,
1VDVD,RDED,VDED,RDL,VLD,PV,LM,LDM,ANGV,ANGE,ANGM
EEM=ME/(ME+MM)
DO 1 I=1,3
RED(I)=REM(I)
VED(I)=VEM(I)
REL(I)=(1.-GAMMA)*REM(I)
1 VEL(I)=(1.-GAMMA)*VEM(I)
CALL UNITV(REM,UXDE)
CALL VXV(REM,VEM,Q)
CALL UNITV(Q,UZDE)
CALL VXV(UZDE,UXDE,UYDE)
DO 2 J=1,3
CDE(1,J)=UXDE(J)
CDE(2,J)=UYDE(J)
2 CDE(3,J)=UZDE(J)
REMMAG=VMAG(REM)
VY=DOT(UYDE,VEM,3)
DO 3 I=1,3
3 OMGED(I)=UZDE(I)*VY/REMMAG
DO 4 I=1,3
RDV(I)=REV(I)-RED(I)
VDV(I)=VEV(I)-VED(I)
RDE(I)=-RED(I)
VDE(I)=-VED(I)
RDL(I)=REL(I)-RED(I)
4 VDL(I)=VEL(I)-VED(I)
C COMPUTE STATES IN ROTATING FRAME
CALL MXV(CDE,RDV,RDVD,3,3)
CALL VXV(OMGED,RDV,TEMP1)
DO 5 I=1,3
5 TEMP2(I)=VDV(I)-TEMP1(I)
CALL MXV(CDE,TEMP2,VDVD,3,3)
CALL MXV(CDE,RDE,RDED,3,3)
CALL VXV(OMGED,RDE,TEMP1)
DO 6 I=1,3
6 TEMP2(I)=VDE(I)-TEMP1(I)
CALL MXV(CDE,TEMP2,VDED,3,3)
CALL MXV(CDE,RDL,RDL,3,3)
CALL VXV(OMGED,RDL,TEMP1)
DO 8 I=1,3
8 TEMP2(I)=VDL(I)-TEMP1(I)
CALL MXV(CDE,TEMP2,VLD,3,3)
C COMPUTE ANGLES AT VEHICLE, EARTH AND MOON
CALL UNITV(REM,UREM)
CALL UNITV(RMV,URMV)
CALL UNITV(REV,UREV)
ANGV=DARCOS(DOT(URMV,UREV,3))/DTR
ANGM=DARCOS(-DOT(UREM,URMV,3))/DTR

```

ANGE=180.0-(ANGV+ANGM)

RETURN

END

```

SUBROUTINE PTRAJ
  IMPLICIT REAL*8(A-H,L-M,O-Z)
  DIMENSION RSV(3),VSV(3),REV(3),VEV(3),RMV(3),VMV(3),RSE(3),
  1VSE(3),RSM(3),VSM(3),RDVD(3),VDVD(3),RDSD(3),VDSD(3),
  2RDMD(3),VDMD(3),PV(6),RSL1(3),VSL1(3),RDL1D(3),VDL1D(3)
  DIMENSION REM(3),VEM(3)
  COMMON/TDATA/TO,T,H,RSV,VSV,REV,VEV,RMV,VMV,RSE,VSE,RSM,VSM,
  1REM,VEM,RSL1,VSL1,RDVD,VDVD,RDSD,VDSD,RDMD,VDMD,
  2RDL1D,VDL1D,PV,LM,LDM,ANGV,ANGE,ANGS
  COMMON/CONST/MS,ME,MM,GL1,AUM,UTIME,UVELM,ERRMAX,DTR
  COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
  TDAY=(T-TO)*UTIME
  HDAY=H*UTIME
  RSVMAG=VMAG(RSV)
  REVMAG=VMAG(REV)
  RMVMAG=VMAG(RMV)
  WRITE(6,101)
  WRITE(6,100)T,TDAY,H,RSVMAG,REVMAG,RMVMAG
  C PRINT STATES IN INERTIAL S-FRAME
  WRITE(6,100)RSV,VSV
  WRITE(6,100)REV,VEV
  WRITE(6,100)RMV,VMV
  WRITE(6,101)
  WRITE(6,100)RSE,VSE
  WRITE(6,100)RSM,VSM
  WRITE(6,100)REM,VEM
  WRITE(6,100)RSL1,VSL1
  C WRITE(6,101)
  PRINT STATES IN ROTATING D-FRAME
  WRITE(6,100)RDVD,VDVD
  WRITE(6,100)RDSD,VDSD
  WRITE(6,100)RDMD,VDMD
  WRITE(6,100)RDL1D,VDL1D
  WRITE(6,101)
  WRITE(6,102)ANGV,ANGE,ANGS,HDAY
  C IF(IPV.EQ.0) GO TO 1
  PRINT PRIMER VECTOR
  WRITE(6,100)PV
  WRITE(6,103)LM,LDM
  1 CONTINUE
  WRITE(6,104)
  100 FORMAT(1H ,1P6D20.11)
  101 FORMAT(1H )
  102 FORMAT(1H ,1P4D20.11)
  103 FORMAT(1H ,1P2D20.11)
  104 FORMAT(1H0,'-----')
  RETURN
  END

```

```

SUBROUTINE PTRAJ3
  IMPLICIT REAL*8(A-H,L-M,O-Z)
  DIMENSION REV(3),VEV(3),RMV(3),VMV(3),REM(3),VEM(3),REL(3),
  IVEL(3),RDVD(3),VOVD(3),RDED(3),VDED(3),RDLD(3),VDLD(3),PV(6)
  COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
  COMMON/TDATA3/T0,T,H,REV,VEV,RMV,VMV,REM,VEM,REL,VEL,RDVD,
  IVDVD,RDED,VDED,RDLD,VDLD,PV,LM,LDM,ANGV,ANGE,ANGM
  TDAY=(T-T0)*UTIME
  HDAY=H*UTIME
  REVMAG=VMAG(REV)
  RMVMAG=VMAG(RMV)
  WRITE(6,101)
  WRITE(6,100)T,TDAY,H,HDAY,REVMAG,RMVMAG
  PRINT STATES IN INERTIAL E-FRAME
  WRITE(6,100)REV,VEV
  WRITE(6,100)RMV,VMV
  WRITE(6,100)REM,VEM
  WRITE(6,100)REL,VEL
  WRITE(6,101)
  PRINT STATES IN ROTATING D-FRAME
  WRITE(6,100)RDVD,VDVD
  WRITE(6,100)RDED,VDED
  WRITE(6,100)RDLD,VDLD
  WRITE(6,102)ANGV,ANGE,ANGM
  IF(IPV.EQ.0) GO TO 1
  WRITE(6,101)
  WRITE(6,100)PV
  WRITE(6,103)LM,LDM
100 FORMAT(1H ,1P6D20.11)
101 FORMAT(1H )
102 FORMAT(1H ,1P3D20.11)
103 FORMAT(1H ,1P2D20.11)
  1 CONTINUE
  RETURN
  END

```

SUBROUTINE FDATA

IMPLICIT REAL\*8(A-H,K-M,O-Z)

DIMENSION RSV(3),VSV(3),REV(3),VEV(3),RMV(3),VMV(3),RSE(3),  
1VSE(3),RSM(3),VSM(3),RSL1(3),VSL1(3),RDVD(3),VDVD(3),RDSD(3),  
2VDSD(3),RDMD(3),VDMD(3),RDL1D(3),VDL1D(3),PV(6),  
3DUM(9),REM(3),VEM(3)

COMMON/TDATA/TO,T,H,RSV,VSV,REV,VEV,RMV,VMV,RSE,VSE,RSM,VSM,  
1REM,VEM,RSL1,VSL1,RDVD,VDVD,RDSD,VDSD,RDMD,VDMD,  
2RDL1D,VDL1D,PV,LM,LDM,ANGV,ANGE,ANGS

C EACH RECORD CONSISTS OF 100 DOUBLE PRECISION UNFORMATED NUMBERS  
C FILED SEQUENTIALLY. THE LAST 15 NUMBERS ARE BLANKS FILLED WITH  
C ZEROS AND MAY BE USED FOR FUTURE EXPANSION.

DO 1 I=1,15

1 DUM(I)=0.

RSVMAG=VMAG(RSV)

VSVMAG=VMAG(VSV)

REVMAG=VMAG(REV)

VEVMAG=VMAG(VEV)

RMVMAG=VMAG(RMV)

VMVMAG=VMAG(VMV)

WRITE(8)T,H,RSV,VSV,REV,VEV,RMV,VMV,RSE,VSE,RSM,VSM,REM,VEM,  
1RSL1,VSL1,RDVD,VDVD,RDSD,VDSD,RDMD,VDMD,RDL1D,VDL1D,  
2PV,LM,LDM,ANGV,ANGE,ANGS,RSVMAG,VSVMAG,REVMAG,VEVMAG,RMVMAG,  
3VMVMAG,DUM

RETURN

END

```

SUBROUTINE FDATA3
  IMPLICIT REAL*8(A-H,K-M,O-Z)
  DIMENSION REV(3),VEV(3),RMV(3),VMV(3),REM(3),VEM(3),REL(3),
1VEL(3),RDVD(3),VDVD(3),RDED(3),VDED(3),RDLD(3),VDLD(3),PV(6),
2DUM(14)
  COMMON/TDATA3/TO,T,H,REV,VEV,RMV,VMV,REM,VEM,REL,VEL,RDVD,
1VDVD,RDED,VDED,RDLD,VDLD,PV,LM,LDM,ANGV,ANGE,ANGM
  DO 1 I=1,14
1 DUM(I)=0.
  REVMAG=VMAG(REV)
  VEV MAG=VMAG(VEV)
  RMVMAG=VMAG(RMV)
  REMMAG=VMAG(REM)
  VEMMAG=VMAG(VEM)
  WRITE(8)T,H,REV,VEV,RMV,VMV,REM,VEM,REL,VEL,RDVD,VDVD,RDED,
1VDED,RDLD,VDLD,PV,LM,LDM,ANGV,ANGE,ANGM,DUM
  RETURN
  END

```

```

SUBROUTINE COMAUG(F,G,TSI,LT,V,L,FG,GG)
  IMPLICIT REAL*8(A-H,L-N,O-Z)
  DIMENSION TSI(3),G(3),LTLI(3,3),LT(3,3),L(3,3),NU(3),GG(3),
1 TEMP1(3,3),TEMP2(3,3),DUM(3),V(3,3)
  CALL MTRANS(LT,L,3,3)
  CALL MXM(LT,V,TEMP1,3,3,3)
  CALL MXM(TEMP1,L,TEMP2,3,3,3)
  CALL INVERT(TEMP2,LTLI)
  CALL MXM(LTLI,LT,TEMP1,3,3,3)
  CALL MXM(TEMP1,V,TEMP2,3,3,3)
  CALL MXV(TEMP2,G,NU,3,3)
  DO 1 I=1,3
1  NU(I)=-NU(I)
  FG=F+DOT(NU,TSI,3)
  CALL MXV(L,NU,DUM,3,3)
  DO 2 I=1,3
2  GG(I)=G(I)+DUM(I)
  WRITE(6,100)FG,GG
  WRITE(6,101)
100 FORMAT(1H ,T8,'FG',T28,'GG'/1H ,1P4D20.11)
101 FORMAT(1H0,T8,'-----')
  RETURN
  END

```

```
SUBROUTINE COMDX(LTS,LS,TSIS,DX)
  IMPLICIT REAL*8(A-H,L,O-Z)
  DIMENSION LTS(3,3),LS(3,3),TSIS(3),DX(3),TEMP1(3,3),TEMP2(3,3)
  CALL MXM(LTS,LS,TEMP1,3,3,3)
  CALL INVERT(TEMP1,TEMP2)
  CALL MXM(LS,TEMP2,TEMP1,3,3,3)
  CALL MXV(TEMP1,TSIS,DX,3,3)
  DO I =1,3
1  DX(I)=-DX(I)
  RETURN
  END
```



```
SUBROUTINE UPX(XD,FD,TESTRD,GD,TSID,LTD,LD,FGD,GGD,S11D,S12D,  
1S21D,S22D,UVID,UVFD,X,F,TESTR,G,TSI,LT,L,FG,GG,S11,S12,S21,S22,  
2UVI,UVF)
```

```
  IMPLICIT REAL*8(A-H,L,N,O-Z)  
  DIMENSION XD(3),GD(3),TSID(3),LTD(3,3),LD(3,3),X(3),G(3),  
1TSI(3),LT(3,3),L(3,3),S11D(3,3),S12D(3,3),S21D(3,3),S22D(3,3),  
2S11(3,3),S12(3,3),S21(3,3),S22(3,3),GG(3),UVID(3),UVFD(3),  
3UVI(3),UVF(3),GGD(3)
```

```
  F=FD
```

```
  TESTR=TESTRD
```

```
  FG=FGD
```

```
  DO 1 I=1,3
```

```
  X(I)=XD(I)
```

```
  G(I)=GD(I)
```

```
  TSI(I)=TSID(I)
```

```
  GG(I)=GGD(I)
```

```
  UVI(I)=UVID(I)
```

```
  UVF(I)=UVFD(I)
```

```
  DO 1 J=1,3
```

```
  LT(I,J)=LTD(I,J)
```

```
  L(I,J)=LD(I,J)
```

```
  S11(I,J)=S11D(I,J)
```

```
  S12(I,J)=S12D(I,J)
```

```
  S21(I,J)=S21D(I,J)
```

```
  S22(I,J)=S22D(I,J)
```

```
1 CONTINUE
```

```
  RETURN
```

```
  END
```

```

SUBROUTINE PVEC(T,R,V,PVO,STM,PV,LM,LDM)
IMPLICIT REAL*8(A-H,L,O-Z)
DIMENSION R(3),V(3),PVO(6),PV(6),STM(6,6),DL(3),DLD(3),
1RTM(3),VTM(3),PVTM(3),PVDTM(3),STNTM(6,6)
COMMON/FLAG/IMTX,IPV,IPTRAJ,IPVTM,IFILE,ITER,ITAR
COMMON/CTM/TTM,RTM,VTM,LTM,LDTM,PVTM,PVDTM,STNTM
CALL MXV(STM,PVO,PV,6,6)
DO 3 I=1,3
DL(I)=PV(I)
3 DLD(I)=PV(I+3)
LM=VMAG(DL)
LDM=DOT(DL,DLD,3)/LM
IF(IPVTM.LE.0) GO TO 2
IF(LM.LE.LTM) GO TO 2
LTM=LM
LDTM=LDM
TTM=T
DO 1 I=1,3
RTM(I)=R(I)
VTM(I)=V(I)
PVTM(I)=DL(I)
PVDTM(I)=DLD(I)
DO 1 J=1,3
STNTM(I,J)=STM(I,J)
STNTM(I,J+3)=STM(I,J+3)
STNTM(I+3,J)=STM(I+3,J)
1 STNTM(I+3,J+3)=STM(I+3,J+3)
2 CONTINUE
RETURN
END

```

```

SUBROUTINE RVEMV(RSV,VSV,RSE,VSE,RSM,VSM,REV,VEV,RMV,VMV,
1 REM,VEM)
  IMPLICIT REAL*8(A-H,O-Z)
  DIMENSION RSV(3),VSV(3),RSE(3),VSE(3),RSM(3),VSM(3),REV(3),
1 VEV(3),RMV(3),VMV(3),REM(3),VEM(3)
  DO 1 J=1,3
    REV(J)=-RSE(J)+RSV(J)
    VEV(J)=-VSE(J)+VSV(J)
    RMV(J)=-RSM(J)+RSV(J)
    VMV(J)=-VSM(J)+VSV(J)
    REM(J)=-RSE(J)+RSM(J)
1 VEM(J)=-VSE(J)+VSM(J)
  RETURN
  END

```

```
C SUBROUTINE MXV(A,V,VN,M,N)
  IMPLICIT REAL*8(A-H,O-Z)
  MATRIX VECTOR MULTIPLICATION A(M,N) V(N)=VN(M)
  DIMENSION A(M,N),V(N),VN(M)
  DO 4 I=1,M
    S=0.
    DO 3 J=1,N
3   S=S+A(I,J)*V(J)
4   VN(I)=S
  RETURN
  END
```

```
C
SUBROUTINE VVT(V,VV, IDIM)
  IMPLICIT REAL*8(A-H,O-Z)
  OUTER PRODUCT OF 2 VECTORS OF DIMENSION IDIM
  DIMENSION V(IDIM),VV(IDIM, IDIM)
  DO 1 I=1, IDIM
  DO 1 J=1, IDIM
1  VV(I, J)=V(I)*V(J)
  RETURN
  END
```

C  
DOUBLE PRECISION FUNCTION DOT(V1,V2, IDIM)  
IMPLICIT REAL\*8(A-H,O-Z)  
DOT PRODUCT OF 2 VECTORS  
DIMENSION V1(IDIM),V2(IDIM)  
S=0.  
DO 1 I=1, IDIM  
S=S+V1(I)\*V2(I)  
DOT=S  
RETURN  
END

```
C  DOUBLE PRECISION FUNCTION VMAG(V)
    IMPLICIT REAL*8(A-H,O-Z)
    MAGNITUDE OF VECTORS OF DIMENSION IDIM
    DIMENSION V(3)
    S=0.
    DO 1 I=1,3
    S=S+V(I)**2
    VMAG=DSQRT(S)
    RETURN
    END
```

```

SUBROUTINE INVERT(B,BI)
  IMPLICIT REAL*8(A-H,O-Z)
  INVERSION OF A 3X3 MATRIX
  DIMENSION B(3,3),BI(3,3)
  D=B(1,1)*(B(2,2)*B(3,3)-B(2,3)*B(3,2))+B(2,1)*(B(1,3)*B(3,2)-
1 B(1,2)*B(3,3))+B(3,1)*(B(1,2)*B(2,3)-B(1,3)*B(2,2))
  BI(1,1)=(B(2,2)*B(3,3)-B(2,3)*B(3,2))/D
  BI(2,1)=(B(2,3)*B(3,1)-B(2,1)*B(3,3))/D
  BI(3,1)=(B(2,1)*B(3,2)-B(2,2)*B(3,1))/D
  BI(1,2)=(B(1,3)*B(3,2)-B(1,2)*B(3,3))/D
  BI(2,2)=(B(1,1)*B(3,3)-B(1,3)*B(3,1))/D
  BI(3,2)=(B(1,2)*B(3,1)-B(1,1)*B(3,2))/D
  BI(1,3)=(B(1,2)*B(2,3)-B(1,3)*B(2,2))/D
  BI(2,3)=(B(1,3)*B(2,1)-B(1,1)*B(2,3))/D
  BI(3,3)=(B(1,1)*B(2,2)-B(1,2)*B(2,1))/D
  RETURN
  END

```



```
SUBROUTINE UNITV(A,UA)
  IMPLICIT REAL*8(A-H,O-Z)
  DIMENSION A(3),UA(3)
  S=0.
  DO 1 J=1,3
1  S=S+A(J)**2
  AMAG=DSQRT(S)
  DO 2 J=1,3
2  UA(J)=A(J)/AMAG
  RETURN
  END
```

```
C SUBROUTINE VxV(V1,V2,V3)
  IMPLICIT REAL*8(A-H,O-Z)
  CROSS PRODUCT OF 2 3-DIMENSIONAL VECTORS
  DIMENSION V1(3),V2(3),V3(3)
  V3(1)=V1(2)*V2(3)-V1(3)*V2(2)
  V3(2)=V1(3)*V2(1)-V1(1)*V2(3)
  V3(3)=V1(1)*V2(2)-V1(2)*V2(1)
  RETURN
  END
```

```
C  SUBROUTINE MTRANS(A,B,IROW,ICOL)
    IMPLICIT REAL*8(A-H,O-Z)
    TRANSPOSE A MATRIX OF IROW X ICOL
    DIMENSION A(IROW,ICOL),B(ICOL,IROW)
    DO 1 I=1,IROW
    DO 1 J=1,ICOL
1  B(J,I)=A(I,J)
    RETURN
    END
```

```
C
SUBROUTINE MXM(A,B,C,K,L,M)
  IMPLICIT REAL*8(A-H,O-Z)
  MATRIX MULTIPLICATION A(K,L) B(L,M)=C(K,M)
  DIMENSION A(K,L),B(L,M),C(K,M)
  DO 4 I=1,K
  DO 4 J=1,M
  S=0.
  DO 3 N=1,L
  3 S=S+A(I,N)*B(N,J)
  4 C(I,J)=S
  RETURN
  END
```

## REFERENCES

1. Pu, C. L. and T. N. Edelbaum, "Four-Body Trajectory Optimization," C. S. Draper Laboratory Report No. R-778, December, 1973.
2. Pu, C. L. and T. N. Edelbaum, "Four-Body Trajectory Optimization," Paper No. 74-169, presented at AIAA 12th Aerospace Sciences Meeting, Washington, D. C., January 30 - February 1, 1974.
3. Goodyear, W. H., "A General Method for the Computation of Cartesian Coordinates and Partial Derivatives of the Two-Body Problems," NASA CR-552, September, 1968.
4. D'Amario, L. A., "Minimum Impulse Three-Body Trajectories," Report No. T-593, C. S. Draper Laboratory, June, 1973.
5. Farquhar, R. W., "The Control and Use of Libration Point Satellites," Report SUDAAR No. 350, Dept. of Aeronautics and Astronautics, Stanford University, July, 1968.