Final Report for
CHEBYCHEV TRAJECTORY OPTIMIZATION
PROGRAM (CHEBYTOP)

D2-121308-1

by
D. W. Hahn, F. T. Johnson, and B. F. Itzen

July 1969

Prepared under Contract NAS2-5185
THE BOEING COMPANY
Aerospace Systems Division
Seattle, Washington

for
MISSION ANALYSIS DIVISION, MOFFETT FIELD, CALIFORNIA
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.</td>
<td>SUMMARY</td>
<td>1</td>
</tr>
<tr>
<td>II.</td>
<td>INTRODUCTION</td>
<td>3</td>
</tr>
<tr>
<td>III.</td>
<td>ANALYSIS</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Definition of the Problem</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Mathematics of the Variable Thrust Solution</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Mathematics of the Constant Isp Solution</td>
<td>9</td>
</tr>
<tr>
<td>IV.</td>
<td>PROGRAM ORGANIZATION</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Input</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Output</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Adjustments and Modifications</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Subroutine Descriptions</td>
<td>18</td>
</tr>
<tr>
<td>V.</td>
<td>PROGRAM LIMITATIONS</td>
<td>45</td>
</tr>
<tr>
<td>IV.</td>
<td>REFERENCES</td>
<td>46</td>
</tr>
<tr>
<td>APPENDICES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Dictionary of Terms</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Operating Instructions</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Sample Cases</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Listings</td>
<td></td>
</tr>
</tbody>
</table>
I. SUMMARY

A digital computer program called the Chebychev Trajectory Optimization Program (CHEBYTOP) is described. The purpose of CHEBYTOP is to provide an analysis tool that will enable the user to rapidly conduct the optimization and parametric analysis of interplanetary missions employing electrically propelled spacecraft. A new analytical and numerical optimization algorithm, called the Chebychev Optimization Method, is used. The resulting computer program operates with greater speed and reliability than other existing methods of comparable accuracy and versatility. In some cases the operating speed is more than a factor of ten faster than conventional methods.

The program is specialized to solve for interplanetary trajectories, with the planets themselves assumed massless. Planetary positions and velocities are computed in the program from stored orbital elements. The elements of the nine planets are available, with a tenth set of elements left for the user to specify. The latter option is useful when designing such missions as an asteroid probe or an out of the ecliptic probe. Either rendezvous (matching the planet's orbital velocity about the sun), flyby (matching only the planet's position at the terminal time) or excess velocity trajectories may be obtained. Excess velocity trajectories are those which arrive or depart from a massless planet with a specified velocity relative to the planet.

CHEBYTOP will compute either variable thrust or constrained thrust, multiple-coast trajectories. Variable thrust answers are computed by the Chebychev Optimization Method. Thrust-limited results are obtained by a prediction scheme which uses the variable thrust answer as a basis for approximating multiple-coast results.

The user has an option of specifying the nature of the power source assumed to drive the electric thrusters. The power can be assumed
either constant or variable. The variable power option approximates a solar power source, causing the power to vary as a function of the radius from the sun. Either power option can be used with either variable or constrained thrust performance.

The program is self-starting. In-other-words, only those boundary conditions of immediate concern to the user need be input. Guesses of undetermined multipliers and control variable state histories are unnecessary.

Coding was done in FORTRAN IV for the IBM 360 series computers. The core requirements are 150,000 bytes. No auxiliary tapes are needed.
II. INTRODUCTION

The development of the Chebychev Trajectory Optimization Program (CHEBYTOP) was motivated by the need for a fast operating and reliable low-thrust trajectory design tool. The determination of optimum finite-thrust trajectories in a central force field normally requires solving a problem in the calculus of variations. Conventional techniques for attacking the problem require large amounts of computer time. CHEBYTOP makes use of a new analytical and numerical optimization method, described in Reference 1, called the Chebychev Optimization Method. Through the use of approximating polynomials the method reduces the variational problem to one of ordinary calculus. Computational speed has been achieved because the method eliminates the need for time-consuming numerical integration and provides for rapid computation of the derivatives of the payoff. In addition, since the method was developed with computational efficiency in mind, the computer program is carefully organized throughout to minimize the number of operations per iteration.

The solution of the optimum thrust-limited problem is divided into two steps within CHEBYTOP. Hence the program is composed of two major sub-programs. The first, VTMODE, solves the unconstrained continuous-thrust case (Variable Thrust Mode) using the Chebychev Optimization Method of Reference 1. The second sub-program, TCTPRE, solves for the constant specific impulse, thrust-limited case, using the prediction scheme described in Reference 2. The mathematical details of these two algorithms are not contained in this document. The Analysis section following presents only the modifications to References 1 and 2 which are incorporated in CHEBYTOP. Copies of these references, if not readily available, can be obtained from the American Institute of Aeronautics and Astronautics.

An attempt was made to create a program which would determine an optimum trajectory given only the required mission design parameters
and which would consistently achieve accurate convergence. Therefore CHEBYTOP is constructed so that it can be used as a subroutine of a mission analysis master program. Hence, auxiliary parameters, such as loop counters, convergence criteria, and mesh points, are program constants. Although CHEBYTOP has proven to be quite reliable, a section is included under Program Organization called Adjustments and Modifications which tells the user how to perform minor changes.
III. ANALYSIS

DEFINITION OF PROBLEM

The two-body equations of motion of an interplanetary vehicle are

\[ \ddot{x} + \frac{kx}{r^3} = a \quad 0 \leq t \leq T \]

where \( x = x(t) \) is the position vector of the vehicle, \( a = a(t) \) is the applied acceleration vector, \( k \) is the gravitational constant of the sun, and \( r = |x| \). Let \( p = p(r) \) be the power level of the power plant of the vehicle with \( p_o = p(1) \). Then the basic problem solved is that of minimizing

\[ J = \int_0^T |a|^2 \frac{p_o}{p(r)} \, dt \]

subject to boundary conditions on \( x \) and thrusting constraints on \( a \).

\( x(o), x(T), \) and \( T \) are always assumed to be fixed. The program has three options regarding \( \dot{x}(o), \dot{x}(T) \). They may be fixed (rendezvous) or vary freely (flyby), or vary over a one or two-dimensional sphere (fixed hyperbolic excess speeds).

Two acceleration options are available. The acceleration, \( a \), may be completely unconstrained (variable thrust) or else \( a \) may be assumed to derive from a constant Isp engine with shut down and start up capability. In the latter case we must have

\[ |a| = \frac{a_o}{\mu} \frac{p}{p_o} \sigma(t) \quad \dot{\mu}(t) = -\frac{a_o}{c} \frac{p}{p_o} \sigma(t) \]

Here \( a_o \) is the initial acceleration of the vehicle at 1 au, \( c \) the exhaust velocity, \( \mu \) the relative mass of the vehicle, and
during powered phase

\[ \sigma(t) = \begin{cases} 1 & \text{during powered phase} \\ 0 & \text{during coast} \end{cases} \]

The program solves the variable thrust optimization problem in exact fashion; that is except for roundoff and truncation errors (and perhaps inadequate convergence), the value of \( J \) obtained by the program can be assumed to be the solution of the mathematical problem as posed. The constant Isp solution, on the other hand, is achieved using certain approximations to the original mathematical problem, and therefore its validity must be ascertained on an empirical basis.

**MATHEMATICS OF VARIABLE THRUST SOLUTION**

The basic mathematical foundation of this part of the program is contained in Reference (1). Several modifications to the development found there are necessary owing to the inclusion of variable power and polynomial patching capabilities in the program. Hence we shall assume the user is familiar with the material in Reference (1), and simply mention modifications here.

It was noted in Reference 1 that for long trajectories it is more efficient to represent the position vector time history by several small order polynomials matching position and velocity at junctions rather than one large order polynomial. This program incorporates that suggestion and provides for six polynomials of 9th order each. For the starting solution each polynomial covers the same elapsed time, however upon obtaining convergence the times are redistributed to obtain a better representation of the trajectory. \( J \) then becomes a weighted sum of the performance index for each leg,

\[ J = \sum_{i=1}^{n_1} 2 \tau_i \phi_i \]

Here \( \tau_i \) is the elapsed time for the \( i \)th leg and \( \phi_i \) is the performance
index of the ith leg as defined by Equation (5) in Reference (1).
All differentiation formulas of Reference 1 can be applied separately
to each $P_i$.

A redefined interpolation procedure was used in Reference 1 to reduce
execution time computing partial derivatives. This procedure matched
the polynomial derivatives at endpoints rather than interpolating the
second and next to last Chebychev points. The inclusion of solar
power eliminates this savings, however, and $P_x, P_{xx}$ are computed according to Equation (29), Reference 1.

The program performs its optimizing iterations using Gauss' method.
A special one dimensional search routine using the secant method
has been included, however, to insure that each iteration yields a
smaller payoff than that of the previous iteration. Gauss' method
seems to find the proper direction in which to take a step, but
occasionally overshoots the step size. Newton's method is also used,
but confined to situations where tracking is desired.

The secant method behaves as follows. Let $f(s)$ be continuously differentiable on $(-\infty, \infty)$. Let $s_1$ and $s_2$ be two first guesses to a minimum $s_o$ of $f$ on this interval. Then $s_o$ may be approximated by the sequence $[s_n]$ defined recursively by the equation
\[
s_{n+1} = (s_{n-1} f'(s_n) - s_n f'(s_{n-1}))/((f'(s_n) - f'(s_{n-1}))
\]

The variable thrust algorithm of Reference 1 must be modified to account
for solar power. In particular, equations (16), (17), and (18) of
the reference are affected. Let $Q$ be a column vector with elements
$Q^\gamma = [p_0/p (r(s^\gamma))]^{1/2}$. Let $AV(m), m = 1, \ldots, n_d$ be a set of column vectors defined by
\[
AV(m) = A^{-1} BAX(m) + Y(m)
\]
(Note that $AV(m)/\tau^2$ is the variable thrust acceleration vector). Now let $G(m)$, $m = 1, \ldots, nd$ be a set of column vectors defined by

$$G(m) = QAV(m)$$

i.e.

$$G(m)^\tau = Q^\tau AV(m)^\tau$$

Then (16) becomes

$$P = 1/2 \sum_{m=1}^{nd} G(m)^T F G(m)$$

Setting $R(m) = FG(m)$, (17) and (18) become respectively

$$P_x(i) = \sum_{m=1}^{nd} G_x(m,i)^T R(m)$$

$$P_{xx}(i,j) = \sum_{m=1}^{nd} G_x(m,i)^T F G_x(m,j) + R(m) G_{xx}(m,i,j)$$

Setting $H(i,j) = Y_x(i,j) + \frac{Qx(j)}{Q} AV(i)$ and $S(m) = QR(m)$, we obtain

$$P_x(i) = (A^{-1}BA)^T S(i) + \sum_{m=1}^{nd} H(m,i) S(m)$$

$$P_{xx}(i,j) = (A^{-1}BA)^T Q^T F Q (A^{-1}BA) \delta_{i,j}$$

$$+ (A^{-1}BA)^T Q^T F Q H(i,j)$$

$$+ H(j,i)^T Q^T F Q (A^{-1}BA)$$

$$+ \sum_{m=1}^{nd} H(m,i) Q^T F Q H(m,j)$$

$$+ R(m)^T G_{xx}(m,i,j)$$

8
The last term is rather complicated. For Gauss' method it is neglected. For Newton's method we set \( G_{xx}(m,i,j) = Q_{Yxx}(m,i,j) \) which is exact in the case of constant power but slightly inaccurate in the case of variable power.

**MATHEMATICS OF CONSTANT ISP SOLUTION**

The basic mathematics of the program's constant ISP prediction scheme are contained in Reference 2. Only the first side condition has been employed in the present program.

The scheme has been modified to include a solar power option. One more assumption has been made in this case, namely that the power level time history is roughly the same for both the variable and constant ISP modes. Then the modification reduces to a simple transformation of variables, as described below.

Let \( p(t) \) be the power time history as determined by the variable thrust program. With the above assumption Equations (6) and (7) of Reference 2 become respectively

\[
\Delta J = \int_{t_0}^{t_1} \left| a_c \right|^2 \frac{p_o}{p(t)} \, dt
\]

\[0 = \int_{t_0}^{t_1} \left| a_c \right| a_v \left| \frac{p_o}{p(t)} \right| \, dt - Jv\]

where \( |a_c| = \frac{a_o}{\mu(t)} \frac{p(t)}{p_o} \sigma(t) \) and \( \dot{\mu}(t) = -\frac{a_o}{c} \frac{p(t)}{p_o} \sigma(t) \)

Here \( a_o \) is the initial acceleration the vehicle would have at a radius of 1 au. Now set \( h_v(t) = a_v(t) \left( \frac{p_o}{p(t)} \right) \)

and \( h_c(t) = a_c(t) \left( \frac{p_o}{p(t)} \right) \), and \( \tau(t) = \int_{t_0}^{t} \frac{p(t)}{p_o} \, dt. \)

The above equations now become

\[ \text{9} \]
\[ \Delta J = \int_{\tau_o}^{\tau_1} |h_c|^2 \, d\tau \]

\[ 0 = \int_{\tau_o}^{\tau_1} |h_c| \left| h_c \right| d\tau - J_v \]

\[ |h_c| = \frac{a_0}{\mu(\tau)} \sigma(\tau) \]

\[ \frac{du}{d\tau} = -\frac{a_0}{c} \sigma(\tau) \]

where \( \tau_o = \tau(t_o) \) and \( \tau_1 = \tau(t_1) \)

These equations are identical to the previous ones except for a change in variable names, and a solution may be obtained with precisely the same algorithm.
IV. PROGRAM ORGANIZATION

INPUT

The program is intended to operate as a subroutine of a more general mission analysis main program. The form of the input subroutine, therefore, is left to the user. Variable thrust answers are obtained by calling the VTMODE subroutine and thrust limited answers by following CALL VTMODE with a call of the prediction subroutine, TCTPRE. Perhaps the simplest mode of program operation is to compile a MAIN program which contains one or more calls of subroutine VTMODE and TCTPRE. The sample cases included in Appendix C use this form of input.

Variable Thrust Input

The call list to VTMODE to produce a variable thrust trajectory is as follows:

CALL VTMODE (NC,NR,NP1,NP2,NB1,NB2,D1,D2,HV1,HV2,NPOW,NA,NT)

The arguments of the subroutine are:

NC    is the number of dimensions (2 or 3), except for out-of-the ecliptic-probes or similar missions which require a three-dimensional starting trajectory, in which case set NC = 1.

NR    is the number of full revolutions around the sun (NR = 0 for travel angles less than 360°, NR < 3, see Limitations)

NP1   is the number of the departure planet, and varies from 1 for Mercury to 9 for Pluto*

* A tenth planet number is available in subroutine EPHEM so that the user can assign an additional set of orbital elements (see sample cases), however the eccentricity of the additional orbit should be kept less than .5 (see Limitations).
NP2 is the analogous number of the arrival planet

NB1 indicates the initial constraint on excess velocity

\begin{align*}
0 & \text{ for flyby} \\
1 & \text{ for fixed excess velocity}
\end{align*}

NB2 indicates the final constraint on excess velocity

\begin{align*}
2 & \text{ for rendezvous}
\end{align*}

D1 is the Julian date of departure (7 significant figures, floating point)

D2 is the Julian date of arrival (7 significant figures, floating point)

HV1 is the initial hyperbolic excess velocity relative to the departure planet (if NB1 = 1) in km/sec.

HV2 is the final hyperbolic excess velocity relative to the arrival planet (if NB2 = 1) in km/sec.

NPOW indicates the type of power supply

\begin{align*}
0 & \text{ for constant} \\
1 & \text{ for variable}
\end{align*}

NA is a control on the accuracy of the acceleration time history

\begin{align*}
0 & \text{ for approximate solution} \\
1 & \text{ for more accurate acceleration time history}
\end{align*}

NT is a control which allows tracking from a previously computed trajectory

\begin{align*}
0 & \text{ no tracking, uses standard starting solution} \\
1 & \text{ tracking}
\end{align*}

**Constant Thrust Input**

The call list to TCTPRE to predict a constant thrust trajectory is

\begin{verbatim}
CALL TCTPRE(TW,SI)
\end{verbatim}

The arguments of the subroutine are as follows:

\begin{align*}
TW & \text{ is the initial* thrust to weight ratio (dimensionless)}
\end{align*}

\begin{footnotesize}
\begin{itemize}
\item[*] In the case of variable power, TW is relative to thrust level at 1 a.u.
\end{itemize}
\end{footnotesize}
SI is the specific impulse in seconds

The following are available as output of the two subroutines through the common block /BDYP/: 

BDY is a $3 \times 3 \times 2$ array containing the vectors $(\dot{x}, a, \dot{a})$ at each endpoint (e.g., BDY $(M,2,1) = a(M)$, $M = 1,3$). BDY is computed for the variable thrust solution only.

PV is the variable thrust value of $\int_0^T a_0^2 \frac{P_0}{p} \, dt$ in au $^2$/yr.$^3$

PC is its constant thrust equivalent.

The values of $\dot{a}$ are somewhat suspect. The dimensions of these quantities are astronomical units and years.

For example a MAIN program to compute an 800 day Jupiter flyby using solar power is given below.

```fortran
COMMON/BDYP/BDY(3,3,2),PV,PC
CALL VTMODE(3,0,3,5,2,0,2444140.,2444940,0.,0.,1,0,0)
CALL TCTPRE(.0000800,5000.)
STOP
END
```

This trajectory is one of the sample cases given in Appendix C.

Comments on Input Constant Selection

The program is specialized to compute trajectories between the planets. It therefore takes advantage of the low orbital inclinations (relative to the ecliptic plane) by solving each trajectory in two dimensions, then adding the third dimension, if desired, in a final iteration. In cases where orbital inclinations are large, such as for out-of-the-ecliptic probes, it may be necessary to perform all iterations in three
dimensions, including the computation of the starting solution. Setting $NC = 1$ causes the starting solution to be computed in three dimensions and all subsequent iterations also. This option obviously increases run times significantly, so should not be used for standard interplanetary transfers.

It is necessary to anticipate the in-plane transfer angle desired. There may exist local optimum solutions for each multiple of one revolution about the sun. Normally the global optimum has the maximum number of revolutions consistent with a smooth transition between the energy levels of the beginning and final orbits. For $NT = 0$ the built-in starting solution fits a smooth, usually monotonic, curve through the endpoints, with the number of solar revolutions specified by $NR$ in the call list. Subsequent iterations will not change this basic revolution number. The user is advised to either refer to planetary ephemerides, or to carefully compare input Julian dates with those of similar solutions he already has on hand, to avoid slip ups in setting the revolution counter, $NR$. If the tracking option ($NT = 1$) is used the counter $NR$ becomes irrelevant. The travel angle will vary continuously with $D1$ and $D2$ (assuming changes in these dates are sufficiently small). When tracking, all elements of the call list may change. However, the user should track so that the position vector time histories of successive trajectories are close. Otherwise some trajectories might take longer to obtain by tracking from a previous solution than by using the built in starting procedure. It is best to be on the conservative side when choosing a tracking step.

Trajectories with specified hyperbolic excess speeds and having travel angles greater than $360^\circ$ are generally more difficult to obtain from scratch than corresponding trajectories with rendezvous (or flyby) boundary conditions. Therefore it is often more efficient to obtain a rendezvous solution with $NT = 0$ and then track hyperbolic speeds upwards with $NT = 1$. Finally, the user should always remember to obtain a trajectory with $NT = 0$ before setting $NT = 1$. 

14
The nature of the boundary conditions must be input through NB1 and NB2. If these constants are set equal to either 0 or 2 any values input for HV1 and HV2 will be ignored. Similarly, if a rendezvous result is called for, it is not sufficient to merely set HV1 or HV2 equal to zero. The settings on NB1 and NB2 tell the program to let the direction of the excess velocity be free, to optimize the direction of the excess velocity for minimum payoff, or to skip all consideration of excess velocity.

The Julian day numbers, D1 and D2, are limited to seven significant figures plus the decimal point. Therefore fractions of days, if input, will be ignored by the IBM 360. The excess velocities HV1 and HV2 are similarly limited.

Before using the variable power option, NPOW = 1, the user should note the form of the power profile in subroutine POWER. This profile was taken from Reference 3. The procedure for altering this profile is discussed in the following section.

The accuracy control, NA, has a very slight influence on the value of the payoff, for either variable thrust solutions (from VTMODE) or thrust limited solutions (from TCTPRE). It can, however, influence the accuracy of the thrust-acceleration time history. When NA is set equal to zero the convergence criterion is payoff improvement only. When NA = 1, after the payoff convergence criterion is satisfied, an additional test is performed on the acceleration time history. Experience has shown that a smooth acceleration profile is the most reliable indicator of the true optimum. Since the longer and more difficult trajectories will require multiple polynomials (as discussed in Section II Analysis) an excellent smoothness test is to look for acceleration discontinuities at the patch points. If these discontinuities exceed a specified amount one additional polynomial is added and more iterations are performed until the payoff convergence criterion is again met. The number of polynomials is increased again, but cannot exceed a total of six.
To reiterate: for preliminary design work and exploratory studies of mission feasibility and launch errors, the approximate mode (\(NA = 0\)) should be adequate, since the user can expect reliable payoff trends. The approximate mode may, if the acceleration profile is badly behaved, 1) result in extraneous thrust or coast periods when the prediction scheme (TCTPRE) is called, 2) slow down convergence in some cases involving large excess velocities at the boundaries, and 3) result in inaccurate values of the output array \textit{BDY}.

Note that, in the case of solar power, the initial thrust to weight ratio should be input at the level corresponding to 1 au from the sun, even though a trajectory may not start from the earth. The ratio of the power at any given radius to the power at 1 au is included in the acceleration relationships (see Analysis). Thus thrust to weight adjustments due to radius from the sun are made automatically.

\textbf{OUTPUT}

Two forms of variable thrust output are provided by the program; a short form and a longer more detailed form. The short form is printed automatically with every call of \texttt{VTMODE}. Samples of both forms are given in Appendix C.

The long form is obtained by calling subroutine \texttt{OUTCAL} after each call of \texttt{VTMODE} for which the long form is desired. For instance Jupiter solar-electric flyby mission is input as follows for the long form output:

```
COMMON/BDYP/BDY(3,3,2),PV,PC
CALL VTMODE(3,0,3,5,20,2444140.,2444940.,0.,0.,1,0,0)
CALL OUTCAL
CALL TCTPRE(.0000800,5000.)
STOP
END
```
Note that subroutine OUTCAL has no call list.

A call of TCTPRE also produces an automatic output, as shown in Appendix C, consisting of the input parameters and a listing of startup and shutoff times for the predicted constant thrust trajectory.

The following is a dictionary of output terms:

- **JV**: the performance index \( \text{m}^2/\text{sec}^3 \)
- **EXCESS VELOCITY AT DEPARTURE**: in km/sec
- **EXCESS VELOCITY AT ARRIVAL**: in km/sec
- **TIME**: tabulated at 26 equally spaced points in DAYS
- **X,Y,Z**: heliocentric ecliptic cartesian coordinates of 1950.0 in au's
- **R**: distance of spacecraft from the sun in au's
- **THETA**: vehicle heliocentric longitude measured in the ecliptic plane from the positive X-axis in DEGREES
- **PHI**: vehicle heliocentric latitude measured from the ecliptic plane in DEGREES
- **AX,AY,AZ**: components of the thrust acceleration parallel to the position coordinates in au/yr^2
- **MAG.A**: the magnitude of the thrust acceleration vector in au/yr^2
- **JC**: the predicted constant thrust performance index in \( \text{m}^2/\text{sec}^3 \)
- **P/Po**: power level relative to that at 1 au

**ADJUSTMENTS AND MODIFICATIONS**

Two program modules may be replaced by the user to change the mathematical model. The first module is subroutine EPHEM. The second consists of subroutines POWER and DPOWER. Subroutine EPHEM constructs position and velocity vectors for any of the planets as a function of an input argument.
(the Julian Date). As delivered, it obtains the position and velocity vectors from conic elements using a Kepler's equation solution good for small eccentricities only. Subroutines POWER and DPOWER are function subroutines which specify the power level and the first derivative of the power level, respectively, as a function of solar radial distance. The only restriction on the replacement of these modules is that the CALL lists must be unchanged. The quantity DELTA in VTMODE determines when convergence to an optimum trajectory has been achieved. (The definition of DELTA is given in Appendix A). For long, highly nonlinear trajectories an optimization process could conceivably terminate before the optimum trajectory is reached. In this case the relative change in payoff between two successive iterations will accidentally be less than DELTA. Therefore DELTA in Statement 22, must be made smaller.

NIT is the maximum number of iterations allowed for achieving an optimum in section 2 of VTMODE. NIT is varied automatically depending on the difficulty of trajectory. It is possible that the nominal values assigned to NIT are insufficient for extremely difficult trajectories, in which case these values should be increased.

There is no question of convergence in using the constant thrust prediction scheme. However, in the interest of increasing the speed of operation, or increasing the accuracy of approximation, the number of mesh points and the number of iterations may be changed. The number of mesh points is the variable NNN in subroutine TCTPRE. This number must be odd and not greater than 301 (unless the dimension statements in COMMON block/PXXPX/are also increased in subroutine TCTPRE and RESCAL). The number of iterations in the solution is controlled by the variable NSTEP in subroutine ROOTR. The accuracy level is approximately $2^{*}\text{-NSTEP}$. NSTEP is nominally equal to 14.

SUBROUTINE DESCRIPTIONS

This contains brief descriptions of the program subroutines. They are
arranged in almost alphabetical order, with the two principle routines VTMODE and TCTPRE appearing first. The index below is in alphabetical listing giving page locations.

### INDEX OF SUBROUTINES

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Page</th>
<th>Subroutine</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIGN</td>
<td>27</td>
<td>OUTPT1</td>
<td>36</td>
</tr>
<tr>
<td>AMAP *</td>
<td>37</td>
<td>OUTPUT</td>
<td>36</td>
</tr>
<tr>
<td>AVTEST</td>
<td>28</td>
<td>OUTTOO</td>
<td>36</td>
</tr>
<tr>
<td>CONST</td>
<td>28</td>
<td>PATCH</td>
<td>36</td>
</tr>
<tr>
<td>DDERIV</td>
<td>40</td>
<td>PAYOFF</td>
<td>37</td>
</tr>
<tr>
<td>DERIV</td>
<td>29</td>
<td>PDERIV</td>
<td>37</td>
</tr>
<tr>
<td>DPOWER *</td>
<td>38</td>
<td>PMAP *</td>
<td>37</td>
</tr>
<tr>
<td>EPHEM</td>
<td>29</td>
<td>POLEVL</td>
<td>37</td>
</tr>
<tr>
<td>FIXUP</td>
<td>31</td>
<td>POWER *</td>
<td>38</td>
</tr>
<tr>
<td>HAD</td>
<td>31</td>
<td>REDIST</td>
<td>38</td>
</tr>
<tr>
<td>INTRDN</td>
<td>32</td>
<td>RESCAL</td>
<td>39</td>
</tr>
<tr>
<td>INTRUP</td>
<td>32</td>
<td>ROOTR</td>
<td>39</td>
</tr>
<tr>
<td>INVINT</td>
<td>32</td>
<td>SEARCH</td>
<td>40</td>
</tr>
<tr>
<td>KROOT</td>
<td>32</td>
<td>SECANT *</td>
<td>40</td>
</tr>
<tr>
<td>MODIFY</td>
<td>33</td>
<td>SQROOT</td>
<td>41</td>
</tr>
<tr>
<td>MODLEG</td>
<td>34</td>
<td>START</td>
<td>42</td>
</tr>
<tr>
<td>MULT1</td>
<td>34</td>
<td>STARTF *</td>
<td>42</td>
</tr>
<tr>
<td>MULT2</td>
<td>34</td>
<td>TCTPRE</td>
<td>25</td>
</tr>
<tr>
<td>MULT3</td>
<td>34</td>
<td>TSHIFT</td>
<td>43</td>
</tr>
<tr>
<td>MULT4</td>
<td>34</td>
<td>VCAL</td>
<td>43</td>
</tr>
<tr>
<td>MULT7</td>
<td>35</td>
<td>VTMODE</td>
<td>20</td>
</tr>
<tr>
<td>MULT8</td>
<td>35</td>
<td>WYDER</td>
<td>44</td>
</tr>
<tr>
<td>MULT9</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTCAL</td>
<td>35</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* These are function subprograms.
VTMODE: VTMODE is the control subroutine of the variable thrust portion of the program. All the iteration logic is contained in VTMODE as well as the calculations for many of the outputs. The call list is discussed under INPUT. The program can be divided into four sections. The first extends to statement 15, the second from 15 to 99, the third from 99 to 940, and the fourth from 940 on. Each section is flow charted separately.

Section 1: This section has two functions. The first is to initialize control parameters and other quantities used in the succeeding steps. The second is to call subroutines which calculate preliminary data. CONST is called to calculate matrices of constants filling the common blocks CONS1, CONS2. EPHEM is called to determine the state of the launch and arrival planets, and finally START is called to determine a starting trajectory or tracking parameters.

Section 2: This section performs Gauss (NO = 2) or Newton (NO = 3) iterations on the position vector X. ALIGN is called to line up the boundary values of X for each leg, and then the payoff is evaluated in PAYOFF. If the payoff on any iteration is larger than a previous iteration, Newton or Gauss has taken too large a step and SEARCH is called to determine an optimal step size. If on any given iteration the percentage change in payoff is less than the prespecified amount DELTA, convergence is assumed and an exit from the loop is executed. Otherwise,
SUBROUTINE VTMODE (CONT.)

SECTION II

2

ALIGN

PAYOFF

PAYOFF INCREASE

SEARCH

CONVERGENCE

3

MULT 1

MODIFY

NO

YES

20

25

22

YES

FIXUP

SQROOT FAIL

YES

NO

SQROOT

PATCH

PDERIV
SECTION IV

SUBROUTINE VTMODE (CONT.)

FLOWCHART:

4

FAILURE

YES

WRITE

DIAGNOSTIC

NO

940

OUTPUT

RETURN
PDERIV is then called to calculate the first and (pseudo) second order partial derivatives of the payoff. PATCH modifies these partials to enable optimizing of hyperbolic excess velocity angles and to account for patching of each leg of the trajectory.

SQROOT solves the set of linear equations arising from the application of Gauss' or Newton's method. (Note that the solution is loaded in the input array PX). In the event SQROOT encounters a negative square root FIXUP is called to appropriately modify the coefficient matrix (PXX) and SQROOT is called again. MODIFY takes the solution delivered by SQROOT and converts it to actual changes in the trajectory parameters, most of which are loaded into the array DX. The old trajectory (X) is then incremented by DX in MULT1.

Section 3: This section performs two tasks. It shifts the trajectory patch times to achieve a better payoff and smoother acceleration magnitude time history. It converts from two to three dimensions (if NC = 3).

Section 4: This section computes output arrays to be used in MAIN, OUTCAL, and PMAP and AMAP.

TCTPRE: This is the control program for prediction of the thrust intervals which will give a constant thrust trajectory approximating an optimal variable thrust trajectory. A flow chart follows. The call list is (TW, SI) where:

\[ TW \] is the initial thrust to weight ratio of the engine at 1 au.
is the specific impulse of the engine in seconds.

It uses the following subprograms in given order to construct the constant thrust trajectory:

- **PMA** to tabulate power used as a function of time
- **RESCAL** to construct time intervals during which equal amounts of energy are consumed
- **AMAP** to tabulate the optimal variable thrust acceleration at these time points
- **ROOTR** to solve the functional equations which determine the thrust intervals.
- **INVINT** to map back from the constant power consumption intervals to corresponding time points
- **OUTTOO** to print the results in a readable format

Subroutines **PMA** and **AMAP** use results stored in COMMON by subroutine **VTMODE**, hence **TCTPRE** operates on data stored by the most recent call **VTMODE**.

In the case of multiple calls, **TCTPRE** will bypass those calculations which need not be repeated by checking variables set in COMMON.

**ALIGN:**

**ALIGN** serves two functions. It constructs the boundary elements of the vector X from the planet ephemerides \((V_1, V_2)\), specified hyperbolic excess speeds \((H1, H2)\), and calculated excess velocity directions \((VR)\).

It then matches position and velocity at the patch points. This is equivalent to matching the last elements.
of $X$ in one leg to the first elements in the next, weighting the derivatives appropriately to account for the difference in leg times.

**AVTEST:**

The purpose of AVTEST is to examine a converged trajectory for smoothness of acceleration and add legs if necessary. Generally START will assign enough legs for adequate payoff convergence, however there may be jump discontinuities in the acceleration magnitude time histories at patch points. The necessary conditions of the variable thrust optimization problem imply a smooth acceleration time history - but only in an idealized mathematical formulation. For the purpose of calculation the trajectory must be discretized and if the discretization is not fine enough it is quite likely that a lower payoff can be achieved with a discontinuity in acceleration at points where the acceleration is not forced to be continuous.

If the original number of legs is greater than one, AVTEST will add a leg when at some patch point the ratio of a jump to the minimum value on either side is greater than .1. If the original number of legs is one, AVTEST will simply add one leg.

**CONST:**

CONST computes the matrices of constants describing Chebychev interpolation, differentiation, and integration. These matrices compose the common blocks CONS1 and CONS2.

The order of the Chebychev fit (NP) is brought in through the call list. During any given run the operations up to statement 59 are performed only once. However, in the variable power mode the matrices D, E, F are destroyed by the subroutine DERIV so that they need
to be recalculated in the case of constant power. For this purpose a flag NCON is also brought in through the call list.

**DERIV:**

This subroutine calculates the first and second partial derivatives of each leg in accordance with the formulas of the section on mathematics of the variable thrust program.

The part of the routine from statement 89 to 99 performs the matrix multiplications implied by Equation (29) of Reference 1.

**EPHEM:**

EPHEM is the subroutine which determines the heliocentric cartesian components of position and velocity of the planets. Inputs are the Julian date D and planet number N (1 for Mercury to 9 for Pluto). The output is the vector V whose first three components are the x, y, z coordinates of the planet and last three components are \( \hat{x} \), \( \hat{y} \), \( \hat{z} \). The units are au and yr.

The matrix E contains the orbital elements of the planets as follows:

- \( E(1,N) \) = semi major axis (au)
- \( E(2,N) \) = eccentricity
- \( E(3,N) \) = inclination to the ecliptic (radians)
- \( E(4,N) \) = longitude of the ascending node (radians)
- \( E(5,N) \) = longitude of perihelion (radians)
- \( E(6,N) \) = mean longitude of epoch (radians)

As originally delivered the program contains the planetary elements given in Table 1.
# TABLE 1 - PLANETARY ORBITAL ELEMENTS

<table>
<thead>
<tr>
<th></th>
<th>semi-major axis (a.u.)</th>
<th>eccentricity</th>
<th>inclination (degrees)</th>
<th>longitude of node (degrees)</th>
<th>longitude of perihelion (degrees)</th>
<th>mean longitude of epoch (degrees)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. MERCURY</td>
<td>.387099</td>
<td>.205627</td>
<td>7.00399</td>
<td>47.85714</td>
<td>76.83309</td>
<td>222.6217</td>
</tr>
<tr>
<td>2. VENUS</td>
<td>.723332</td>
<td>.006793</td>
<td>3.39423</td>
<td>76.31972</td>
<td>131.0083</td>
<td>174.2943</td>
</tr>
<tr>
<td>3. EARTH</td>
<td>1.000000</td>
<td>.016726</td>
<td>0.</td>
<td>0.</td>
<td>102.2525</td>
<td>100.1581</td>
</tr>
<tr>
<td>4. MARS</td>
<td>1.523691</td>
<td>.093368</td>
<td>1.84991</td>
<td>49.24903</td>
<td>335.3227</td>
<td>258.7673</td>
</tr>
<tr>
<td>5. JUPITER</td>
<td>5.202803</td>
<td>.048435</td>
<td>1.30536</td>
<td>100.0444</td>
<td>13.67823</td>
<td>259.8311</td>
</tr>
<tr>
<td>6. SATURN</td>
<td>9.538843</td>
<td>.055682</td>
<td>2.48991</td>
<td>113.3075</td>
<td>92.26447</td>
<td>280.6713</td>
</tr>
<tr>
<td>7. URANUS</td>
<td>19.18195</td>
<td>.047209</td>
<td>7.73058</td>
<td>73.79630</td>
<td>170.0108</td>
<td>141.3050</td>
</tr>
<tr>
<td>8. NEPTUNE</td>
<td>30.05779</td>
<td>.008575</td>
<td>1.77375</td>
<td>131.3398</td>
<td>44.27395</td>
<td>216.9409</td>
</tr>
<tr>
<td>10.</td>
<td>1.0</td>
<td>0.</td>
<td>45</td>
<td>270</td>
<td>0.</td>
<td>100.1581</td>
</tr>
</tbody>
</table>

Epoch = 2436935 (1 Jan. 1960)

The DO LOOP 1 solves Kepler's equation assuming small eccentricity (less than about .5) and the remainder of the routine converts the conic coordinates to cartesian coordinates.

A provision has been made for a tenth body. One simply replaces the elements in the last continuation card of the DATA statement for E with desired elements. If the eccentricity of the orbit of this body is too high the solution of Kepler's equation will not converge. Therefore the user should associate a number greater than 10 with the body and replace EPHEM with his own subroutines for highly eccentric orbits.

**FIXUP**

FIXUP is called when SQROOT encounters a negative square root, i.e. when Pxx is not positive definite. This can occur only when NO = 3 (i.e. Newton's method is being employed) or NH = 3 (i.e. hyperbolic excess speeds are specified and second order derivatives of VR are being used in the calculation of Pxx).

The function of FIXUP is to reconstruct the Pxx and Px matrices. In the latter case they are reconstructed exactly as delivered by PDERIV. In the former case Pxx is modified by subtracting the term which distinguishes Newton's method from Gauss' method.

Basically reconstruction is possible without going through DERIV again because Pxx is symmetric and the elements above the main diagonal are never destroyed. Thus it is merely necessary to store the elements along the main diagonal.

**HAD**

This subroutine is called from DDERIV and is only used when hyperbolic excess speeds are specified. The function of HAD is to modify the position vector and its second
derivative to account for change in pointing angles of hyperbolic excess velocity during the one-dimensional search. In effect, HAD is a combination of MULT3 and MULT4 in case the input vector has zeros for all but the 2nd and NP-1 elements.

INVINT: This routine interpolates a tabulated function defined on the set \( \{0, \frac{1}{N-1}, \frac{2}{N-1}, \ldots, \frac{N-1}{N-1}\} \) for a 30 element set of arguments. Linear interpolation is used. The call list is \((S, F, N)\)

- \(S\) is an array of length 30. On input it contains the arguments to be interpolated. On termination it contains the corresponding dependent variables.
- \(F\) is an array of length \(N\). It contains the value of the dependent variable tabulated at equal intervals in its argument.
- \(N\) is the size of the array \(F\).

KROOT, INTRDN, INTRUP: These subprograms evaluate the function \(\sigma(t)\) defined implicitly as follows

\[
\sigma(t) = \begin{cases} 
1 & K(t) > 0 \\
0 & K(t) < 0 
\end{cases}
\]

\[
m(t) = 1 - \frac{\sigma_0}{\sigma_0} \int_0^t \sigma(s) \, ds
\]

\[
K(t) = \frac{a_v(t)}{m(t)} - \frac{1}{c} \int_0^t \frac{c_0}{m(s)^2} \frac{a_v(s)}{m(s)^2} \, ds - K
\]

The program assumes all the above functions are defined on the closed interval \([0,1]\) and that the function \(a_v(t_i)\) is tabulated at equal length subintervals.

On each subinterval the program evaluates \(K(t)\) as if
In addition the program evaluates the auxilliary function

\[ J_i = \int_0^t a_v(t) \frac{c\delta \sigma(t)}{m(t)} \, dt \]

The program returns the array TSW which is the tabulation of the switching times starting with the first "switch off".

If \( \sigma(0) = 0 \), TSW(1) = 0

The call list is (AV,NAV,C,BZ,K,JI,TSW) where the first five elements are inputs and the next two elements are output.

- AV is the array \( a_v(t_i) \) \( i = 1, 2, \ldots, \) NAV
- NAV is the number of elements in the array AV.
- C,BZ,K are the constants \( c, \beta_o, K \) appearing in the above equations.
- JI is the function previously defined as \( J_i \)
- TSW is the output array of length 30 tabulating each time that \( \sigma(t) \) changes its value.

Note: If \( \sigma(t) \) switches more than 30 times the subroutine will abort.

MODIFY: MODIFY takes the solution of the SQROOT subroutine (PX) and extracts quantities determining the actual change in the position vector \( (X) \). In the case of fixed hyperbolic
excess speeds the first and/or last few elements of PX are the changes in the pointing angles of the excess velocities (DAL1,DAL2). The remaining elements constitute direct changes in the X vector itself and are loaded into the vector DX.

**MODLEG:** The primary input to MODLEG is a time S between 0 and the trip time (TT). MODLEG then determines to which leg this time corresponds (LN) and what fraction of the total time of the leg has elapsed up to this time. S is then replaced by the latter quantity.

**MULT1:** A trivial subroutine which adds two matrices.

**MULT2:** A trivial subroutine which multiplies two matrices.

**MULT3:** The purpose of this subroutine is to convert back and forth between two different vector representations of the trajectory. Internally the program uses vectors containing the position of the vehicle evaluated at Chebychev points. However the boundary conditions and continuity conditions at patch points require the knowledge of the derivative of position with respect to time at endpoints. The construction of a modified position vector for this purpose is discussed in Reference 1. When MULT3 is called with V as its first argument it is converting the modified vector to an unmodified one. Vice versa when called with U.

**MULT4:** This subroutine performs the matrix multiplication implied by the equation

\[
Y(m) = \sum_{k=1}^{2} U(k) A(k) U(k) X(m) \\
\text{for } m = 1, \ldots, nd
\]
if $L = 1$, or

$$\mathbf{Y}(m) = \sum_{k=1}^{2} \mathbf{U}(k)\mathbf{A}(k)^T \mathbf{U}(k)\mathbf{X}(m)$$

$m = 1, \ldots, nd$

if $L = 2$

(See Reference 1 for interpretation).

**MULT7:** This subroutine performs the matrix multiplication implied by the equation

$$C(k) = \mathbf{A}(k)^T \mathbf{B}(k)$$

$k = 1, 2$

If NS is 2, $C(k)$ is assumed to be symmetric and some savings result.

**MULT8:** This subroutine performs the matrix multiplication implied by the equation

$$\mathbf{B} = \sum_{k=1}^{2} \mathbf{U}(k)^T ((-1)^k), \mathbf{A}(k), \mathbf{A}(k)\mathbf{Y})$$

If NS is 2, $\mathbf{B}$ is assumed to be symmetric and some savings result. See Reference 1 for definitions of $\mathbf{U}$ and $\mathbf{Y}$.

**MULT9:** This subroutine performs the matrix multiplication implied by the equation

$$\mathbf{C} = \sum_{k=1}^{2} \mathbf{U}(k)^T \mathbf{A}^T \mathbf{U}(k) \mathbf{B}$$

If NS is 2, $\mathbf{C}$ is assumed to be symmetric and some savings result.

**OUTCAL:** OUTCAL computes quantities related to the trajectory at 26 equally spaced times and stores them in the matrix $\mathbf{B}$ for delivery to OUTPUT.
OUTCAL has no call list, all inputs are transferred from VTMODE through common blocks. The vectors, WV, XV, AV are approximately the coefficients of the polynomials representing the power level, position vector, and acceleration vector respectively. Thus it is necessary to use the subroutines MODLEG and POLEVL to evaluate these quantities at the desired times.

OUTPT1: This routine prints a 4 line trajectory summary, delimiting each summary with a line of asterisks. Appendix C contains a typical trajectory summary.

OUTPUT: This subroutine prints out a one page summary of the variable thrust optimal trajectory. All the parameters appear in the call list. See Appendix C for a sample output.

OUTTOO: This subroutine displays the results of the thrust-limited predicted performance in the format shown in Appendix C.

PATCH: PATCH has two functions. In the first section it modifies the derivative matrices for the first and last legs in case hyperbolic excess velocity directions are to be optimized. The derivatives of P with respect to X at the endpoints are replaced by the derivatives of P with respect to α and β (See Reference 1).

In the second portion of the subroutine the derivatives are patched together at junction points. The payoff J is of the form

\[ J = \sum_{LN=1}^{NL} 2T3(LN)P_{LN}(X) \]

where \( P_{LN} \) is the payoff corresponding to the LN leg. \( P_{LN} \) depends only on the LN leg of the vector X. Thus the
derivatives of J with respect to any given element of X can be calculated by differentiating just one function $P_{LN}$ -- unless that element happens to be one of the last two elements of a leg. Then the succeeding $P$ also depends on this element. PATCH adds the two components of the derivative together to form the total derivative of J.

**PAYOFF:** The primary purpose of PAYOFF is to call WYDER for each leg of the trajectory. The payoff for each leg as calculated by WYDER is then weighted and accumulated to form the total payoff for the mission. (See section on mathematics of variable thrust program).

To save storage most of the quantities calculated by WYDER are temporarily stored in the matrix of second partials, Pxx.

**PDERIV:** The only purpose of PDERIV is to call DERIV for each leg of the trajectory. Most of the inputs to DERIV have already been calculated by WYDER in PAYOFF and DDERIV.

**PMAP, AMAP:** PMAP and AMAP provide TCTPRE with the variable thrust power level and acceleration magnitude respectively at time H. The vectors necessary to calculate these values are obtained from VTMODE through the common block OUT.

MODLEG is called to refer H to the proper leg and POLEVL is then used to evaluate the power level and acceleration polynomials.

**POLEVL:** Roughly, POLEVL evaluates a polynomial with coefficients Y at a point S. The result is stored in G. The routine is capable of evaluating three polynomials at a time, since the position and acceleration vectors of the trajectory have three cartesian components.
Actually the polynomial must be a weighed sum of Chebychev polynomials, and $Y$ is a certain combination of the weights. To be more precise, this routine evaluates $f(s)$ in equation (22) of Reference 1, with

$$Y = \sum_{k=1}^{2} U(k)^T A(k) U(k) Y_k$$

i.e. $\hat{f}(s) = \sum_{k=1}^{2} T(k,s)^T U(k) Y_k$

**POWER:** The only argument of this subroutine is heliocentric radius $R$ in au. The purpose of the routine is to calculate the power level of the powerplant at radius $R$. This power level is relative to that at one au, i.e., $\text{POWER}(1) = 1$. Note that all quantities in this subroutine are double precision.

The nominal power profile is as follows:

$$\text{POWER}(R) = \begin{cases} 
\frac{2.825}{R^2} - \frac{1.825}{R^{2.5}}, & \text{if } R > .652 \\
1.329, & \text{if } R \leq .652 
\end{cases}$$

**DPOWER:** This subroutine calculates the derivative of the power level with respect to $R$ as computed in POWER above. (This implies that the power profile must be a different table function of $R$. In the nominal case

$$\text{DPOWER}(R) = \begin{cases} 
- \frac{5.65}{R^3} + \frac{4.5625}{R^{3.5}}, & \text{if } R > .652 \\
0, & \text{if } R \leq .652 
\end{cases}$$

**REDIST:** This subroutine takes the new patch times calculated by TSHIFT and redistributes the legs of the trajectory to fit these times. The redistribution is done by
evaluating the old position vector polynomial at the new
Chebychev points of each leg.

The patch time dependent quantities \( T_2, T_3, \) and \( TP \)
are also evaluated anew.

**RESCAL:** This subprogram solves the differential equation:

\[
\frac{dt}{dt} = p(t)
\]
tabulating the solution:

\[
t_i = g(t_i)
\]
Where \( i = 0,1,2...,N, \) \( p(t_i) > 0, \) \( N \leq 301 \)

The call list is CALL RESCAL \((P, G, N, S)\)

- \( P \) is the input array \( P(t_i) \) tabulated at equal
  intervals in \( t \).
- \( G \) is the output array \( g(t_i) \) tabulated at
  equal intervals in \( t \).
- \( N \) is the number of points in the array \( P \) and \( G \).
- \( S \) is \( \int_0^1 pdt \) assuming \( t_0 = 0, \) \( t_N = 1 \)

Simpson's Rule is applied to tabulated \( \tau \) as a function
of \( t \). Then linear interpolation is used to get \( t = g(\tau) \).
at equal intervals.

**ROOTR:** This subprogram uses subroutine KROOT to iterate the
solution of the equations defined in KROOT to satisfy
the constraint \( J_v - J_{\iota} = 0 \) where:

\[
J_v = \int_0^T |a_v|^2 dt \quad \text{defined by VTMODE}
\]
\[
J_{\iota} = \int_0^T |a_v||a_c|dt \quad \text{defined by KROOT}
\]
since $J_i$ is a monotone function of $K$ (one of the arguments of KROOT) an interval halving method is used to satisfy the constraint to single precision accuracy.

Before beginning the iteration the routine tests to see if a solution is feasible (the continuous thrust case) and if so, puts upper and lower limits on $K$.

The call list is:

$$(AV, NAV, C, BZ, TSW, LGF)$$

$AV$ is the array $a_v(t_i)$ $i = 1, 2, \ldots, NAV$

$NAV$ is the number of elements in the array $AV$

$C, BZ$ are constants passed on to KROOT

$TSW$ is the output array of length 30, tabulating each time that the constant thrust switches value.

$LGF$ is the error indicator signifying when a mission is impossible even with continuous thrust.

SEARCH, SECANT, DDERIV:

SEARCH is called if some iteration produces a distinctly higher payoff than the previous one; that is if $P(X) > P(X-DX)$. We know that $P$ must initially decrease in the direction $DX$ since the positive definite characters of $Pxx$ implies that Gauss' method is stable. Therefore for some $RO \in (1, 0), P(X+RODX)$ must have a minimum.

SECANT actually provides the logic for finding that value of $RO$. The first and second guesses by SECANT are $RO = 0$ and $RO = -.5$. All subsequent estimates are obtained using the formula of the secant method (see Mathematics of Variable Thrust Solution). SECANT stops the search when
the percentage change in payoff and absolute value of the directional derivative are below specified tolerances, or else twenty iterations have been taken.

DDERIV calculates the payoff (P) and derivative of the payoff in the direction DX (PRO) for each value of RO delivered by SECANT. (A slight modification is made by the subroutine HAD if pointing angles of hyperbolic excess velocities are involved).

SQROOT: SQROOT solves the equation \( AX = B \) for \( X \), on the assumption that \( A \) is symmetric and positive definite. The technique used is the square root method (see Reference 1).

In our case \( A \) is the matrix of (pseudo) second partial derivatives of \( J \). Because of the division of the trajectory into legs this matrix has the following form:

\[
\begin{pmatrix}
& & & \\
& & & \\
& & & \\
& & & \\
& & & \\
& & & \\
& & & \\
\end{pmatrix}
\]

The coupling occurs only in subspaces corresponding to the elements of \( X \) common to two legs, i.e., position and velocity at patch points. By taking advantage of the structure of this matrix the square root process is essentially reduced to taking the square root of each block. At the beginning of the subroutine IND is set equal to 3. Upon successful completion of the routine IND is set equal to 2. If the routine encounters a negative square root (only possible when \( NO = 3 \) or \( NH = 3 \), and the matrix is not positive definite) a return is executed with IND still 3 — signaling failure.
START: START has two functions. The first is to choose the number of legs (NL) necessary for adequate definition of the trajectory. The choice depends on the travel angle (Q) and percent change of radius (DR) between the launch and arrival planets. Each leg is then assigned an equal portion of the trip time (TT).

The second function is to construct a starting trajectory matching the boundary conditions of the planets. In order to facilitate calculations the starting trajectory is constructed in spherical coordinates and then converted to cartesian coordinates. The starting trajectory itself is basically a spiral (always directed counterclockwise relative to the ecliptic unless one of the planets is retrograde). The precise nature of the spiral is determined by the function subroutine STARTF.

STARTF: STARTF evaluates a function f(s) defined on [0,1]. This function is continuously differentiable and has the following properties: f(0) = 0, f(1) = 1, f'(0) = DZ, f'(1) = DL. In addition f'(s) > 0, if DZ > 0 and DL > 0.

The first four properties insure that the starting trajectory boundary conditions match those of the terminal planets. The last condition insures that the starting trajectory is positively oriented if neither planet is retrograde. The call list is (DZ,DL,S) where:

- DZ is f'(0)
- DL is f'(1)
- S is any number in the interval [0,1]

The procedure is as follows:

if  DZ + DL > 2
\[
\begin{align*}
f(S) &= \left( \frac{D_1 - D_Z}{2} \right) S^2 + (D_Z)S \ B + g(S) + h(S) \\
\text{where} \\
B &= \left( \frac{2}{D_Z + D_1} \right) / \left( 1 + \sqrt{1 - \frac{2}{D_Z + D_1}} \right) \\
g(S) &= \begin{cases} 
0 & S \leq 1-B \\
\frac{D_1}{2} \left( \frac{1-B}{B} \right) (S+B-1)^2 & S > 1-B 
\end{cases} \\
h(S) &= \begin{cases} 
\frac{D_Z}{2} (1-B)S(2-S_B) & S < B \\
\frac{D_Z}{2} (1-B) \ B & S \geq B 
\end{cases}
\end{align*}
\]

and if \( D_Z + D_1 \leq 2 \)

\[
f(S) = S(D_Z + S(3 - 2D_Z - D_1 + S(D_Z + D_1-2)))
\]

**TSHIFT:** This subroutine computes the longitude of the trajectory as a function of time. It then divides the trajectory into NLP arcs of equal longitude and determines the elapsed time for each arc. These times are fed into REDIST, which redistributes the legs of the trajectory accordingly.

**VCAL:** The input to this subprogram is a vector \( VR(m), M = 1,\ldots,ND \) representing the direction of the relative velocity between the vehicle and planet at launch or arrival. \( VCAL \) normalizes the magnitude of this vector to 1 and then computes its first and second order partial derivatives (\( VRA \) and \( VRAA \) respectively) with respect to heliocentric longitude and latitude.
WYDER: WYDER is called for each leg of the trajectory. It has several tasks. First it calculates the payoff \( P \) for the leg. Secondly it constructs the acceleration vector \( \text{AV} \) and the power vector \( \text{WY} \), and thirdly it calculates quantities which are used in DERIV \( (B,Q,G,H) \).

The quantity \( G \) is calculated only if \( \text{NO} = 3 \), i.e., Newton's method is being employed.
V. PROGRAM LIMITATIONS

Trajectories should not be attempted which require more than three revolutions about the sun. As the in-plane transfer angle increases the rate of convergence generally decreases. In cases of very slow convergence the program's automatic convergence criteria may mistakenly decide the optimum has been reached, cease iterating and provide the normal successful printout. If the user should find himself in this situation it would be advisable to either make multiple runs in the vicinity and examine the results for consistency, or change the convergence criterion in VTMODE, recompile, and run again.

Neither the departure or arrival "planet" can have orbits of high eccentricity. If the user exercises the option of specifying the elements of a tenth fictitious planet he must remember that comets are not allowed. This limitation is imposed by subroutine EPHEM. If the user desires to intercept comets or other bodies with highly eccentric orbits it is a simple matter to substitute a suitable routine for the solution of Kepler's equations (see subroutine EPHEM).

The program makes use of a great deal of double precision. The penalty for double precision is slight on the IBM 360 system; however, this may not be so on some other systems. If the program is converted to operate on another computer lacking high speed multiple precision, large increases in run time are to be expected. On the other hand, some computers have a longer, single precision word length (for example the SRU 1108) which makes it possible to eliminate most of the double precision and still achieve comparable accuracy.
VI. REFERENCES


DICTIONARY OF TERMS *

AV: AV is a triply subscripted array (AV(I,M,LN)) and is the value of the Mth coordinate of the acceleration at the Ith Chebychev point of the LNth leg.

BDY: BDY is a triply subscripted array (BDY(M,J,K)) filled with quantities associated with the boundary values of the trajectory. BDY (M, 1,1) M = 1,...,ND are the components of the velocity of the vehicle relative to the launch planet; BDY (M,2,1) M = 1,...,ND are the components of the applied acceleration of the vehicle at the launch planet; and BDY (M,3,1) are the derivatives of that acceleration at the launch planet.

BOUND: A logical variable which is true if and only if hyperbolic excess speeds are specified.

CHEB: A singly subscripted array containing the Chebychev points of [0,1] of order NP.

D: Julian day number, used in subroutine EPHEM

D1: Julian date of departure

D2: Julian date of arrival

D3: Same as D1

D4: Same as D2

DELTA: Convergence tolerance for Gauss' and Newton's iterations

DX: DX is a triply subscripted array (DX(I,M,LN)) containing the change in X from one iteration to next.

* Special groups of terms are also defined in various sections elsewhere in the document (see Input, Output or specific subroutines).
E: A matrix containing the elements of the planets (see description of subroutine EPHEM).

EPOCH: Julian date at epoch (base time). In EPHEM it is 2436935.

H1: Hyperbolic excess speed at launch planet (au/yr)

H2: Hyperbolic excess speed at arrival planet (au/yr)

H3: Same as H1 but in km/sec

H4: Same as H2 but in km/sec

HV1: Initial hyperbolic excess velocity relative to the departure planet (km/sec)

HV2: Final hyperbolic excess velocity relative to the arrival planet (km/sec)

JV: Payoff in au²/yr³ (same as P)

N: NP/2

N1: A parameter which adjusts the indexing in SQROOT to account for differing boundary conditions at launch.

N2: Same as N1, but for arrival.

NA: Acceleration number  
\{ 0 for approximate solution  
\{ 1 for more accurate acceleration time history

NB1: Launch boundary number  
\{ 0 flyby  
\{ 1 specified hyperbolic excess speed  
\{ 2 rendezvous

NB2: Arrival boundary number (Same convention as NB1)

NB3: Same as NB1

NB4: Same as NB2

NC: Number of dimensions (used in call list of VTMODE)
NCON: A flag for the computation of certain matrices in subroutine CONST.

NCOUN: Same as NCOUNT.

NCOUNT: Counter used in TCTPRE to determine when new trajectory has been calculated by VTMODE.

ND: Number of dimensions of the trajectory.

NF: A flag which determines whether TSHIFT has been called earlier.

NFAIL: A flag which is 2 if SQROOT successfully solves its set of linear equations, and 3 if not.

ND2: 2ND

NIT: Maximum number of iterations allowed for achieving an optimum.

NH: An integer taking the values 2 or 3. NH is used only when hyperbolic excess speeds are specified, and represents an option in the calculation of Pxx. Normally actual components of the vector X comprise the set of independent parameters to be optimized. In the case of flyby \( X(2,M,1) \) (and/or \( X(NPM1,M,NL) \)), \( M = 1, \ldots, ND \) belong to this set. However when hyperbolic excess speeds are specified these quantities become partially constrained. In order to regain a free parameter set new unconstrained parameters \( \alpha \) (and \( \beta \) if \( ND=3 \)) are introduced, representing the heliocentric longitude (latitude) of the hyperbolic excess velocity. Now \( X(2,M,1) \) becomes a function of \( \alpha \) (and \( \beta \)) (see Reference 1). This functional relationship is highly non-linear, so that it is sometimes desirable to include terms involving second order derivatives with respect to \( \alpha \) (and \( \beta \)) to Pxx when employing Gauss' method. NH=3 signifies these terms are added, otherwise NH=2; when Newton's method is being employed (i.e. NO=3), NH is also 3.
NL: Number of legs of the trajectory

NLP: Number of legs (same as NL)

NO: Order number  
   \{ 2 for Gauss' method  
   3 for Newton's method  

NP: Number of Chebychev points per leg.

NP1: Departure planet number, varies from 1 for Mercury to 9 for Pluto.

NP2: Arrival planet number

NP3: Same as NP1

NP4: Same as NP2

NPD: (NP)(ND)

NPD1: NP - ND

NPD2: NPD - ND2

NPML: NP - 1

NPO: Same as NPOW

NPW: Power number  
     \{ 1 for variable power  
     0 for constant power  

NR: Number of full revolutions around the sun (NR = 0 for travel angles less than 360°).

NT: Tracking number  
    \{ 0 for no tracking  
    1 for tracking  

P: Payoff in au²/yr³

PLANET: A logical variable which is true if one of the planets is Mercury, Pluto or ficticious.

PS: Payoff in au²/yr³.
PX: PX is a doubly subscripted array (PX(K,LN)), filled in DERIV with first order partial derivatives of the payoff for each leg. PX is also filled in SQROOT with the solution of the linear set of equations defined by Gauss' and Newton's methods.

PXX: PXX is a triply subscripted array (PXX(K1, K2, LN)) filled in DERIV with second order partial derivatives of the payoff for each leg.

SI: Specific impulse of thrusters (sec).

STEP: Fraction of increment of hyperbolic excess speed.

T: T(LN) is a singly subscripted array representing the duration in years of the LNth leg of the trajectory.

T2: T2(LN) = T(LN)^2

T3: T3(LN) = T(LN)^3

TA: Travel angle (radians) of trajectory

TANGLE: A logical variable which is true if and only if the travel angle of the trajectory is greater than 3\pi.

TEST: Relative change in payoff from one iteration to the next.

TP: TP(LN) = (2\pi T(LN))^2

TW: Thrust to weight ratio of spacecraft relative to power level at 1 a.u.

TT: Trip time in years.

V1: Position and velocity of launch planet in a.u. and yr, with V1(1) = x, V1(2) = y, V1(3) = z, V1(4) = \dot{x}, V1(5) = \dot{y}, V1(6) = \dot{z}.

V2: Same as V1 for arrival planet

VR: VR is a doubly subscripted array (VR(M,NB), M = 1, ..., ND-1, NB=1,2) representing the initial (NB=1) and final (NB=2) directions of hyperbolic excess velocity.
VRA: VRA is a triply subscripted array \((VRA(M,I,NB); M=1,\ldots,ND; I=1,\ldots,ND-1; NB=1,2)\) representing the first order partial derivatives of \(VR\) with respect to heliocentric longitude and/or latitude, that is, if

\[
VR(\cdot, NB) = \begin{pmatrix} \cos \alpha \cos \beta \\ \sin \alpha \cos \beta \\ \sin \beta \end{pmatrix}, \quad \text{then}
\]

\[
VRA(M,1) = \frac{\partial VR(M,NB)}{\partial \alpha} \quad \text{and} \quad VRA(M,2) = \frac{\partial VR(M,NB)}{\partial \beta}.
\]

VRAA: A quadruply subscripted array \((VRAA(M,I,J,NB); M=1,\ldots,ND; I=1,\ldots,ND-1; J=1,\ldots,ND-1; NB=1,2)\) representing the second order partial derivatives of \(VR\) with respect to heliocentric longitude and/or latitude.

That is \(VRAA(M,1,1,NB) = \frac{\partial^2 VR(M,NB)}{\partial \alpha^2}\), \(VRAA(M,2,2,NB) = \frac{\partial^2 VR(M,NB)}{\partial \beta^2}\).

\(VRAA(M,1,2,NB) = URAA(M,2,1,NB) = \frac{\partial^2 VR(M,NB)}{\partial \alpha \partial \beta}.
\)

WV: WV is a doubly subscripted array \((WV(I,LN))\) and is equal to \(\sqrt{\frac{\rho}{\rho_0}}\) \((\rho=\text{the power level})\) at the Ith Chebychev point of the LNth leg.

X: X is the same array as XV except that the coordinates of the trajectory at the 2nd and next to last Chebychev points of any leg are replaced by normalized derivatives at the endpoints of the leg. If \(x(M,t)\) is the Mth coordinate of the trajectory, at time \(t\), then

\[
X(2,M,LN) = T(LN) \left. \frac{dx(M,t)}{dt} \right|_{t=t_f}^t \quad \text{and} \quad X(NP-1,M,LN) = -T(LN) \left. \frac{dx(M,t)}{dt} \right|_{t=t_f}^t
\]

where \(T(LN)\) is the duration of the LNth leg and \(t_i\) and \(t_f\) are the initial and final times respectively of the leg.
XS: Same as X, and used to save trajectories.

XV: XV is a triply subscripted array (XV(I,M,LN)) which defines the trajectory. To be more specific, XV is the value of the Mth coordinate of the trajectory at the Ith Chebychev point in the LNth leg.
APPENDIX B

OPERATING INSTRUCTIONS

The program is coded in FORTRAN IV to run on the IBM System 360 computer. It was validated on an IBM 360/75 using OS Release 16 MVT. The deck was compiled using Release 16 FORTRAN G and Release 16 FORTRAN H, Optimization Level 02. It should operate, with only minor changes in control cards, on any IBM system 360 with a FORTRAN compiler and 150,000 bytes available core.

The program is coded as a subprogram so that it may be used as part of a larger mission design master program. Its initial usage, however, is expected to be as a stand alone design tool. In this mode the subprograms are stored on a disk where they may be retrieved and combined with main program each time they are needed. The main program is then a sequence of calls to the subprograms, with the input data passed on as literal constants in the call list. Several examples are given in Appendix C. The input data are discussed under Program Organization - Input.

Computation time varies on a given machine with the type of trajectory being solved. The actual time spent by the computer calculating an optimum trajectory is a function of the number of patched polynomials (see Analysis) and the number of iterations required. Hence the time to compute "easy" versus "hard" trajectories can vary by two orders of magnitude. The first sample case in Appendix C is an Earth to Mars rendezvous. It is an "easy" trajectory. The actual computation time used by the central processing unit of the IBM 360/75 was .3 seconds. Compilation, link editing, and various interrupts for I/O are not included. In other words, on almost any machine the actual computations for this sample case should consume an insignificant time compared to the total run time.
APPENDIX C

SAMPLE CASES

Mars Rendezvous

This is a two-dimensional trajectory having zero hyperbolic excess ve-
locity at Earth and Mars. It takes only one 10 point polynomial for
the 200 day trip. This simple case is included mainly as a time check.
A single trajectory will execute in .3 seconds on the IBM 360/75 using
the Level 02 FORTRAN H compiler. Additional cases of the same or smaller
one leg trajectories execute in .1 seconds. The time is reduced because
a number of computations are performed only for the initial case and
need not be repeated. These execution times are very small compared
with other machine operations, such as compilation, link editing and
various unavoidable delays. Execution times using the FORTRAN G com-
piler increase by more than 50%.

The main program for this trajectory is, starting in card column 7;

    COMMON/BDYP/BDY(3,3,2),PV,PC
    CALL VTMODE(2,0,3,4,2,2,2446538.,2446738.,0.,0.,0,0,0)
    STOP
    END

The short form output appears on the following page.
Jupiter Flyby

This is on 800 day Earth to Jupiter mission using solar power. Departure from earth is with zero excess velocity. The arrival excess velocity at Jupiter is unconstrained. The variable thrust output on the next page is followed by a constant specific impulse output. Note that an accurate acceleration history was requested (NA = 1). If, for this case, NA had been set equal to zero, two extraneous thrust periods would have appeared.

However, the payoff is little changed by the NA setting. In fact the two payoffs are

\[ JC = 12.8831 \quad \text{for } NA = 0 \]
\[ JC = 12.7077 \quad \text{for } NA = 1 \]

The main program is as follows:

```fortran
CALL VTMODE(3,0,3,5,2,0,2444140.,2444940.,0.,0.,1,1,0)
CALL OUTCAL
CALL TCTPRE(.00008,500.)
STOP
END
```

C-3
### VARIABLE THRUST VARIABLE POWER TRAJECTORY

#### 3 DIMENSIONAL TRAJECTORY USING 3 PATCHED POLYNOMIALS WITH 10 CHEBYCHEV POINTS

#### PERFORMANCE INDEX (J) \( 9.592 \)

#### DEPARTURE PLANET EARTH \( \text{ARRIVAL PLANET JUPITER} \)

#### DEPARTURE DAY \( 2444146.0 \) \( \text{ARRIVAL DAY} \) \( 2444946.0 \)

#### EXCESS VELOCITY AT EARTH \( 0.3 \) \( \text{CONSTRAINED} \)

#### EXCESS VELOCITY AT JUPITER \( 7.033 \) \( \text{UNCONSTRAINED} \)

<table>
<thead>
<tr>
<th>TIME (days)</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>R</th>
<th>THETA</th>
<th>PHI</th>
<th>AX</th>
<th>Aw</th>
<th>Az</th>
<th>Mag. A</th>
<th>P/S-P</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>1.003</td>
<td>0.605</td>
<td>-0.337</td>
<td>1.003</td>
<td>-0.328</td>
<td>-0.630</td>
<td>5.56</td>
<td>7.84</td>
<td>-0.12</td>
<td>7.86</td>
<td>0.996</td>
</tr>
<tr>
<td>0.4</td>
<td>0.292</td>
<td>0.096</td>
<td>0.299</td>
<td>1.289</td>
<td>0.896</td>
<td>0.292</td>
<td>3.76</td>
<td>4.23</td>
<td>0.13</td>
<td>7.95</td>
<td>0.911</td>
</tr>
<tr>
<td>0.5</td>
<td>-0.113</td>
<td>1.251</td>
<td>0.003</td>
<td>1.256</td>
<td>0.918</td>
<td>0.622</td>
<td>-2.93</td>
<td>2.38</td>
<td>0.18</td>
<td>3.00</td>
<td>0.758</td>
</tr>
<tr>
<td>0.6</td>
<td>-0.672</td>
<td>1.335</td>
<td>0.003</td>
<td>1.494</td>
<td>1.107</td>
<td>0.410</td>
<td>-1.93</td>
<td>1.02</td>
<td>0.18</td>
<td>2.19</td>
<td>0.597</td>
</tr>
<tr>
<td>0.7</td>
<td>-1.897</td>
<td>1.866</td>
<td>0.003</td>
<td>1.969</td>
<td>1.522</td>
<td>0.260</td>
<td>-1.83</td>
<td>0.37</td>
<td>0.65</td>
<td>1.97</td>
<td>0.467</td>
</tr>
<tr>
<td>0.8</td>
<td>-1.648</td>
<td>1.203</td>
<td>0.003</td>
<td>2.540</td>
<td>143.87</td>
<td>0.29</td>
<td>-6.83</td>
<td>0.36</td>
<td>0.65</td>
<td>1.97</td>
<td>0.467</td>
</tr>
<tr>
<td>0.9</td>
<td>-2.156</td>
<td>1.061</td>
<td>0.003</td>
<td>2.314</td>
<td>152.72</td>
<td>0.38</td>
<td>-6.57</td>
<td>0.26</td>
<td>0.64</td>
<td>1.94</td>
<td>0.404</td>
</tr>
<tr>
<td>1.0</td>
<td>-4.478</td>
<td>2.895</td>
<td>0.003</td>
<td>1.970</td>
<td>128.68</td>
<td>0.40</td>
<td>-6.30</td>
<td>0.23</td>
<td>0.49</td>
<td>0.254</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>-2.740</td>
<td>0.715</td>
<td>0.003</td>
<td>2.832</td>
<td>165.38</td>
<td>0.54</td>
<td>-6.28</td>
<td>0.19</td>
<td>0.43</td>
<td>0.217</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>-3.288</td>
<td>0.526</td>
<td>0.003</td>
<td>3.673</td>
<td>175.15</td>
<td>0.51</td>
<td>-6.23</td>
<td>0.15</td>
<td>0.40</td>
<td>0.189</td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>-3.200</td>
<td>0.539</td>
<td>0.003</td>
<td>3.599</td>
<td>174.22</td>
<td>0.07</td>
<td>-6.19</td>
<td>0.12</td>
<td>0.36</td>
<td>0.167</td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>-3.117</td>
<td>0.138</td>
<td>0.003</td>
<td>3.927</td>
<td>177.76</td>
<td>0.73</td>
<td>-6.18</td>
<td>0.10</td>
<td>0.33</td>
<td>0.149</td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>-3.725</td>
<td>0.058</td>
<td>0.003</td>
<td>3.726</td>
<td>180.89</td>
<td>0.79</td>
<td>-6.12</td>
<td>0.09</td>
<td>0.32</td>
<td>0.135</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td>-3.113</td>
<td>0.052</td>
<td>0.003</td>
<td>3.921</td>
<td>183.98</td>
<td>0.84</td>
<td>-6.09</td>
<td>0.08</td>
<td>0.31</td>
<td>0.124</td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td>-4.681</td>
<td>0.444</td>
<td>0.003</td>
<td>4.166</td>
<td>188.41</td>
<td>0.89</td>
<td>-6.07</td>
<td>0.07</td>
<td>0.31</td>
<td>0.114</td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td>-4.232</td>
<td>0.634</td>
<td>0.003</td>
<td>4.282</td>
<td>188.52</td>
<td>0.94</td>
<td>-6.06</td>
<td>0.06</td>
<td>0.30</td>
<td>0.106</td>
<td></td>
</tr>
<tr>
<td>1.9</td>
<td>-4.307</td>
<td>0.061</td>
<td>0.003</td>
<td>4.494</td>
<td>193.63</td>
<td>0.99</td>
<td>-6.05</td>
<td>0.06</td>
<td>0.30</td>
<td>0.099</td>
<td></td>
</tr>
<tr>
<td>2.0</td>
<td>-4.487</td>
<td>-1.005</td>
<td>0.003</td>
<td>4.599</td>
<td>192.62</td>
<td>1.03</td>
<td>-6.05</td>
<td>0.05</td>
<td>0.29</td>
<td>0.093</td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>-4.594</td>
<td>-1.105</td>
<td>0.003</td>
<td>4.745</td>
<td>194.47</td>
<td>1.07</td>
<td>-6.04</td>
<td>0.05</td>
<td>0.29</td>
<td>0.088</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>-4.688</td>
<td>-1.592</td>
<td>0.003</td>
<td>4.683</td>
<td>196.23</td>
<td>1.11</td>
<td>-6.03</td>
<td>0.04</td>
<td>0.28</td>
<td>0.084</td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>-4.777</td>
<td>-1.535</td>
<td>0.003</td>
<td>5.112</td>
<td>197.84</td>
<td>1.14</td>
<td>-6.02</td>
<td>0.03</td>
<td>0.27</td>
<td>0.080</td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>-4.841</td>
<td>-1.703</td>
<td>0.003</td>
<td>5.133</td>
<td>199.40</td>
<td>1.16</td>
<td>-6.01</td>
<td>0.02</td>
<td>0.26</td>
<td>0.077</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>-4.901</td>
<td>-1.672</td>
<td>0.011</td>
<td>5.267</td>
<td>200.88</td>
<td>1.17</td>
<td>-6.01</td>
<td>0.01</td>
<td>0.25</td>
<td>0.074</td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td>-4.951</td>
<td>-2.331</td>
<td>0.016</td>
<td>5.352</td>
<td>222.31</td>
<td>1.24</td>
<td>-6.01</td>
<td>0.01</td>
<td>0.25</td>
<td>0.071</td>
<td></td>
</tr>
<tr>
<td>2.7</td>
<td>-4.991</td>
<td>-2.186</td>
<td>0.121</td>
<td>5.451</td>
<td>233.66</td>
<td>1.27</td>
<td>-6.01</td>
<td>0.01</td>
<td>0.20</td>
<td>0.069</td>
<td></td>
</tr>
</tbody>
</table>
Saturn Flyby

A series of five Earth to Saturn flybys are computed varying hyperbolic excess velocity at earth. The main program is:

```
COMMON/BDYP/BDY(3,3,2),PV,PC
CALL VTMODE(3,0,3,6,2,0,2446750.,2448550.,0.,0.,0,0,0)
DO 1 I = 1,4
HV1 = 1.0 + (I-1)*1.0
CALL VTMODE(3,0,3,6,1,0,2446750.,2448550.,HV1,0.,0,0,1)
1 CONTINUE
STOP
END
```

This illustrates tracking in hyperbolic excess speed from a previously computed rendezvous solution.
| VARIABLE THRUST CONSTANT POWER TRAJECTORY IN 2 DIMENSIONS USING 3 LEGS OF 10 CHEBYSHEV POINTS |
|---|---|---|---|---|---|---|---|---|---|
| DEPARTURE PLANET IS EARTH | AT JULIAN DAY 2444750.0 | ARRIVAL PLANET IS SATURN | AT JULIAN DAY 2448550.0 |
| EXCESS VELOCITY AT EARTH IS | 0.0 | CONSTRAINED | AT SATURN IS | 0.869490E-01 | UNCONSTRAINED |
| PERFORMANCE INDEX IS | 0.592061E-01 | CONSTANT POWER |
| | | | | | | | | | |
| VARIABLE THRUST CONSTANT POWER TRAJECTORY IN 3 DIMENSIONS USING 3 LEGS OF 10 CHEBYSHEV POINTS |
|---|---|---|---|---|---|---|---|---|---|
| DEPARTURE PLANET IS EARTH | AT JULIAN DAY 2444750.0 | ARRIVAL PLANET IS SATURN | AT JULIAN DAY 2448550.0 |
| EXCESS VELOCITY AT EARTH IS | 0.593390E-01 | CONSTRAINED | AT SATURN IS | 0.812348E-01 | UNCONSTRAINED |
| PERFORMANCE INDEX IS | 0.470252E-01 | CONSTANT POWER |
| | | | | | | | | | |
| VARIABLE THRUST CONSTANT POWER TRAJECTORY IN 3 DIMENSIONS USING 3 LEGS OF 10 CHEBYSHEV POINTS |
|---|---|---|---|---|---|---|---|---|---|
| DEPARTURE PLANET IS EARTH | AT JULIAN DAY 2444750.0 | ARRIVAL PLANET IS SATURN | AT JULIAN DAY 2448550.0 |
| EXCESS VELOCITY AT EARTH IS | 0.200000E-01 | CONSTRAINED | AT SATURN IS | 0.778638E-01 | UNCONSTRAINED |
| PERFORMANCE INDEX IS | 0.371300E-01 | CONSTANT POWER |
| | | | | | | | | | |
| VARIABLE THRUST CONSTANT POWER TRAJECTORY IN 3 DIMENSIONS USING 3 LEGS OF 10 CHEBYSHEV POINTS |
|---|---|---|---|---|---|---|---|---|---|
| DEPARTURE PLANET IS EARTH | AT JULIAN DAY 2444750.0 | ARRIVAL PLANET IS SATURN | AT JULIAN DAY 2448550.0 |
| EXCESS VELOCITY AT EARTH IS | 0.300000E-01 | CONSTRAINED | AT SATURN IS | 0.768120E-01 | UNCONSTRAINED |
| PERFORMANCE INDEX IS | 0.294414E-01 | CONSTANT POWER |
| | | | | | | | | | |
| VARIABLE THRUST CONSTANT POWER TRAJECTORY IN 3 DIMENSIONS USING 3 LEGS OF 10 CHEBYSHEV POINTS |
|---|---|---|---|---|---|---|---|---|---|
| DEPARTURE PLANET IS EARTH | AT JULIAN DAY 2444750.0 | ARRIVAL PLANET IS SATURN | AT JULIAN DAY 2448550.0 |
| EXCESS VELOCITY AT EARTH IS | 0.400000E-01 | CONSTRAINED | AT SATURN IS | 0.777259E-01 | UNCONSTRAINED |
| PERFORMANCE INDEX IS | 0.215882E-01 | CONSTANT POWER |
| | | | | | | | | | |

**NOT REPRODUCIBLE**
Out-of-the-Ecliptic Probe

This is an example of using the three-dimensional start option. The goal is to achieve a 1 au circular solar orbit inclined 45° to the ecliptic. A tenth set of orbital elements were inserted in Subroutine EPHEM. They are

\[
\begin{align*}
  a &= 1.000000 \\
  e &= 0. \\
  i &= 0.8 \text{ radians} \\
  \Omega &= 4.7 \text{ radians} \\
  \varpi &= 0 \text{ radians} \\
  L &= 1.748089
\end{align*}
\]

The trajectory assumes solar power. The launch date is near-optimum for the 720 day trip time. The main program is

```plaintext
COMMON/BDY/BDY(3,3,2),PV,PC
CALL VTMODE(1,1,3,10,2,2,2443958.92444678.,0.,O.,19,0,0)
CALL OUTCAL
STOP
END
```
### Variable Thrust Variable Power Trajectory

#### 3 Dimensional Trajectory Using 4 Patched Polynomials with 10 Chebyshev Points

**Performance Index (TV)** 17,840

<table>
<thead>
<tr>
<th>Departure Planet</th>
<th>Arrival Planet Tenth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Earth</td>
<td>2442.98</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Excess Velocity at Earth</th>
<th>0.00</th>
<th>Constrained</th>
</tr>
</thead>
<tbody>
<tr>
<td>Excess Velocity at Tenth</td>
<td>0.00</td>
<td>Constrained</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>R</th>
<th>Theta</th>
<th>Phi</th>
<th>Ax</th>
<th>Ay</th>
<th>Az</th>
<th>Mag</th>
<th>A</th>
<th>P/Po</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>-0.095</td>
<td>-0.071</td>
<td>0.000</td>
<td>0.997</td>
<td>-175.90</td>
<td>0.00</td>
<td>-0.27</td>
<td>-0.15</td>
<td>0.58</td>
<td>0.66</td>
<td>1.003</td>
<td></td>
</tr>
<tr>
<td>28.50</td>
<td>-0.450</td>
<td>-0.139</td>
<td>0.000</td>
<td>1.000</td>
<td>-147.61</td>
<td>0.24</td>
<td>-0.22</td>
<td>0.02</td>
<td>2.81</td>
<td>2.82</td>
<td>0.993</td>
<td></td>
</tr>
<tr>
<td>57.00</td>
<td>-0.500</td>
<td>-0.070</td>
<td>0.000</td>
<td>1.005</td>
<td>-118.86</td>
<td>1.36</td>
<td>-0.20</td>
<td>0.15</td>
<td>4.35</td>
<td>4.35</td>
<td>0.994</td>
<td></td>
</tr>
<tr>
<td>85.50</td>
<td>-0.045</td>
<td>-1.017</td>
<td>0.005</td>
<td>1.021</td>
<td>-52.53</td>
<td>3.63</td>
<td>-0.15</td>
<td>0.26</td>
<td>4.88</td>
<td>4.89</td>
<td>0.987</td>
<td></td>
</tr>
<tr>
<td>115.00</td>
<td>0.425</td>
<td>0.5023</td>
<td>0.120</td>
<td>1.023</td>
<td>-65.70</td>
<td>6.71</td>
<td>-0.05</td>
<td>0.33</td>
<td>4.32</td>
<td>4.34</td>
<td>0.976</td>
<td></td>
</tr>
<tr>
<td>144.00</td>
<td>0.738</td>
<td>0.6111</td>
<td>0.174</td>
<td>1.025</td>
<td>-37.75</td>
<td>9.75</td>
<td>-0.08</td>
<td>0.30</td>
<td>2.81</td>
<td>2.83</td>
<td>0.974</td>
<td></td>
</tr>
<tr>
<td>173.00</td>
<td>0.854</td>
<td>-0.176</td>
<td>0.060</td>
<td>1.030</td>
<td>-4.90</td>
<td>11.51</td>
<td>0.37</td>
<td>0.09</td>
<td>0.72</td>
<td>0.82</td>
<td>0.968</td>
<td></td>
</tr>
<tr>
<td>202.00</td>
<td>0.679</td>
<td>0.0421</td>
<td>1.041</td>
<td>10.40</td>
<td>0.82</td>
<td>-0.17</td>
<td>-1.43</td>
<td>1.66</td>
<td>0.956</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>231.00</td>
<td>0.750</td>
<td>0.7277</td>
<td>0.137</td>
<td>1.056</td>
<td>4.41</td>
<td>7.49</td>
<td>1.32</td>
<td>-0.33</td>
<td>3.19</td>
<td>3.46</td>
<td>0.963</td>
<td></td>
</tr>
<tr>
<td>260.00</td>
<td>0.380</td>
<td>0.593</td>
<td>0.030</td>
<td>1.063</td>
<td>69.08</td>
<td>1.64</td>
<td>0.37</td>
<td>-4.27</td>
<td>4.59</td>
<td>0.933</td>
<td></td>
<td></td>
</tr>
<tr>
<td>289.00</td>
<td>0.028</td>
<td>0.057</td>
<td>-0.10</td>
<td>1.064</td>
<td>91.17</td>
<td>-5.25</td>
<td>1.69</td>
<td>-0.38</td>
<td>-4.56</td>
<td>4.88</td>
<td>0.933</td>
<td></td>
</tr>
<tr>
<td>318.00</td>
<td>0.433</td>
<td>0.309</td>
<td>-0.26</td>
<td>1.053</td>
<td>117.59</td>
<td>-13.29</td>
<td>-1.69</td>
<td>-0.36</td>
<td>3.99</td>
<td>4.27</td>
<td>0.964</td>
<td></td>
</tr>
<tr>
<td>347.00</td>
<td>0.781</td>
<td>0.565</td>
<td>-0.203</td>
<td>1.032</td>
<td>144.11</td>
<td>-20.92</td>
<td>1.00</td>
<td>-0.25</td>
<td>-2.60</td>
<td>2.82</td>
<td>0.993</td>
<td></td>
</tr>
<tr>
<td>376.00</td>
<td>0.910</td>
<td>0.100</td>
<td>-0.417</td>
<td>1.007</td>
<td>173.71</td>
<td>-24.57</td>
<td>0.32</td>
<td>0.11</td>
<td>-0.60</td>
<td>0.69</td>
<td>0.993</td>
<td></td>
</tr>
<tr>
<td>405.00</td>
<td>0.327</td>
<td>0.340</td>
<td>-0.371</td>
<td>0.983</td>
<td>205.13</td>
<td>-22.34</td>
<td>-0.82</td>
<td>0.57</td>
<td>1.15</td>
<td>1.83</td>
<td>1.018</td>
<td></td>
</tr>
<tr>
<td>434.00</td>
<td>0.534</td>
<td>0.770</td>
<td>-0.226</td>
<td>0.946</td>
<td>235.28</td>
<td>-13.53</td>
<td>-2.07</td>
<td>0.76</td>
<td>3.24</td>
<td>3.92</td>
<td>1.040</td>
<td></td>
</tr>
<tr>
<td>463.00</td>
<td>0.113</td>
<td>0.943</td>
<td>0.002</td>
<td>0.950</td>
<td>263.07</td>
<td>0.10</td>
<td>-2.75</td>
<td>0.58</td>
<td>4.07</td>
<td>4.95</td>
<td>1.055</td>
<td></td>
</tr>
<tr>
<td>492.00</td>
<td>0.320</td>
<td>0.850</td>
<td>0.252</td>
<td>2.005</td>
<td>15.53</td>
<td>-2.66</td>
<td>0.30</td>
<td>3.77</td>
<td>4.51</td>
<td>1.064</td>
<td></td>
<td></td>
</tr>
<tr>
<td>521.00</td>
<td>0.649</td>
<td>0.511</td>
<td>0.493</td>
<td>0.941</td>
<td>321.70</td>
<td>28.75</td>
<td>-1.37</td>
<td>0.14</td>
<td>2.30</td>
<td>2.68</td>
<td>1.065</td>
<td></td>
</tr>
<tr>
<td>550.70</td>
<td>0.782</td>
<td>-0.025</td>
<td>0.537</td>
<td>0.946</td>
<td>356.12</td>
<td>34.47</td>
<td>0.12</td>
<td>-0.03</td>
<td>0.14</td>
<td>0.18</td>
<td>1.056</td>
<td></td>
</tr>
<tr>
<td>579.00</td>
<td>0.439</td>
<td>0.638</td>
<td>0.472</td>
<td>0.966</td>
<td>391.41</td>
<td>29.32</td>
<td>1.61</td>
<td>-0.20</td>
<td>-1.45</td>
<td>2.56</td>
<td>1.039</td>
<td></td>
</tr>
<tr>
<td>608.00</td>
<td>0.472</td>
<td>0.836</td>
<td>0.269</td>
<td>0.582</td>
<td>422.39</td>
<td>15.92</td>
<td>2.66</td>
<td>-0.29</td>
<td>-3.41</td>
<td>4.33</td>
<td>1.020</td>
<td></td>
</tr>
<tr>
<td>637.00</td>
<td>0.030</td>
<td>0.991</td>
<td>-0.023</td>
<td>0.965</td>
<td>445.36</td>
<td>-1.32</td>
<td>3.01</td>
<td>-0.37</td>
<td>-3.97</td>
<td>5.00</td>
<td>1.006</td>
<td></td>
</tr>
<tr>
<td>666.00</td>
<td>-0.277</td>
<td>0.901</td>
<td>0.336</td>
<td>1.000</td>
<td>467.20</td>
<td>-12.48</td>
<td>2.63</td>
<td>-0.41</td>
<td>-3.59</td>
<td>4.87</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>695.00</td>
<td>-0.556</td>
<td>0.591</td>
<td>0.596</td>
<td>1.000</td>
<td>528.51</td>
<td>-15.92</td>
<td>1.64</td>
<td>-0.38</td>
<td>-2.38</td>
<td>2.92</td>
<td>0.999</td>
<td></td>
</tr>
</tbody>
</table>

---

Note: The table represents the trajectory data with specific parameters for each time step, including position, velocity, and other relevant factors for a 3-dimensional trajectory using patched polynomials with 10 Chebyshev points.
# APPENDIX D

## LISTINGS

### INDEX OF SUBROUTINES FOR APPENDIX D

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Page</th>
<th>Subroutine</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIGN</td>
<td>D-13</td>
<td>OUTPT1</td>
<td>D-17</td>
</tr>
<tr>
<td>AMAP*</td>
<td>D-31</td>
<td>OUTPUT</td>
<td>D-16</td>
</tr>
<tr>
<td>AVTEST</td>
<td>D-19</td>
<td>OUTTOO</td>
<td>D-28</td>
</tr>
<tr>
<td>CONST</td>
<td>D-22</td>
<td>PATCH</td>
<td>D-11</td>
</tr>
<tr>
<td>DDERIV</td>
<td>D-8</td>
<td>PAYOFF</td>
<td>D-7</td>
</tr>
<tr>
<td>DERIV</td>
<td>D-10</td>
<td>PDERIV</td>
<td>D-9</td>
</tr>
<tr>
<td>DPOWER*</td>
<td>D-25</td>
<td>PMAPI</td>
<td>D-31</td>
</tr>
<tr>
<td>EPHEM</td>
<td>D-24</td>
<td>POLEVL</td>
<td>D-19</td>
</tr>
<tr>
<td>FIXUP</td>
<td>D-13</td>
<td>POWER*</td>
<td>D-25</td>
</tr>
<tr>
<td>HAD</td>
<td>D-18</td>
<td>REDIST</td>
<td>D-15</td>
</tr>
<tr>
<td>INTRDN</td>
<td>D-30</td>
<td>RESCAL</td>
<td>D-27</td>
</tr>
<tr>
<td>INTRUP</td>
<td>D-31</td>
<td>ROOTR</td>
<td>D-27</td>
</tr>
<tr>
<td>INVINT</td>
<td>D-26</td>
<td>SEARCH</td>
<td>D-7</td>
</tr>
<tr>
<td>KROOT</td>
<td>D-29</td>
<td>SECANT*</td>
<td>D-8</td>
</tr>
<tr>
<td>MODIFY</td>
<td>D-12</td>
<td>SQROOT</td>
<td>D-23</td>
</tr>
<tr>
<td>MODLEG</td>
<td>D-19</td>
<td>START</td>
<td>D-5</td>
</tr>
<tr>
<td>MULT1</td>
<td>D-20</td>
<td>STARTF*</td>
<td>D-4</td>
</tr>
<tr>
<td>MULT2</td>
<td>D-20</td>
<td>TCTPRE</td>
<td>D-25</td>
</tr>
<tr>
<td>MULT3</td>
<td>D-20</td>
<td>TSHIFT</td>
<td>D-14</td>
</tr>
<tr>
<td>MULT4</td>
<td>D-20</td>
<td>VCAL</td>
<td>D-18</td>
</tr>
<tr>
<td>MULT7</td>
<td>D-21</td>
<td>VMODE</td>
<td>D-2</td>
</tr>
<tr>
<td>MULT8</td>
<td>D-21</td>
<td>WYDER</td>
<td>D-6</td>
</tr>
<tr>
<td>MULT9</td>
<td>D-22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTCAL</td>
<td>D-16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*These are function subprograms.*
COMMON/BDYP/BDY(3,3,2),PV,PC
CALL VTMODE(2,0,3,4,2,2,2446538.,2446738.,0.,0.,0.,0.,0.,0)
STOP
END

SUBROUTINE VTMODE(NC,NR,NP3,NP4,NB3,NB4,D3,D4,H3,H4,NPO,NA,NT)
COMMON/BDYP/BDY(3,3,2),PV,PC
DOUBLE PRECISION X(10,3,6),PIX,PIX,G,VR,DVR,VRA,VRAA,DAL,TTP,DX
DOUBLE PRECISION A,AA,BA,U,V,CHEB,TT,T,T2,T3,TP,P12,XV,AV,WV,P
COMMON/SCRACH/DX(30,6)
COMMON/PXXPX/PXX(30,30,6),PX(30,6),G(90,6)
COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NO,NH
COMMON/OUT/XV(30,6),AV(10,3,6),WV(10,6)
COMMON/CONS1/A5,5,2,AA5,5,2,BA5,5,2,U10,2,V10,2,CHEB10
COMMON/TIMEQ/TT,T6,T26,T36,TP6,P12,TTP
COMMON/COUNT/NCOUNT,NCOUNT,NCOUNT,NLP
COMMON/VREL/VR(3,2),VRA(3,2,2),VRAA(3,2,2),DVR(3,2),DAL(3,2,2)
COMMON/PUT/NP1,NP2,D1,D2,HI,H2,PS,V16,VI6,V26
NF=0
IF(NT.EQ.1.AND.NP1.EQ.NP3.AND.NP2.EQ.NP4.AND.D1.EQ.D3.AND.
C02.EQ.D4) NF=1
NCOUNT=NCOUNT+1
P12=6.28318530717958647692D0
D1=D3
D2=D4
TTP=TT
TT=(D2-D1)/365.25
NP1=NP3
NP2=NP4
NB1=NB3
NB2=NB4
CALL EPHEM(D1,NP1,V1)
CALL EPHEM(D2,NP2,V2)
H1=.2104*H3*FLOAT(2-NB3)
H2=.2104*H4*FLOAT(2-NB4)
P=0.
NP=10
NPM1=NP-1
N=NP/2
5 IF((NCON.EQ.0).OR.(NPO.EQ.0.AND.NPOW.NE.0)) CALL CONST(NP,N,NCON)
NPOw=NPO
ND=NC
IF(NC.EQ.3.AND.NT.EQ.0) ND=2
IF(NC.EQ.1) ND=3
NIT=30
DELTA=.01
CALL START(X,XV,NR,NLP,TA,NT)
IF(NL.EQ.1.AND.NLP.EQ.1) NF=1
IF(ABS(TA).GT.9.424777) DELTA=.001
IF(ABS(TA).GT.9.424777) NIT=100
10 NPDP=NP*ND
PD=2*ND
NPD1=NPD-ND
NPD2=NPD-ND2
15 DO 99 IT=1,NIT
   NO=2
   IF(NF.EQ.1.OR.NT.EQ.1) NO=3
   CALL ALIGN(X)
   CALL PAYOFF(X,P)
20 IF(IT.GT.1.AND.P.GT.(1.+DELTAF)*PV) CALL SEARCH(X,P,DX)
   TEST=DABS((PV-P)/P)
   PV=P
22 IF(TEST.LT.DELTA.AND.IT.GT.1.AND.NB1.EQ.NB3.AND.NB2.EQ.NB4)
   CGO TO 100
   IF(IT.EQ.NIT) GO TO 940
   NB1=NB3
   NB2=NB4
   IF(IT.EQ.1.AND.NT.EQ.1.AND.NB3.EQ.1) NB1=2
   IF(IT.EQ.1.AND.NT.EQ.1.AND.NB4.EQ.1) NB2=2
   CALL PDERIV
   NH=3
   IF(NB1.NE.1.AND.NB2.NE.1) NH=2
   N1=1+((ND-2)*NB1**2+(4-ND)*NB1+2*ND)/2
   N2=NPD-((ND-2)*NB2**2+(4-ND)*NB2+2*ND)/2
25 CALL PATCH(T,H1,H2)
   CALL SQROOT(PXX,PX,PX,NPD,NL,ND2,N1,N2,NFAIL)
   TEST1=(VAR(3,2,1)*PX(ND2,1)**2+(FLOAT(2-ND)*PX(ND2-1,1))**2
   TEST2=(VAR(3,2,2)*PX(NPD2+1,NL)**2+(FLOAT(2-ND)*PX(NPD2+2,NL))**2
   IF(NB1.EQ.1.AND.NH.EQ.3.AND.TEST1.GT.2.) NFAIL=3
   IF(NB2.EQ.1.AND.NH.EQ.3.AND.TEST2.GT.2.) NFAIL=3
   IF(NFAIL.EQ.2) GO TO 30
   IF(NO.EQ.2.AND.NH.EQ.2) GO TO 940
   CALL FIXUP
   IF(NC.EQ.2) NH=2
   IF(NO.EQ.3) NO=2
   GO TO 25
30 CALL MODIFY(PX,DX,H1,H2)
   IF(NB1.EQ.1) CALL MULT1(VR(1,1),DVR(1,1),VR(1,1),1.DO,1,ND)
   IF(NB2.EQ.1) CALL MULT1(VR(1,2),DVR(1,2),VR(1,2),1.DO,1,ND)
   DO 39 LN=1,NL
39 CALL MULT1(X(1,1,1),DX(1,1,1),X(1,1,1),1.DO,ND,NP)
99 CONTINUE
100 CONTINUE
200 IF((NF.EQ.1).OR.(NT.EQ.1.AND.NL.EQ.NLP)) GO TO 750
   NF=1
   CALL TSHIFT(X,XV,NL,NLP,NP,N)
   GO TO 15
750 IF(NA.EQ.0) GO TO 925
   CALL AVTEST(AV,T2,NLP,NL,ND,NP)
   GO TO 775
775 IF(NLP.EQ.NL) GO TO 800
   CALL TSHIFT(X,XV,NL,NLP,NP,N)
   GO TO 15
800 GO TO 925
925 IF(ND.EQ.NC.OR.ND.EQ.3) GO TO 950
   ND=NC
   DC 930 LN=1,NL
   DC 930 I=1,NP

930  \( x(i,3,ln)=0. \)
940 CONTINUE
    NCOUNT=0
    WRITE(6,1009)
950 CONTINUE
    H1=0.*
    H2=0.*
    DO 975 M=1,ND
      IF(NB3*EQ.2) VR(M,1)=AV(I,M,1)
      IF(NB4*EQ.2) VR(M,2)=AV(NP,M,NL)
      BDY(M,1,1)=X(2,M,1)/T(1)-V1(M+3)
      BDY(M,1,2)=-X(NPM1,M,NL)/T(NL)-V2(M+3)
      H1=H1+BDY(M,1,1)**2
      H2=H2+BDY(M,1,2)**2
      BDY(M,2,1)=AV(I,M,1)*T2(1)
      BDY(M,2,2)=AV(NP,M,NL)*T2(NL)
      Z1=AV(2,M,1)
      Z2=AV(NPM1,M,NL)
      DO 973 I=1,NP
         Z1=Z1+U(I,1)*AV(I,M,1)
      973 Z2=Z2+U(I,2)*AV(I,M,NL)
      BDY(M,3,1)=Z1*T3(1)
      975 BDY(M,3,2)=-Z2*T3(NL)
      H1=SQRT(H1)/.2104
      H2=SQRT(H2)/.2104
      PS=.7121*PV
      DO 980 LN=1,NL
         CALL MULT4(A,XV(1,LN),XV(1,LN),ND,NP,N,1)
         CALL MULT4(A,AV(1,1,LN),AV(1,1,LN),ND,NP,N,1)
         CALL MULT4(A,AVW(1,1,LN),AVW(1,1,LN),1,NP,N,1)
      980 DO 980 M=1,ND
      DO 980 I=1,NP
      980 AV(I,M,LN)=AV(I,M,LN)*T2(LN)
      CALL OUTPT1(NP3,NP4,D3,D4,H1,H2,PS,NL,ND,NP,NB3,NB4,NPOW)
      RETURN
1009 FORMAT(/4X39HFOLLOWING TRAJECTORY FAILED TO CONVERGE) END

BLOCK DATA
DOUBLE PRECISION PXG
DOUBLE PRECISION VR,DVR,VRA,VRAA,DAL
COMON/COUNT/NCOUNT,NCOUN,NLP
COMON/PXXPX/PXX(30,30,6),PX(30,6),G(90,6)
COMON/VREL/VR(3,2),VKA(3,2,2),VRAA(3,2,?,2),DVR(3,2),DAL(3,2,2)
DATA PXG /6120*0.00/
DATA NCOUNT,NCOUN,NLP/3*0/
DATA VR,VRA,VRAA,DVR,DAL /60*0.00/
END

REAL FUNCTION STARTF(D2,D1,S)
IF(DZ+D1.GT.2.) GO TO 1
STARTF= S*(DZ+S*(3.-2.*DZ-D1+S*(DZ+D1-2.*)))
RETURN
1 B=(2./(DZ+D1))/(1.+SQR(1.-2./(DZ+D1))))
STARTF= B*S*(DZ+S*(D1-DZ)**5)
IF(S.GT.1.-B) STARTF=STARTF+.5*(S+B-1.)**2*D1*(1.-B)/B
IF(S.LT.B) GO TO 2
STARTF=STARTF+.5*B*(1.-B)*DZ
RETURN
2 STARTF=STARTF+.5*S*DZ*(1.-B)*(2.-S/B)
RETURN
END

SUBROUTINE START(X,XV,NR,NLP,TA,NT)
DOUBLE PRECISION X(10,3,6),XV(10,3,6),VR,DVR,VRA,VRAA,DAL
DOUBLE PRECISION A,AA,BA,UY,CHEB,T1,T2,T3,TP,PI2,TPW,W
COMMON/PUT/NP1,NP2,D1,D2,H1,H2,PS,V1(6),V2(6)
COMMON/VREL/VR(3,2),VRA(3,2),VRAA(3,2,2),DVR(3,2),DAL(3,2,2)
COMMON/TIMEQ/T1,T2(6),T3(6),TP(6),PI2,TPW
COMMON/PARAM/NL,NO,NP,N,NPD,NPM1,N2D2,NPD1,NPOW,NB1,NB2,NO,NH
COMMON/CONS/AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
Q1=ATAN2(V1(1),V1(1))
Q2=ATAN2(V2(1),V2(1))
Q=Q2-Q1+6.283185
Q=AMOD(Q,6.283185)
Q=Q+6.283185*FLOAT(NR)
IF(NT.NQ.1) Q=TA+ATAN2(SIN(Q-TA),COS(Q-TA))
TA=Q
R01=V1(1)**2+V1(2)**2
R02=V2(1)**2+V2(2)**2
R1=SQRT(R01+V1(3)**2)
R2=SQRT(R02+V2(3)**2)
R=R2-R1
Q1=ATAN(V1(3)/SQRT(R01))
Q2=ATAN(V2(3)/SQRT(R02))
Q=Q2-Q1
DR=AMAX1(R1,R2)/AMIN1(R1,R2)
NLP=1.+Q*.31830989
IF(DR.GT.2.) NLP=1.+Q*.4244132
IF(DR.GT.6.) NLP=1.+Q*.6366198
NLP=MNO(NLP,6)
IF(NT.NQ.0) NL=NLP
DO 5 LN=1,NL
IF(NT.NQ.0) T(LN)=T/FLOAT(NL)
IF(NT.NQ.1) T(LN)=T(LN)**2
T2(LN)=1./T(LN)**2
T3(LN)=T2(LN)/T(LN)
TP(LN)=(PI2*T(LN))**2
IF(NT.NQ.1) RETURN

R1T=(V1(1)*V1(4)+V1(2)*V1(5)+V1(3)*V1(6))/R1
R2T=(V2(1)*V2(4)+V2(2)*V2(5)+V2(3)*V2(6))/R2
Q1T=(R1*V1(6)-R1T*V1(3))/(R1*R01)
Q2T=(R2*V2(6)-R2T*V2(3))/(R2*R02)
Q1T=(V1(1)*V1(5)-V1(2)*V1(4))/RO1
Q2T=(V2(1)*V2(5)-V2(2)*V2(4))/RO2
IF(R*EQ.0.) R=0.0000001
IF(Q*EQ.0.) Q=0.0000001
IF(O*EQ.0.) O=0.0000001
R1S=TT*R1T/R
R2S=TT*R2T/R
Q1S=TT*Q1T/Q
Q2S=TT*Q2T/Q
Q1S=TT*Q1T/Q
Q2S=TT*Q2T/Q
W=0.
DO 10 LN=1,NL
DO 8 I=1,NP
S=(W+CHEB(I)*T(LN))/TT
D=Q1S+STARTF(Q1S,Q2S,S)*O
E=Q1S+STARTF(Q1S,Q2S,S)*Q
F=R1S+STARTF(R1S,R2S,S)*R
XV(I,1,LN)=F*COS(E)*COS(D)
XV(I,2,LN)=F*SIN(E)*COS(D)
8 XV(I,3,LN)=F*SIN(D)
10 W=W+T(LN)
DO 79 LN=1,NL
CALL MULT3(U,XV(1,1,LN),X(1,1,LN),ND,NP)
DO 13 M=1,ND
VR(M,1)=V1(M+3)
13 VR(M,2)=V2(M+3)
RETURN
END

SUBROUTINE WYDER(E,F,AV,W,B,Q,G,H,TP,P)
DOUBLE PRECISION A,AA,BA,U,V,CHEB,WR,R1,R2,DSQRT,POWER,DPOWER,TP,
DOUBLE PRECISION E,AV(10,3),W(10,3),B(10,3),G(10,3,3),
C(10,3),H(10,3,3),R1(10,3),R2(10,3),R3(10,3),R4(10,3),P(10,3),
COMMON/CONS1/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10),
COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,NND2,NPD2,NPD1,NPDW,NB1,NB2,NC,NH,
DO 39 I=1,NP
Z=0.
DO 8 M=1,ND
8 Z=Z+E(I,M)**2
R2=1./Z
R1=DSQRT(R2)
R3(I)=TP*R2*R1
R4(I)=3.*R1*R3(I)
W(I)=1.
WR=0.
IF(NPDW*EQ.0) GO TO 10
W(I)=1./DSQRT(POWER(1./R1))
WR=-DPOWER(1./R1)*W(I)**2/2.
10 DO 19 M=1,ND
AV(I,M)=F(I,M)+R3(I)*E(I,M)
Q(I,M)=W(I)*AV(I,M)
C(I,M)=R1*E(I,M)
```fortran
19  D(I,M) = W*R*AV(I,M) - 3.0*R3(I)*C(I,M)
   DO 29  M = 1, ND
   DO 29  M = 1, NC
   H(I,M1,M2) = C(I,M2)*D(I,M1)
29  IF (M1.EQ.M2) H(I,M1,M2) = H(I,M1,M2) + R3(I)
   CONTINUE
   CALL MULT4(AA, Q, B, ND, NP, N, 0)
   CALL MULT2(Q, B, P, ND, NP)
   DO 49  I = 1, NP
   Z = 0.
   DO 45  M = 1, ND
   Q(I,M) = W(I)*B(I,M)
   45  Z = Z + Q(I,M)*D(I,M)
   DO 49  M = 1, ND
   B(I,M) = R3(I)*Q(I,M) + Z*C(I,M)
   IF (NC.LT.3) RETURN
   DO 79  I = 1, NP
   R1 = 0.
   DO 75  M = 1, ND
   R1 = R1 + Q(I,M)*C(I,M)
   DO 79  M = 1, ND
   D(I,M1) = 2.5*C(I,M1)*R1 - Q(I,M1)
   75  D(I,M2) = 1.0
   Z = C(I,M1)*D(I,M2) + C(I,M2)*D(I,M1)
   IF (M1.EQ.M2) Z = Z - R1
   79  G(I,M1,M2) = R4(I)*Z
   RETURN
   END

SUBROUTINE PAYOFF(X,P)
DOUBLE PRECISION A, AA, BA, U, V, CHEB, XV, AV, WV, TT, T, T2, T3, TP, PL, G
COMMON/PXXPX/PXX(30,6), PX(30,6), G(90,6)
COMMON/PARAM/NL, ND, NP, N, NP, NP, N2, NP, NP2, NP1, NP, NP, N2, N3, N4, N5, N6, N7, N8, N9, N10
COMMON/TIMEQ/T, T(6), T2(6), T3(6), TP(6), PL
COMMON/CONS/A(5,5,2), AA(5,5,2), BA(5,5,2), U(10,2), V(10,2), CHEB(10)
COMMON/OUT/XV(30,6), AV(30,6), WV(10,6)
P = 0.
DO 5  LN = 1, NL
   CALL MULT3(V, X(1, LN), XV(1, LN), ND, NP)
   CALL MULT4(BA, XV(1, LN), PXX(1, 9, LN), ND, NP, N1)
   CALL WYER(XV(1, LN), PXX(1, 9, LN), AV(1, LN), WV(1, LN), PXX(1, 1, LN),
   CPXX(1, 2, LN), G(1, LN), PXX(1, 6, LN), TP(LN), PL)
5  P = P + PL*T3(LN)
RETURN
END

SUBROUTINE SEARCH(X,P,DX)
DOUBLE PRECISION A, AA, BA, U, V, CHEB
DOUBLE PRECISION VR, DVR, VRA, VRAA, DAL, RHO
DOUBLE PRECISION X(30,6), DX(30,6), PXX, PX, P, PRO, RO, SECANT, Z, W
COMMON/PXXPX/PXX(30,30,6), PX(30,6)
```
SUBROUTINE DDERIV(RC, R, PR, NS)
DOUBLE PRECISION VR, DVR, VRA, VRAA, DAL
COMMON/VREL/VR(3, 2), VRA(3, 2, 2), VRAA(3, 2, 2), DVR(3, 2), DAL(3, 2, 2)
DOUBLE PRECISION DH(3, 2), DHRO(3, 2), H, RHO, U(3), V(3), DSQRT
DOUBLE PRECISION XV, AV, WY, TT, T2, T3, TP, PI2, PX, PXG
COMMON/PXX/PPX/PXX(30, 30, 6), PX(30, 6), G(90, 6)
COMMON/TIME/PXV(30, 6), AV(30, 6), WY(10, 6)
COMMON/PUT/NP1, NP2, D1, D2, H1, H2, PS, V1(6), V2(6)
RHO=RC+1.
DO 50 NB=1, 2
IF((NB.EQ.1 .AND. NB1.NE.1) .OR. (NB.EQ.2 .AND. NB2.NE.1)) GO TO 50
LN=(NB-1)*NL+(2-NB)
COMMON/PARAM/NL, ND, NP, N, NPO, NPM, ND2, NPD, NP1, NPOW, NB1, NB2, NO, NH
COMMON/VREL/VR(3, 2), VRA(3, 2, 2), VRAA(3, 2, 2), DVR(3, 2), DAL(3, 2, 2)
COMMON/CONS/A(5, 5, 2), AA(5, 5, 2), BA(5, 5, 2), U(10, 2), V(10, 2), CHEB(10)
PR=0.0
DO 5 LN=1, NL
CALL MULT3(V, DX(1, LN), PXX(1, 11, LN), ND, NP)
5 CALL MULT4(8A, PXX(1, 11, LN), PXX(1, 10, LN), ND, NP, N, 1)
CALL DDERIV(0, 0D0, P, PR, 0)
RO=SECANT(P, PR)
RHO=RH+1.0
DO 8 M=1, ND
IF(NB1.EQ.1) VR(M, 1)=DVR(M, 1)+RHO*(DAL(M, 1, 1)+.5*RHO*DAL(M, 2, 1))
8 IF(NB2.EQ.1) VR(M, 2)=DVR(M, 2)+RHO*(DAL(M, 1, 2)+.5*RHO*DAL(M, 2, 2))
DO 10 LN=1, NL
10 CALL MULTI(X(1, LN), DX(1, LN), X(1, LN), RO, ND, NP)
CALL ALIGN(X)
RETURN
END
H=T(LN)*(FLOAT(NB-1)*H2+FLOOR(2-NB)*H1)
Z=0.
W=0.
DO 30 M=1,ND
V(M)=DV(M)*NB+RHO*(DAL(M,1,NB)+.5*RHO*DAL(M,2,NB))
U(M)=DAL(M,1,NB)+RHO*DAL(M,2,NB)
Z=L+V(M)**2
30 W=W+U(M)*V(M)
Z=1.0/DSQRT(Z)
DO 40 M=1,ND
DH(M,NB)=H*(Z*V(M)-VR(M,NB))
40 DHRO(M,NB)=H*Z*(U(M)-V(M)*W*Z**2)
50 CONTINUE
IF(NS.NE.0) P=0.
PRO=0.
DO 5 LN=1,NL
IF(NS.EQ.0) GO TO 4
CALL MULT1(XV1(LN),PXX(1,11,LN),PXX(1,12,LN),RO,ND,NP)
CALL MULT1(PXX(1,9,LN),PXX(1,10,LN),PXX(1,13,LN),RO,ND,NP)
IF(NB1.EQ.1.AND.LN.EQ.1)
CCALL HAD(DH(1,1),PXX(1,12,LN),PXX(1,13,LN),1)
IF(NB2.EQ.1.AND.LN.EQ.NL)
CCALL HAD(DH(1,2),PXX(1,12,LN),PXX(1,13,LN),2)
CALL WYER(PXX(1,12,LN),PXX(1,13,LN),AV(1,LN),WV(1,LN),
CPXX(1,1,LN),PXX(1,2,LN),G(1,LN),PXX(1,6,LN),TP(LN),Z)
P=P+Z*T3(LN)
4 CONTINUE
DO 3 I=1,30
PXX(1,13,LN)=PXX(I,10,LN)
3 PXX(I,12,LN)=PXX(I,11,LN)
IF(NB1.EQ.1.AND.LN.EQ.1)
CCALL HAD(DHRO(1,1),PXX(1,12,LN),PXX(1,13,LN),1)
IF(NB2.EQ.1.AND.LN.EQ.NL)
CCALL HAD(DHRO(1,2),PXX(1,12,LN),PXX(1,13,LN),2)
CALL MULT2(PXX(1,13,LN),PXX(1,2,LN),W,ND,NP)
CALL MULT2(PXX(1,12,LN),PXX(1,1,LN),Z,ND,NP)
5 PRO=PRO+(W+Z)*T3(LN)*2.
RETURN
END

SUBROUTINE PDERIV
DOUBLE PRECISION XV,AV,WV,TT,T,T2,T3,TP,PI2,PXX,PX,G
COMMON/PXXPX/PXX(30,30,6),PX(30,6),G(90,6)
COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NO,NH
COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP16),PI2
COMMON/OUT/XV(30,6),AV(30,6),WV(10,6)
DO 5 LN=1,NL
L1=1
L2=NP
IF(LN.EQ.1) L1=MAX0(2,NB1+1)
IF(LN.EQ.NL) L2=NP-MAX0(1,NB2)
5 CALL DERIV(PXX(1,1,LN),PXX(1,1,LN),PXX(1,1,LN),PXX(1,2,LN),
CG(1,LN),PXX(1,6,LN),WV(1,LN),T3(LN),L1,L2)
RETURN
END

SUBROUTINE DERIV(PXX, PX, BD, Q, G, HH, W, T3, L1, L2)
COMMON/PARAM/ND, N1, N2, NP, N, NPW, N1D, N1P, N2P, N1PD, N2PD, N1PW, N2PW, N1PP, N2PP
DOUBLE PRECISION P(10, 10), B(10, 3), Q(10, 3), H(10, 3, 3), W(10), HH(10, 3, 3)
DOUBLE PRECISION PX(30), PXX(30, 30), G(10, 3, 3), R, S, T3
COMMON/SCRATCH/R(10, 10), S(10, 10)
DOUBLE PRECISION AA, BA, U, V, CHEB, C, D, E, F, Z
COMMON/CONS1/A(5, 5, 2), AA(5, 5, 2), BA(5, 5, 2), U(10, 2), V(10, 2), CHEB(10)
COMMON/CONS2/C(10, 10), D(10, 10), E(10, 10), F(10, 10)
DO 20 I = 1, NP
DO 20 M1 = 1, ND
DO 20 M2 = 1, ND
20 H(I, M1, M2) = HH(I, M1, M2)
L3 = MIN0(L1, 2)
L4 = MAX0(L2, NP1M1)
CALL MULT9(BA, Q, Q, ND, NP, N, 2)
DO 49 M1 = 1, ND
DO 39 I1 = L3, L4
39 P(I1) = B(I1, M1) + Q(I1, M1)
K1 = (L1 - 2) * ND + M1
DO 49 I1 = L1, L2
K1 = K1 + ND
49 PX(K1) = (P(I1) + V(I1, 1) * P(2) + V(I1, 2) * P(PNP1M1)) * T3
IF(NPW.EQ.0) GO TO 81
DO 80 I1 = 1, NP
DO 80 I2 = 1, I1
F(I1, I2) = W(I1) * C(I1, I2) * W(I2)
80 F(I1, I1) = F(I1, I1)
CALL MULT9(BA, F, E, NP, N, 1)
CALL MULT9(BA, E, D, NP, N, 2)
81 DU 99 M1 = 1, ND
DU 99 M2 = 1, M1
DO 89 I1 = L3, L4
J2 = L4
IF(M1.EQ.M2) J2 = I1
DO 89 I2 = L3, J2
Z = 0.
DO 85 M3 = 1, ND
85 Z = Z + H(I1, M1, M2) * H(I2, M3, M2)
Z = H(I1, M2, M1) * E(I2, I1) + E(I1, I2) * H(I2, M1, M2) + Z * F(I1, I2)
IF(NP1W.EQ.3) AND I1.EQ.I2) Z = Z + G(I1, M1, M2)
IF(M1.EQ.M2) Z = Z + D(I1, I2)
IF(M1.EQ.M2) S(I2, I1) = Z
89 S(I1, I2) = Z
DO 95 I1 = L3, L4
J2 = L4
IF(M1.EQ.M2) AND (I1.NE.2) AND (I1.NE.NP1M1) J2 = I1
DO 95 I2 = L3, J2
95 R(I1, I2) = S(I1, I2) + V(I2, 1) * S(I1, 2) + V(I2, 2) * S(I1, NP1M1)
K1 = (L1 - 2) * ND + M1
DO 99 I1 = L1, L2
99 RETURN
K1 = K1 + ND
J2 = L2
IF (M1.EQ.M2) J2 = I1
K2 = (L1 - 2) * ND + M2
DO 99 I2 = L1, J2
   K2 = K2 + ND
   PX(K1, K2) = (R(I1, I2) + V(I1, 1) * R(1, I2) + V(I1, 2) * R(NPM1, I2)) * T3
99 PX(K2, K1) = PX(K1, K2)
RETURN
END

SUBROUTINE PATCH(T, H1, H2)
DOUBLE PRECISION XV, AV, XV, VA(3), VB(3), VAA(3), VAB(3), VBB(3)
DOUBLE PRECISION PX, PX, T(6), Z, Q, R, P, S, VR, DVR, VRA, VRAA, DAL
COMMON/ PARAM/NL, ND, NP, N, NPD, NPM1, ND2, NPD2, NPD1, NPOW, NB1, NB2, NO, NH
COMMON/ VREL/ VR(3, 2), VRA(3, 2, 2), VRAA(3, 2, 2, 2), DVR(3, 2), DAL(3, 2, 2)
COMMON/ PX, PX, PX(30, 30, 6), PX(30, 6)
COMMON/ OUT/ XV(30, 6), AV(30, 6), WV(30, 2)
FNO = FLOAT(NH - 2)
19 DO 20 LN = 1, NL
   DO 20 I = 1, NPD
      XV(I, LN) = PX(I, LN)
20 CONTINUE
AV(I, LN) = PX(I, LN)
DO 10 NB = 1, Z
   IF (NB.EQ.1 .AND. NB1 .NE. 1) OR (NB.EQ.2 .AND. NB2 .NE. 1) GO TO 10
   LN = (NB - 1) * NL + (2 - NB)
   P = T(LN) * (H2 * FLOAT(NB - 1) + H1 * FLOAT(2 - NB))
   M1 = NPD2 * (NB - 1) + NO * (2 - NB)
   M2 = (NPD2 + 1) * (NB - 1) + ND2 * (2 - NB)
   M3 = (NPD2 + 2) * (NB - 1) + (ND2 - 1) * (2 - NB)
   DO 5 M = 1, ND
      VA(M) = P * VRA(M, 1, NB)
      VB(M) = P * VRA(M, 2, NB)
      VAB(M) = P * VRAA(M, 1, 2, NB)
      VBB(M) = P * VRAA(M, 2, 2, NB)
  5 CONTINUE
   VAA(M) = P * VRAA(M, 1, 1, NB)
   DO 7 I = 1, NPD
      Q = 0.
      R = 0.
      DO 6 M = 1, ND
         R = R + PX(I, M1, M, LN) * VB(M)
       6   Q = Q + PX(I, M1, M, LN) * VA(M)
      WV(I, 2) = R
    7   WV(I, 1) = Q
      P = 0.
      S = 0.
      Q = 0.
      R = 0.
      Z = 0.
      DO 8 M = 1, ND
         S = S + PX(M1, M, LN) * VB(M)
         P = P + PX(M1, M, LN) * VA(M)
      Z = Z + WV(M1, M, 1) * VB(M) + FNO * PX(M1, M, LN) * VAB(M)
R=R+\text{wv}(M1+M,2) \ast \text{vb}(M) + \text{fn0} \ast \text{px}(M1+M,LN) \ast \text{vbb}(M)
Q=Q+\text{wv}(M1+M,1) \ast \text{va}(M) + \text{fn0} \ast \text{px}(M1+M,LN) \ast \text{vaa}(M)
\text{px}(M3,LN)=P
\text{px}(M4,LN)=S
\text{pxx}(M3,M3,LN)=Q
\text{pxx}(M4,M4,LN)=R
\text{if}(\text{nb} . \text{eq} .1) \text{pxx}(M3,M4,LN)=Z
\text{if}(\text{nb} . \text{eq} .2) \text{pxx}(M4,M3,LN)=Z
\text{do} 9 \text{i}=1,\text{npd2}
\text{k}=\text{nd2}+1
\text{if}(\text{nb} . \text{eq} .2) \text{pxx}(M3,I,LN)=\text{wv}(I,1)
\text{if}(\text{nb} . \text{eq} .2) \text{pxx}(M4,I,LN)=\text{wv}(I,2)
\text{if}(\text{nb} . \text{eq} .1) \text{pxx}(K,M4,LN)=\text{wv}(K,2)
\text{if}(\text{nb} . \text{eq} .1) \text{pxx}(K,M3,LN)=\text{wv}(K,1)
9 \text{continue}
\text{if}(\text{nl} . \text{eq} .1) \text{return}
\text{do} 2 \text{ln}=2,\text{nl}
Q=\text{t}(\text{ln}-1)
R=\text{t}(\text{ln})
\text{do} 1 \text{j}=1,\text{npd2}
\text{do} 1 \text{i}=1,\text{nd}
\text{pxx}(\text{nd2}+J,\text{nd}+I,LN)=\text{pxx}(\text{nd2}+J,\text{nd}+I,LN) \ast \text{r}
Z=\text{pxx}(\text{npd1}+I,J,LN-1)
\text{pxx}(\text{npd1}+I,J,LN-1)=-\text{pxx}(\text{npd2}+I,J,LN-1) \ast \text{q}
1 \text{pxx}(\text{npd2}+I,J,LN-1)=Z
\text{do} 4 \text{i}=1,\text{nd}
Z=\text{pxx}(\text{npd1}+I,LN-1)+\text{pxx}(I,LN)
\text{pxx}(\text{npd1}+I,LN-1)=-\text{pxx}(\text{npd2}+I,LN-1) \ast \text{q}+\text{pxx}(\text{nd1}+I,LN) \ast \text{r}
\text{pxx}(\text{npd2}+I,LN-1)=Z
\text{do} 4 \text{j}=1,\text{nd}
\text{if}(\text{j} . \text{gt} . \text{i}) \text{go to} 4
Z=\text{pxx}(\text{npd1}+I,\text{npd1}+J,LN-1)+\text{pxx}(I,J,LN)
\text{pxx}(\text{npd1}+I,\text{npd1}+J,LN-1)=\text{pxx}(\text{npd2}+I,\text{npd2}+J,LN-1) \ast \text{q} \ast \text{q} \ast \text{r} \ast \text{r}
\text{c}+\text{pxx}(\text{nd}+I,\text{nd}+J,LN) \ast \text{r} \ast \text{r}
\text{pxx}(\text{npd2}+1,\text{npd2}+1,LN-1)=Z
4 \text{pxx}(\text{npd1}+I,\text{npd2}+J,LN-1)=\text{pxx}(\text{nd}+I,\text{nd}+J,LN) \ast \text{r} \ast \text{pxx}(\text{npd2}+I,\text{npd1}+J,LN-1) \ast \text{q}
2 \text{continue}
\text{return}
\text{end}

\text{subroutine modify(px,dx,hi,h2)}
\text{double precision px(30,6),dx(10,3,6),vr,vra,vraa,dal,dv}
\text{double precision tt,t,t2,t3,tp,p12,dal1(2),dal2(2)}
\text{commcn/timeq/tt,t(6),t1(6),t2(6),t3(6),tp(6),p12}
\text{common/param/nl,nd,np,n,pd,npm1,nd2,npd2,npd1,npd1,npo,nb1,nb2,n0,nh}
\text{common/vrel/vr(3,2),vra(3,2,2),vraa(3,2,2,2),dv(3,2),dal(3,2,2)}
\text{dal1(1)=-px(\text{nd2},1)}
\text{dal1(2)=-px(\text{npd2}+1,\text{nl})}
\text{dal2(1)=float(2-\text{nd}) \ast px(\text{nd2}-1,1)}
\text{dal2(2)=float(2-\text{nd}) \ast px(\text{npd2}+2,\text{nl})}
\text{do 50 nb=1,2}
\text{if}(\text{nb} . \text{eq} .1 . \text{and} . \text{nb} . \text{eq} .1 . \text{or} . (\text{nb} . \text{eq} .2 . \text{and} . \text{nb} . \text{eq} .1)) \text{go to} 50
\text{do 45 m=1,nd}

D-12
DVR(M,NB)=DAL(M,1,NB)*VRA(M,1,NB)+VRA(M,2,NB)*DAL2(NB)
DAL(M,2,NB)=FLOAT(NH-2)*(VRAA(M,1,1,NB)*DAL1(NB)**2+VRAA(M,2,2,NB)
C*DAL2(NB)**2+2.*VRAA(M,1,2,NB)*DAL1(NB)*DAL2(NB))
45 CONTINUE
50 DO 5 M=1,ND
51 DO 4 LN=1,NL
52 DC I=1,NP
53 K=(I-1)*ND+M
4 DX(I,M,LN)=-PX(K,LN)
4 IF(NB1.EQ.0) DX(2,M,1)=0.
4 IF(NB2.EQ.0) DX(NPM1,M,LN)=0.
5 DX(1,M,1)=0.
5 DX(NP,M,LN)=0.
5 IF(NL.EQ.1) RETURN
5 DO 10 LN=2,NL
5 DO 10 M=1,ND
6 DX(2,M,LN)=T(LN)*DX(2,M,LN)
6 DX(NPM1,M,LN-1)=-DX(2,M,LN)*T(LN-1)/T(LN)
10 DX(NP,M,LN-1)=DX(1,M,LN)
RETURN
END

SUBROUTINE ALIGN(X)
DOUBLE PRECISION X(10,3,6),TT,T,T2,T3,TP,P12,VR,DVR,VRA,VRAA,DAL
COMMON/VREL/VR(3,2),VRAA(3,2,2),VRAA(3,2,2),DVR(3,2),DAL(3,2,2)
COMMON/NP1,NP2,D1,D2,H1,H2,PS,V1(6),V2(6)
COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),P12
COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,NPD2,NPM1,NPM1,ND2,NPD2,NPD2,NPOW,NB1,NB2,NO,NH
3 DO 1 M=1,ND
3 DVR(M,1)=VR(M,1)-DVR(M,1)
3 DVR(M,2)=VR(M,2)-DVR(M,2)
3 IF(NB1.EQ.1) CALL VCAL(VR(1,1),VRA(1,1,1),VRAA(1,1,1,1),ND)
3 IF(NB2.EQ.1) CALL VCAL(VR(1,2),VRA(1,1,2),VRAA(1,1,1,2),ND)
3 DO 10 M=1,ND
5 IF(NB1.NE.0) X(2,M,1)=T(1)*(V1(M+3)+H1*VR(M,1))
5 IF(NB2.NE.0) X(NPM1,M,LN)=T(LN)*(-V2(M+3)+H2*VR(M,2))
5 X(1,M,1)=V1(M)
10 X(NP,M,LN)=V2(M)
5 IF(NL.EQ.1) RETURN
5 DO 5 LN=2,NL
5 DO 5 M=1,ND
5 X(NPM1,M,LN-1)=-X(2,M,LN)*T(LN-1)/T(LN)
5 X(NP,M,LN-1)=X(1,M,LN)
RETURN
END

SUBROUTINE FIXUP
COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NO,NH
DOUBLE PRECISION TT,T,T2,T3,TP,P12
COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),P12
D-13
COMMON/CONSL/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
COMMON/SCRACH/R(10,10),S(10,10)
COMMON/PX/PX/PX(30,30,6),PX(30,6),G(10,3,3,6)
COMMON/OUT/XV(30,6),AV(30,6),MV(10,6)
DO 20 LN=1,NL
DO 20 I=1,NPD
DO 15 J=1,I
15 PX(I,J,LN)=PX(J,I,LN)
PX(I,LN)=XV(I,LN)
DO 20 I=1,NPD
20 PX(I,I,LN)=AV(I,I,LN)
IF(ND.EQ.2) RETURN
DO 65 I1=1,NP
DO 65 I2=1,NP
Z=V(I1,1)*V(I2,1)
IF(I1.EQ.2)Z=Z+V(I2,1)
IF(I2.EQ.2)Z=Z+V(I1,1)
R(I1,I2)=Z
DO 100 LN=1,NL
L1=1
L2=NP
IF(LN.EQ.1) L1=MAX0(L1,NB1+1)
IF(LN.EQ.NL) L2=NP-MAX0(1,NB2)
L3=MN0(L1,L2)
L4=MAX0(L2,NPM1)
DO 99 M1=1,ND
DO 99 M2=1,M1
K1=(L1-I2)*ND*M1
DO 99 I1=L1,L2
K1=K1+ND
J2=L2
IF(M1.EQ.M2) J2=I1
K2=(L1-I2)*ND*M2
DO 99 I2=L1,J2
K2=K2+ND
Z=G(2,M1,M2,LN)*R(I1,I2)+G(NPM1,M1,M2,LN)*S(I1,I2)
IF(I1.EQ.I2)Z=Z+G(1,I1,M1,M2,LN)
PXX(K1,K2,LN)=PXX(K1,K2,LN)+Z*T3(LN)
99 PX(K1,K2,LN)=PXX(K1,K2,LN)
100 CONTINUE
RETURN
END

SUBROUTINE TSHIFT(X,XV,NL,NLP,ND,NP,N)
DOUBLE PRECISION X(30,6),XV(10,3,6)
DOUBLE PRECISION A,AA,BA,U,W,CHEB,TT,T,T2,T3,TP,PI2,TS
COMMON/TIME/T,T(6),T(6),T(6),TP(6),PI2
COMMON/CONSL/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
DIMENSION THETA(61),TIME(61)
IF(NL.EQ.1.AND.NLP.EQ.1) RETURN
EX=0.
DO 50 LN=1,NL
50 CALL MUL3(V,X(1,LN),XV(1,1,LN),ND,NP)
D-14
I=1
THET=0.
TS=0.
DO 3 I=1,NL
DO 2 J=1,NP
I=I+1
X=V(J,1,I)
Y=V(J,2,I)
THETA(I)=ATAN2(Y,X)
TIME(I)=TS+T(I)*CHEB(J)
DO 1 K=1,6
IF(THT.LT.THETA(I)+.01) GO TO 27
1 THETA(I)=THETA(I)+PI2
27 IF(THETA(I).GT.THETA+1.57.AND.I.GT.2)GO TO 950
THET=THETA(I)
THETA(I)=THETA*(XC**2+YC**2)**EX
2 CONTINUE
3 TS=T(I)+TS
SUBARC=(THETA(I)-THETA(2))/FLOAT(NLP)
I=1
ARC=THETA(2)+SUBARC
DO 4 J=4,I
IF(THETA(J-1).LT.ARC) GO TO 4
TP(I)=TIME(J-2)+(ARC-THETA(J-2))*(TIME(J-1)-TIME(J-2))
ARC=ARC+SUBARC
I=II+1
IF(I.EQ.NLP) GO TO 5
4 CONTINUE
5 TP(NLP)=TT
IF(NLP.LT.2) GO TO 900
DO 6 I=2,NLP
6 TP(NLP+2-I)=TP(NLP+2-I)-TP(NLP+1-I)
900 CALL REDIST(X,XV,NL,NLP,ND,NP,N)
950 RETURN
END

SUBROUTINE REDIST(X,Y,NL,NLP,ND,NP,N)
DOUBLE PRECISION A,AA,BA,U,V,CHEB,TT,T2,T3,TP,PI2,S,W
DIMENSION G(3)
DOUBLE PRECISION X(30,6),Y(10,3,6)
COMMON/TIMES/T,TT,T(6),T2(6),T3(6),TP(6),PI2
COMMON/CONS/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
DO 5 LN=1,NL
5 CALL MULT4(A,Y(1,1,LN),X(1,LN),ND,NP,N,1)
W=0.
DO 20 LN=1,NLP
DO 13 I=1,NP
S=W+TP(LNP)*CHEB(I)
20 CALL MODLEG(T,NL,S,LN)
CALL POLEV(X(1,LN),G,S,ND,NP,N)
DO 13 M=1,ND
13 Y(I,1,LNP)=G(M)
D-15
20 \( W = W + TP(LNP) \)
DO 30 LNP = 1, NLP
CALL MULT3(U, Y(1, 1, LNP), X(1, LNP), ND, NP)
T(LNP) = TP(LNP)
T2(LNP) = 1. / T(LNP)**2
T3(LNP) = T2(LNP) / T(LNP)
30 TP(LNP) = (P12 * T(LNP))**2
NL = NLP
RETURN
END

SUBROUTINE OUTCAL
DOUBLE PRECISION TT, T, T2, T3, TP, P12, S, XV, AV, WV
COMMNC/PUT/NP1, NP2, D1, D2, H1, H2, PS, V1(6), V2(6)
COMMNC/TIMEQ/TT, T(6), T2(6), T3(6), TP(6), P12
COMMNC/PARAM/NL, ND, NP, N, NP1, NP2, NP3, NP4, NP5, NP6, NP7,
COMMNC/OUT/XV(30, 6), AV(30, 6), WV(10, 6)
COMMNC/SCRACH/B(26, 12), G(3), H(3)
G(3) = 0.
H(3) = 0.
DO 20 I = 1, 26
S = TT*FLOAT(I-1) / 25.
B(I, 1) = 365.25*S
CALL MODLEG(T, NL, S, LN)
CALL POLEVL(XV(I, LN), Z, S, I, NP, N)
B(I, 12) = 1. / Z**2
CALL POLEVL(XV(I, LN), G, S, ND, NP, N)
CALL POLEVL(AV(I, LN), H, S, ND, NP, N)
W = 0.
L = 0.
DO 15 M = 1, 3
B(I, M+1) = G(M)
B(I, M+7) = H(M)
W = W + G(M)**2
15 Z = Z + H(M)**2
B(I, 5) = SQRT(W)
B(I, 11) = SQRT(Z)
Z = 57.29578*ATAN2(G(2), G(1))
IF(I.EQ.1) GO TO 17
DO 16 K = 1, 3
IF(Z - B(I-1, 6) .GT. 180.) Z = Z - 360.
16 IF(Z - B(I-1, 6) .LT. -180.) Z = Z + 360.
17 B(I, 6) = Z
20 B(I, 7) = 57.29578*ATAN(G(3) / SQRT(G(2)**2 + G(1)**2))
CALL OUTPUT(E, NP1, NP2, D1, D2, H1, H2, PS, NL, ND, NP, NB1, NB2, NPW)
RETURN
END

SUBROUTINE OUTPUT(A, P1, P2, D1, D2, H1, H2, JV, NL, ND, NP, NB1, NB2, NPW)
DIMENSION A(26, 12), UNCGST(3)
DOUBLE PRECISION PLANET(12), POWOPT(2)
INTEGER P1, P2
DATA UN CST/4H UN/,4H ,4H /  
DATA POW OPT/ 8HCONSTANT , 8HVARIABLE /  
DATA PLANET/ 8HMERCURY , 8HVENUS , 8HEARTH , 8HMARS ,  
18HJUPITER , 8HSATURN , 8HURANUS , 8HNEPTUNE , 8HPLUTO ,  
2 8HTENTH , 8HELEVENTH , 8HTWELFTH /  
WRITE(6,4321)  
4321 FORMAT(1H1)  
WRITE(6,101) POW OPT(NPOW+1)  
WRITE(6,120) ND,NL,NP  
WRITE(6,104) JV  
WRITE(6,102) PLANET(P1) , PLANET(P2)  
WRITE(6,103) D1,D2  
WRITE(6,121) PLANET(P1) , H1 , UN CST(NB1+1) , UN CST(3)  
WRITE(6,121) PLANET(P2) , H2 , UN CST(NB2+1) , UN CST(3)  
WRITE(6,109)  
DO 1 I=1,26  
1 WRITE(6,100) (A(I,J),J=1,12)  
RETURN  
100 FORMAT(9.2,4F8.3,6F8.2,F8.3)  
101 FORMAT( 25X,16Hvariable THRUST ,A8,21H POWER TRAJECTORY )  
102 FORMAT(//15X,16Hdeparture PLANET ,2X, A8,  
 1 10X,14Harrival PLANET ,2X, A8)  
103 FORMAT(//15X,16Hdeparture DAY ,F10.0,  
 1 10X,14Harrival DAY ,F10.0 )  
104 FORMAT(//15X,25Hperformance INDEX (JV) , F10.3 )  
109 FORMAT(//34H , THETA PHI  
 1 ,25H R  
 4 ,23H AX AY AZ  
 2 ,30H MAG. P/P0  
 3 //)  
120 FORMAT(//15X,15,29H dimensional TRAJECTORY USING , I2  
 2 ,25H patched polynomials with , I3  
 1 ,18H chebychev points )  
121 FORMAT(//15X,19Hexcess VELOCITY AT ,A8, F10.3 ,2X,A4,  
111HCCNstrained , A4 )  
END  

SUBRUTINE OUPUT1(P1,P2,D1,D2,H1,H2,JV,NL,ND,NP,NB1,NB2,NPOW)  
DIMENSION UN CST(3)  
DIMENSION A(4)  
DATA A/4H**** , 4H**** , 4H**** , 4H**** /  
INTEGER P1,P2  
DOUBLE PRECISION PLANET(12),POW OPT(2)  
DATA UN CST/4H UN/,4H ,4H /  
DATA POW OPT/ 8HCONSTANT , 8HVARIABLE /  
DATA PLANET/ 8HMERCURY , 8HVENUS , 8HEARTH , 8HMARS ,  
18HJUPITER , 8HSATURN , 8HURANUS , 8HNEPTUNE , 8HPLUTO ,  
2 8HTENTH , 8HELEVENTH , 8HTWELFTH /  
WRITE(6,700) A,A,A,A,A,A  
WRITE(6,5432)  
WRITE(6,700) A,A,A,A,A,A,A  
WRITE(6,101) POW OPT(NPOW+1) ,ND,NL,NP  
WRITE(6,221) PLANET(P1) , D1 , PLANET(P2) , D2
WRITE(6,121) PLANET(P1), H1, UNCST(NB1+1),PLANET(P2),H2 ,
UNCST(NB2+1), UNCST(3) WRITE(6,104) JV,POWOPT(NPOW+1),UNCST(3)
WRITE(6,700) A,A,A,A,A,A,A,
WRITE(6,5432) WRITE(6,700) A,A,A,A,A,A,A,
RETURN
121 FORMAT(5X,19HECESS VELOCITY AT ,A8,2HIS,E20.6,2X,A4,
211HCUNSTRAINED ,7H , AT , A8,2HIS , E20.6,2X,A4,
111HCUNSTRAINED , A4 )
101 FORMAT(5X,16HVARIEL THRUST ,A8,20H POWER TRAJECTORY IN , 12 ,1X,
116HDIMENSIONS USING ,12 , 8H LEGS OF ,13,17H CHEBYCHEV POINTS )
221 FORMAT(5X,20HDEPARTURE PLANET IS ,A8,14H AT JULIAN DAY ,F10.1 ,
1 20H ARRIVAL PLANET IS ,A8,14H AT JULIAN DAY ,F10.1 )
104 FORMAT(5X,21HPERFORMANCE INDEX IS ,E20.6,4X,A8,10H POWER ,A4 )
700 FORMAT(1H ,28A4 )
5432 FORMAT(10X/10X)
END

SUBROUTINE VCAL(VR,VRA,VRAA,ND)
DOUBLE PRECISION VR(3),VRA(3,2),VRAA(3,2,2),DSQRT,Z
IF(ND,EQ,2) VR(3)=0.
Z=DSQRT(1./(VR(1)**2+VR(2)**2+VR(3)**2))
DO 10 M=1,3
VR(M)=Z*VR(M)
VRAA(M,1,1)=-VR(M)
10 VRAA(M,2,2)=-VR(M)
VRAA(3,1,1)=0.
VRA(1,1)=-VR(2)
VRA(2,1)=VR(1)
VRA(3,1)=0.
VRA(3,2)=DSQRT(VR(1)**2+VR(2)**2)
Z=VR(3)/VRA(3,2)
VRA(1,2)=-Z*VR(1)
VRA(2,2)=-Z*VR(2)
VRAA(1,1,2)=-VRAA(2,2)
VRAA(2,1,2)=VRAA(1,2)
VRAA(3,1,2)=0.
VRAA(1,2,1)=VRAA(1,1,2)
VRAA(2,2,1)=VRAA(2,1,2)
VRAA(3,2,1)=VRAA(3,1,2)
RETURN
END

SUBROUTINE HAD(DH,X,BAX,NB)
DOUBLE PRECISION A,AA,BAU,V,CHEB,DF(3),X(10,3),BAX(10,3)
DOUBLE PRECISION VDH(2),VDHP,VDHM,V1,V2
COMMON/PARAM/NL,ND,NP,N,NPO,NM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NG,NH
COMMON/CONS/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),VU(10,2),CHEB(10)
DC 50 M=1,ND
VDH(NB)=(1.+V(2,1))*DH(M)
VDH(3-NB)=V(2,2)*DH(M)
X(Z, M) = X(Z, M) + VDH(1)
X(NP*M+1, M) = X(NP*M+1, M) + VDH(2)
VDHP = VDH(2) + VDH(1)
VDHM = VDH(2) - VDH(1)
DO 50 I = 1, N
K = NP + 1 - I
V1 = BA(I, 2, 1) * VDHM
V2 = BA(I, 2, 2) * VDHHP
BAX(I, M) = BAX(I, M) + V2 - V1
50    BAX(K, M) = BAX(K, M) + V2 + V1
RETURN
END

SUBROUTINE AVTEST(AV, T2, NLP, NL, ND, NP)
DOUBLE PRECISION AV(10, 3, 6), T2(6)
DIMENSION AVM(2, 6)
IF (NL .EQ. 1) NLP = 2
IF (NL .EQ. 1) RETURN
DO 525 LN = 1, NL
Z1 = 0.
Z2 = 0.
DO 500 M = 1, ND
Z1 = Z1 + AV(1, M, LN)**2
500    Z2 = Z2 + AV(NP, M, LN)**2
AVM(1, LN) = SQRT(Z1) * T2(LN)
AVM(2, LN) = SQRT(Z2) * T2(LN)
DG 10 LN = 2, NL
RATIO = (AVM(1, LN) - AVM(2, LN-1)) / AMINI(AVM(1, LN), AVM(2, LN-1))
IF (ABS(RATIO) .GT. 1) NLP = MINO(6, NL + 1)
10 CONTINUE
RETURN
END

SUBROUTINE PLEVEL(Y, G, S, ND, NP, N)
DOUBLE PRECISION Y(10, 3), H(10), S, Z
DIMENSION G(3)
H(1) = 1.
H(2) = 2.*S - 1.
Z = 2.*H(2)
DO 1 I = 3, NP
1     H(I) = Z*H(I-1) - H(I-2)
3     DG 10 M = 1, ND
Z = 0.
DO 5 I = 1, N
K = NP + 1 - I
Z = Z + (Y(K, M) - Y(I, M)) * H(2*I) + (Y(K, M) + Y(I, M)) * H(2*I - 1)
10    G(M) = .5*Z
RETURN
END

SUBROUTINE MODLEG(T, NL, S, LN)
DOUBLE PRECISION T(NL), S, W
W = 0.
DO 10 K = 1, NL
    LN = K
    W = W + T(K)
    IF (S .LT. W) GO TO 11
10 CONTINUE
11 S = 1. - (W - S) / T(LN)
RETURN
END

SUBROUTINE MULT1(X, Y, Z, RO, ND, NP)
DOUBLE PRECISION X(10, 3), Y(10, 3), Z(10, 3), RO
DO 1 M = 1, ND
DO 1 I = 1, NP
1 Z(I, M) = X(I, M) + RO * Y(I, M)
RETURN
END

SUBROUTINE MULT2(X, Y, Z, ND, NP)
DOUBLE PRECISION X(10, 3), Y(10, 3), Z
Z = 0.
DO 1 M = 1, ND
DO 1 I = 1, NP
1 Z = Z + X(I, M) * Y(I, M)
RETURN
END

SUBROUTINE MULT3(V, X, Y, ND, NP)
DOUBLE PRECISION V(10, 2), X(10, 3), Y(10, 3), Z1, Z2
DO 5 M = 1, ND
    Z1 = X(2, M)
    Z2 = X(NP - 1, M)
    DO 3 I = 1, NP
        Z1 = Z1 + V(I, 1) * X(I, M)
        Z2 = Z2 + V(I, 2) * X(I, M)
    3 Y(I, M) = X(I, M)
    Y(2, M) = Z1
5 Y(NP - 1, M) = Z2
RETURN
END

SUBROUTINE MULT4(A, X, Y, ND, NP, N, L)
DOUBLE PRECISION A(5, 5, 2), X(10, 3), Y(10, 3), U1(5), U2(5), V1, V2
DO 99 M = 1, ND
    DO 49 I = 1, N
        K = NP + 1 - I
        U1(I) = X(K, M) - X(I, M)
    49 U2(I) = X(K, M) + X(I, M)
DO 99 I = 1, N
99 CONTINUE
K=NP+1-I
V1=0.
V2=0.
DO 79 J=1,N
IF(L.GT.1) GO TO 75
V1=V1+A(I,J,1)*U1(J)
V2=V2+A(I,J,2)*U2(J)
GO TO TC 79
75 V1=V1+A(J,1,1)*U1(J)
V2=V2+A(J,1,2)*U2(J)
79 CONTINUE
Y(I,N)=V2-V1
99 Y(K,N)=V2+V1
RETURN
END

SUBROUTINE MULT7(A,B,C,N,N)
DOUBLE PRECISION A(5,5,2),B(5,5,2),C(5,5,2),D(5,5,2),Z
DO 2 I=1,N
M=N
IF(NS.EQ.2) M=I
DO 2 J=1,M
DO 2 K=1,2
Z=0.
DO 1 L=1,N
1 Z=Z+A(L,I,K)*B(L,J,K)
IF(NS.EQ.2) D(J,I,K)=Z
2 D(I,J,K)=Z
DO 3 K=1,2
DO 3 I=1,N
DO 3 J=1,N
3 C(I,J,K)=D(I,J,K)
RETURN
END

SUBROUTINE MULT8(A1,A2,B,NP,N,N)
DOUBLE PRECISION A1(5,5,2),A2(5,5,2),B(10,10)
DO 2 I=1,N
K=NP+1-I
M=N
IF(NS.EQ.2) M=I
DO 2 J=1,M
L=NP+1-J
B(K,L)=A2(I,J,2)+A2(I,J,1)
B(K,J)=A1(I,J,2)-A1(I,J,1)
B(I,L)=A2(I,J,2)-A2(I,J,1)
IF(NS.EQ.1) GO TO 2
B(J,I)=B(I,J)
B(L,K)=B(K,L)
B(J,K)=B(K,J)
B(L,I)=B(I,L)
SUBROUTINE MULT9(A,B,C,NP,N,NS)
DOUBLE PRECISION A1(5,5,2),A2(5,5,2),A(5,5,2),B(10,10),C(10,10)
DO 1 I=1,N
  K=NP+I-1
  DO 1 J=1,N
    L=NP+I-J
    A1(I,J,1)=B(J,I)-B(J,K)
    A1(I,J,2)=B(J,I)+B(J,K)
    A2(I,J,1)=B(L,K)-B(L,I)
    A2(I,J,2)=B(L,K)+B(L,I)
  CALL MULT7(A,A1,A1,N,NS)
  CALL MULT7(A,A2,A2,N,NS)
  CALL MULT8(A1,A2,C,NP,N,NS)
  RETURN
END

SUBROUTINE CONST(NP,N,NCON)
DOUBLE PRECISION FNP1,F1,F2,DCOS,DSIN,Z,A,AA,BA,U,V,CHEB,C,D,E,F
DOUBLE PRECISION P(10),Q(10),R(10)
COMMON/CONS1/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
COMMON/CONS2/C(10,10),D(10,10),E(10,10),F(10,10)
IF(NCON.GT.0) GO TO 59
NPM1=NP-1
FNP1=FLOAT(NPM1)
DO 1 I=1,NP
  Z=3.14159265358979323846D0*FLOAT(I-1)/FNP1
  P(I)=DSIN(Z)**2
  CHEB(I)=DSIN(Z/2.)**2
  IF(I.NE.1) Q(I)=2.*(-1.)**I/CHEB(I)
1  R(I)=0.DCOS(Z)
  Q(1)=-(1.+2.*FNP1**2)/3.*Q(NP)=.5*Q(NP)
DO 4 I=1,N
  DO 4 J=1,N
    IF(I.EQ.J) GO TO 3
    II=1+ABS(I-J)
    JJ=I+J-1
    Z=4.*(-1.)**((I+J+1)/(P(II)*P(JJ))
    IF(I.EQ.1) GO TO 2
    BA(I,J,1)=Z*((1.+P(J)/P(I))*R(I)**2+2.*R(J)**2)
    BA(I,J,2)=Z*R(I)*R(J)*((3.+P(J)/P(I))
      GO TO 3
2    BA(I,J,2)=Z*(8.+2.*Q(1)*P(J))*R(J)
    BA(I,J,1)=Z*(8.+2.*(Q(1)-2.)*P(J))
3    DO 4 K=1,2
      L=(2*I-K)*(J-1)
      M=L/NPM1
      L=L-M*NPM1+1
F1=FLOAT(4*(J+1-K)**2-1)
F2=FLOAT(4*(J-1)**2-1)
AA(I,J,K)=-5/F1-.5/F2
4 A(I,J,K)=2.*(-1.)*M*R(L)/FNPM1
Z=64.*R(2)/P(2)**2
DO 5 I=1,NP
K=NP+1-I
U(I,1)=Q(I)
IF(I.EQ.2) U(I,1)=Q(1)-1.
V(I,1)=(Q(NPML)-Q(K))*Q(2)*Q(I))/Z
IF(I.EQ.2) V(I,1)=Q(2)/Z-1.
IF(I.EQ.NPML) V(I,1)=-Q(NPML)/Z
U(K,2)=U(I,1)
5 V(K,2)=V(I,1)
DO 50 J=1,N
IF(J.EQ.1) GO TO 35
BA(J,J,2)=2.*((1.*FNPL)**2)/3.-(1.*3.*R(2*J-1))/P(2*J-1)/P(J)
BA(J,J,1)=BA(J,J,2)-8.*R(2*J-1)/P(2*J-1)
GO TO 40
35 BA(J,J,2)=(4.*(4.*FNPL)**2*(FNPL**2-5.)))/15.
BA(J,J,1)=BA(J,J,2)-(8.*(1.*FNPL)**2)/3.
40 A(I,1,2)=5*A(I,J,2)
A(N,J,1)=5*A(N,J,1)
DO 50 K=1,2
BA(J,1,K)=5*BA(J,1,K)
50 A(J,1,K)=5*A(J,1,K)
CALL MULT7(AA,A,AA,N,1)
CALL MULT7(AA,A,AA,N,2)
CALL MULT8(AA,A,AA,C,NP,N,2)
59 DO 60 I=1,NP
DO 60 J=1,NP
60 F(I,J)=C(I,J)
CALL MULT9(BA,F,E,NP,N,1)
CALL MULT9(BA,E,D,NP,N,2)
NCON=1
RETURN
END

SUBROUTINE SQROOT(A,B,X,N,M,NCPL,N1,N2,IND)
DOUBLE PRECISION A(30,30,6),B(30,30,6),X(30,30,6),W,Z,DSQRT
IND=3
NCPL=N1
NCPL3=NCPL
NCPL2=N
DO 45 KK=1,M
IF(KK.EQ.M) NCPL2=N2
IF(KK.EQ.1) GO TO 9
DO 5 J=1,NCPL
JM=J-1
X(J,KK)=X(N-NCPL+J,KK-1)
DO 2 I=1,NCPL
2 A(I,J,KK)=A(N-NCPL+I,N-NCPL+J,KK-1)
DO 5 I=NCPL,NCPL2
D-23
Z=A(I,J,KK)
IF(J.EQ.1) GO TO 5
DO 4 K=1,JM1
4  Z=Z-A(I,K,KK)*A(J,K,KK)
5  A(I,J,KK)=Z*A(J,J,KK)
9  DO 40 J=NCPL,NCP2
   JP1=J+1
   JM1=J-1
   Z= A(J,J,KK)
   W=B(J,KK)
   IF(J.EQ.NCP3)GO TO 20
   DO 10 K=NCP3,JM1
   W=W-A(J,K,KK)*X(K,KK)
   Z=Z-A(J,J,KK) ** 2
10  IF(Z.LT.0.) RETURN
20  A(J,J,KK)= DSQRT(1./Z)
   X(J,KK)=W*A(J,J,KK)
   IF(J.EQ.NCP2) GO TO 43
   DO 40 I=JP1,NCP2
   Z= A(I,J,KK)
   IF(J.EQ.NCP3)GO TO 40
   DO 30 K=NCP3,JM1
30  Z=Z-A(I,J,KK)*A(J,K,KK)
40  A(I,J,KK)=Z*A(J,J,KK)
43  NCP3=1
45  NCPL=NCPL+1
   NCPL=N+1-N2
   NCP3=NCPL
   NCP2=N
   DO 92 KK=1,M
   KKK=M+1-KK
   IF(KK.EQ.M) NCP2=N+1-N1
   IF(KK.EQ.1)GO TO 89
   DO 85 J=1,NCPL
85  X(N-NCPL+J,KK)=X(J,KK+1)
89  DO 90 J=NCPL,NCP2
   L=N+1-J
   JM1=J-1
   Z= X(L,KK)
   IF(J.EQ.NCP3) GO TO 90
   DO 80 I=NCP3,JM1
   K=N+1-I
80  Z=Z-A(K,L,KKK)*X(K,KKK)
90  X(L,KKK)= Z*A(L,L,KKK)
NCP3=1
92  NCPL=NCPL+1
IND=2
RETURN
END

SUBROUTINE EPHREM(D,N,V)
DIMENSION V(6),E(6,10)
DATA E / .387099, .205627, .122427, .8352647, 1.34099, 3.885481, 7.23332

D-24
DOUBLE PRECISION FUNCTION POWER(R)
DOUBLE PRECISION R, DSQRT, A1, A2
DATA A1, A2 / 2.8250 50, 1.182550 /
IF(R GT 5.0) GO TO 1
POWER = (A1 / (A2 / (1.0 * A1)) ** 2) GO TO 1
RETURN
POWER = (A1 - A2 / DSQRT(R)) / R ** 2
RETURN
END

DOUBLE PRECISION FUNCTION DPOWER(R)
DOUBLE PRECISION R, DSQRT, A1, A2
DATA A1, A2 / -5.650 0, 4.562500 /
DPOWER = (A1 + A2 / DSQRT(R)) / R ** 3
IF(DPOWER LT 0.) DPOWER = 0.
RETURN
END

SUBROUTINE TCTPRE(TW, SI)
COMMON/BDYP/BDY(3,3), PV, PC
COMMON/COUNT, NCOUNT, NLP
COMMON/TIMEQ/ TT

D-25
DOUBLE PRECISION TT
DIMENSION ST(30)
COMMON/PXXX/PX/PR(301),TYYM(301),VAAA(301)
10 IF(NCOUNT.EQ.0) RETURN
11 IF(NCOUNT.EQ.NCOUNT) GO TO 950
WRITE(6,4321)
4321 FORMAT(1H1)
NCOUNT=NCOUNT
NNN=101
DT=TT/(NNN-1)
DO 1 I=1,0NNN
1 PR(I)=PMap((I-1)*DT)
CALL RESCAL (PR,TYYM,NNN,TAVSTR)
DO 4 I=1,NNN
4 VA(I)=AMAP(TYYM(I)*TT)/PMap(TYYM(I)*TT)
T=TT*TAVSTR
950 CONTINUE
B0=TT*31557600./SI
C=SI*65280./31557600.
BOT=B0*T
CT=C/T
CALL ROOTK(VA,NNN,CT,BOT,ST,LSGF)
8 IF(LGF) 2,2,3
3 TB=ST(I)*T
DO 199 I=1,30
U=TT*365.25*ST(I)
K=I-2*(1/2)
IF(I.GT.1.AND.ST(I).LT.000001) ST(I)=1.
IF(I.GT.1.AND.K.EQ.1) TB=TB+(ST(I)-ST(I-1))*T
IF(ST(I).GT.999999) GO TO 200
199 CONTINUE
200 CONTINUE
PC=(TB*(B0*C)**2)/(1.-B0*TB)
PS=.7121*PC
CALL INVINT(ST,TYYM,NNN)
T=TT
DO 201 I=1,30
201 ST(I)=T*365.25*ST(I)
CALL OUTTGG(TW,SI,PS,ST)
RETURN
2 CONTINUE
50 WRITE(6,1000)
1000 FORMAT(1X//44XMISSION IMPOSSIBLE EVEN WITH CONTINUOUS THRUST)
ST(1)=TT
ST(2)=0.
CALL OUTTGG(TW,SI,PS,ST)
RETURN
END

SUBROUTINE INVINT(ST,TAU,N)
DIMENSION ST(10),TAU(N)
DO 1 I=1,30
IF(ST(I).GE.1.) GO TO 1

D-26
TS = ST(I)*(N-1)
NTS = TS
TS = TS - NTS
ST(I) = TAU(NTS+1) + TS*(TAU(NTS+2) - TAU(NTS+1))
1 CONTINUE
RETURN
END

SUBROUTINE RESCAL(P, TAU, N, SCALE)
DIMENSION P(N), TAU(N)
COMMON/PXXPX/PR(301), TYME(301), VA(301), T(301), TAUS(301)
DOUBLE PRECISION SUM
SUM = 0.
T(1) = 0.
TAUS(1) = 0.
TAU(1) = 0.
DU 1 = 2, N+2
T(I) = FLOAT(I-1)/(N-1)
T(I+1) = FLOAT(I)/(N-1)
TAUS(I) = SUM + (5.*P(I-1)+8.*P(I)-P(I+1))/4.
SUM = SUM + P(I-1)+4.*P(I)+P(I+1)
TAUS(I+1) = SUM
1 CONTINUE
DO 4 I = 2, N
S = (I-1)*SUM/(N-1)
DU 2 = 2, N
L = J+K-1
IF(L.EQ.N) GO TO 3
IF(S.LE.TAUS(L)) GO TO 3
2 CONTINUE
J = L
4 TAU(I) = T(J-1) + T(S-TAUS(J-1))/TAUS(J-1)**(T(J)-T(J-1))
SCALE = SUM/(3*N-3)
RETURN
END

SUBROUTINE ROOTRIA(V, NAV, C, BZ, TSW, LGF)
DIMENSION AV(NAV), TSW(30)
DATA NSTEP/14/
REAL JV, KL, KH, K, JVTEST
LGF = 1
AVMAX = AV(1)
JV = AV(1)**2
DO 1 I = 2, NAV
IF(AVMAX.LT.AV(I)) AVMAX = AV(I)
IF(AVMAX.LT.AV(I+1)) AVMAX = AV(I+1)
1 JV = JV + 4.*AV(I)**2 + 2.*AV(I+1)**2
JV = (JV - AV(NAV)**2)/FLOAT(NAV*3-3)
IF(1.-BZ) 20, 20, 9
JVTEST = BZ*C*AV(I)
DU 10 I = 2, NAV
10 RETURN
```fortran
10    JVTEST=JVTEST+4.*(BZ*C*AV(I)/(1.-BZ*FLOAT(I-1)/FLOAT(NAV-1)))
     C+2.*(BZ*C*AV(I+1)/(1.-BZ*FLOAT(I)/FLOAT(NAV-1)))
     JVTEST=(JVTEST-BZ*C*AV(NAV)/(1.-BZ))/FLOAT(NAV*3-3)
    IF(JV-JVTEST) 20,20,50
20    KH=AVMAX
     KL=KH
30    KL=KL-.5*KH
     JVTEST=JV
     CALL KROOT(AV,NAV,C,BZ,KL,JVTEST,TSW)
    IF(JV-JVTEST) 40,30,30
40    DO 4 I=1,NSTEP
     K=0.5*(KH+KL)
     JVTEST=JV
     CALL KROOT(AV,NAV,C,BZ,K,JVTEST,TSW)
    IF(JV-JVTEST) 2,5,3
2    KL=K
    GO TO 4
3    KH=K
4    CONTINUE
5    CONTINUE
50    CONTINUE
    RETURN
    LGF=-1
    RETURN
END

SUBROUTINE OUTTOOT(TW,SI,JC,T)
DIMENSION A(4)
DATA A/4H**** , 4H**** , 4H**** , 4H**** /
DIMENSION T(30)
DOUBLE PRECISION THRUST(10),OFF(10),AT(10),TIME(10)
DATA THRUST/10*8H THRUST /
DATA AT/10*8H AT /
DATA OFF/ 8H OFF , 8H ON , 8H OFF , 8H ON , 8H ON , 8H ON , 8H ON ,
     8H ON ,
DATA TIME/10*8H TIME /
WRITE(6,200)
WRITE(6,201) TW
WRITE(6,202) SI
WRITE(6,203) JC
J=1
WRITE(6,5432)
DO 1 I=2,30
    IF(T(I)) 2,2,1
1    J=1
2    CONTINUE
    J1=(J-1)/10+1
    J2=J
    J=(J2+J1-1)/J1
    DO 3 KKK=1,J1
    J=(J+1)/2
    J= 2*J
```

D-28
SUBROUTINE KRUOT(AV, NAV, C, BZ, K, JI, TSW)
DIMENSION AV(NAV), TSW(30)
REAL JI
REAL K, JV, ML, KL, IGL, MN, KN, IGN
INTEGER T, ST
DO 1 I = 1, 30
1 TSW(I) = 0.
DT = 1. / FLOAT(NAV - 1)
ST = 0.
B = BZ
JV = 0.
XJI = JI * 1.05
JI = 0.
ML = 1.
KL = AV(I) - K
IGL = C * B * AV(I)
IF(KL .GE. 0.) GO TO 21
ST = 1.
B = 0.
21 CONTINUE
DO 13 T = 2, NAV
IF(ST .GT. 29) GO TO 12
IF(B .EQ. 0.) GO TO 2
MN = ML - B * DT
IGN = C * B * AV(T) / MN**2
DJ = 0.5 * (IGL + IGN) * DT
KN = AV(T) / MN - (JV + DJ) / C

END
IF(KN) 8, 5, 7
5 B=0.
   JV=JV+DJ
   JI=JI+0.5*(IGL*ML+IGN*MN)*DT
   GO TO 6
7 CONTINUE
   JI=JI+0.5*(IGL*ML+IGN*MN)*DT
   JV=JV+DJ
   GO TO 4
8 ST=ST+1
   CALL INTRDN(AV(T-1), BZ, C, K, FLOAT(T-2)*DT, DT, ML, KL, IGL, JV, TSW(ST))
1, JI)
   B=0.
   GO TO 11
2 CONTINUE
   KN=AV(T)/ML-K-JV/C
   IF(KN) 14, 3, 9
3 CONTINUE
   MN=ML
   B=BZ
   IGN=C*B*AV(T)/ML**2
6 ST=ST+1
   TSW(ST)=FLOAT(T-1)/FLOAT(NAV-1)
   GO TO 4
9 ST=ST+1
   CALL INTRUP(AV(T-1), BZ, C, K, FLOAT(T-2)*DT, DT, ML, KL, IGL, JV, TSW(ST))
1, JI)
   B=BZ
   GO TO 11
4 CONTINUE
   ML=MN
   IGL=IGN
14 CONTINUE
   KL=KN
   GO TO 11
11 CONTINUE
   IF(ML.LT.0) JI=1.0E40
   IF(XJI.LT.JI) RETURN
13 CONTINUE
12 CONTINUE
RETURN
END

SUBROUTINE INTRDN(AV, B, C, K, T, DT, ML, KL, IGL, JV, TSW, JI)
DIMENSION AV(2)
REAL KN, KN, IGN
REAL ML, KL, IGL, K
REAL JI, JV
MN=ML-B*DT
IGN=C*B*AV(2)/MN**2
KN=AV(2)/MN-.5*(IGL+IGN)*DT+JV/C-K
DTS= KL*DT/|KL-KN|
TSW=T+DTS
D-30
MN=ML-B*DTS
AV2=(AV(1)*(DT-DTS)+AV(2)*DTS)/DT
IGN=C*B*AV/ML**2
DJ=0.5*(IGL+IGN)*DTS
JV=JV+DJ
JI=JI+0.5*(IGL*ML+IGN*MN)*DTS
KL=AV/ML-JV/C-K
IGL=C*B*AV(2)/ML**2
ML=MN
IF(KL.GT.0.) KL=0.
RETURN
END

SUBROUTINE INTRUP(AV,B,C,K,DT,ML,IG,JP,TSW,J1)
DIMENSION AV(2)
REAL MN,KN,IGN
REAL ML,KL,IGL,K
REAL JI, JV
KN=AV(2)/ML-JV/C-K
DTS= KL*DT/(KL-KN)
AV1=(AV(1)*(DT-DTS)+AV(2)*DTS)/DT
TSW=T+DTS
DTS=CT-DTS
MN=ML-B*DTS
IGN=C*B*AV(2)/ML**2
IGL=C*B*AV(2)/MN**2
JV=JV+0.5*(IGN+IGL)*DTS
JI=JI+0.5*(IGN*ML+IGL*MN)*DTS
ML=MN
KL=AV(2)/ML-JV/C-K
IF(KL.LT.0.) KL=0.
RETURN
END

REAL FUNCTION PMAP(H)
DOUBLE PRECISION XV,AV,WV,TT,T,T2,T3,TP,P12,S
COMMON/PARAM/ML,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NO
COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),P12
COMMON/DOUT/XV(30,6),AV(30,6),WV(10,6)
S=H
CALL MODLEG(T,NL,S,SN)
CALL POLEV(WV(1,LN),G,S,1,ND,NO)
PMAP=1./G**2
RETURN
END

REAL FUNCTION AMAP(H)
DOUBLE PRECISION XV,AV,WV,TT,T,T2,T3,TP,P12,S
COMMON/DOUT/XV(30,6),AV(30,6),WV(10,6)
COMMON/PARAM/ML,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NO
COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),P12

D-31
DIMENSION G(3)
S=H
CALL MODLEG(T,NL,S,LN)
CALL POLEVL(AV(1,LN),G,S,ND,NP,N)
Z=0.
DO 15 M=1,ND
15 Z=Z+G(M)**2
AMAP=SQRT(Z)
RETURN
END