2

CR 73359

AVAILABLE TO THE PUBLIC

BOEING

SEATTLE, WASHINGTON

N69-34537	
(ACCESSION NUMBER)	(THRU)
96	1
(PAGES)	(CODE)
NASA-CR-#73359	30
(NASA CR OR TMX OR AD NUMBER)	(CATEGORY)

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

PRECEDING PAGE BLANK NOT FILMED.

TABLE OF CONTENTS

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

$$\ddot{x} + \frac{kx}{r^3} = a \qquad\qquad 0 \le t \le T$$

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

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

Here a_o is the initial acceleration of the vehicle at 1 au, c the exhaust velocity, μ the relative mass of the vehicle, and

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

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,

$$J = \sum_{i=1}^{nl} 2\tau_i^{-3} P_i$$

Here τ_i is the elapsed time for the ith leg and P_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 Px, Pxx 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 $Q^\nu = [p_o/p\ (r(s^\nu))]^{1/2}$. Let AV(m), m = 1,..,nd 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 $G(m)$, $m = 1,..,nd$ be a set of column vectors defined by

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

i.e.

$$G(m)^\nu = Q^\nu \; AV(m)^\nu$$

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

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

$$Pxx(i,j) = \sum_{m=1}^{nd} Gx(m,i)^T \, FGx(m,j) + R(m)^T Gxx(m,i,j)$$

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

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

$$\begin{aligned} Pxx\,(i,j) = {} & (A^{-1}BA)^T \; Q^T FQ(A^{-1}BA) \; \delta_{i,j} \\ & + (A^{-1}BA)^T \; Q^T FQH(i,j) \\ & + H(j,i)^T \; Q^T FQ \; (A^{-1}BA) \\ & + \sum_{m=1}^{nd} H(m,i) \; Q^T FQH(m,j) \\ & + R(m)^T \; Gxx(m,i,j) \end{aligned}$$

The last term is rather complicated. For Gauss' method it is neglected. For Newton's method we set Gxx(m,i,j) = QYxx(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

$$\Delta J = \int_{t_o}^{t_1} |a_c|^2 \frac{p_o}{p(t)} \, dt$$

$$0 = \int_{t_o}^{t_1} |a_c| \, |a_v| \frac{p_o}{p(t)} \, dt - Jv$$

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 $\tau(t) = \int_{t_o}^{t} \frac{p(t)}{p_o} \, dt$.

The above equations now become

$$\Delta J = \int_{\tau_o}^{\tau_1} |h_c|^2 \, d\tau$$

$$0 = \int_{\tau_o}^{\tau_1} |h_c| \; |h_c| d\tau - J_v$$

$$|h_c| = \frac{a_o}{\mu(\tau)} \sigma(\tau)$$

$$\frac{d\mu}{d\tau} = - \frac{a_o}{c} \sigma(\tau)$$

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^o, 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 analagous number of the arrival planet	
NB1	indicates the initial constraint on excess velocity	0 for flyby 1 for fixed excess velocity 2 for rendezvous
NB2	indicates the final constraint on excess velocity	
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	0 for constant 1 for variable
NA	is a control on the accuracy of the acceleration time history	0 for approximate solution 1 for more accurate acceleration time history
NT	is a control which allows tracking from a previously computed trajectory	0 no tracking, uses standard starting solution 1 tracking

Constant Thrust Input

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

```
CALL TCTPRE(TW,SI)
```

The arguments of the subroutine are as follows:

TW is the initial* thrust to weight ratio (dimensionless)

* In the case of variable power, TW is relative to thrust level at 1 a.u.

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 x 3 x 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 $\int_0^T |a|^2 \frac{P_o}{P} dt$ 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.

```
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 ephermerides, 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° 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.

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 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.

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 VTMODE. Samples of both forms are given in Appendix C.

The long form is obtained by calling subroutine OUTCAL after each call of 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 m^2/sec^3
EXCESS VELOCITY AT <u>DEPARTURE</u> in km/sec	
EXCESS VELOCITY AT <u>ARRIVAL</u> 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 m^2/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**(-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

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

* 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.

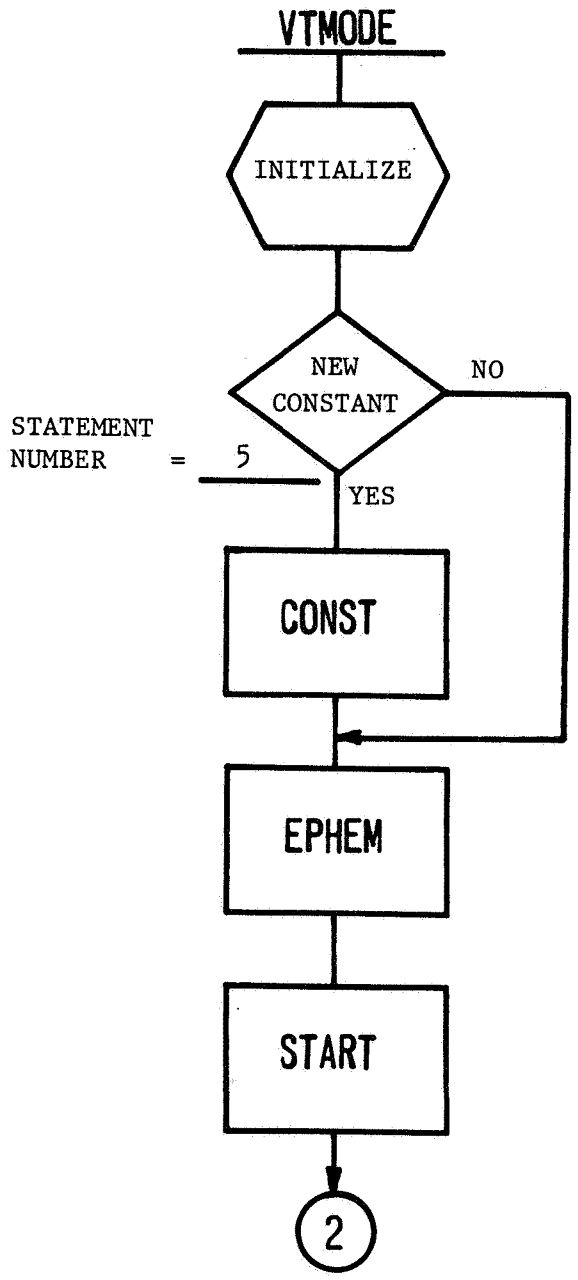
Section 1: This section has two functions. The first is to initialize control parameters and other quantities used in the succeeding steps. The second is to call subroutines which calculate preliminary data. CONST is called to calculate matrices of constants filling the common blocks CONS1, CONS2. EPHEM 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,

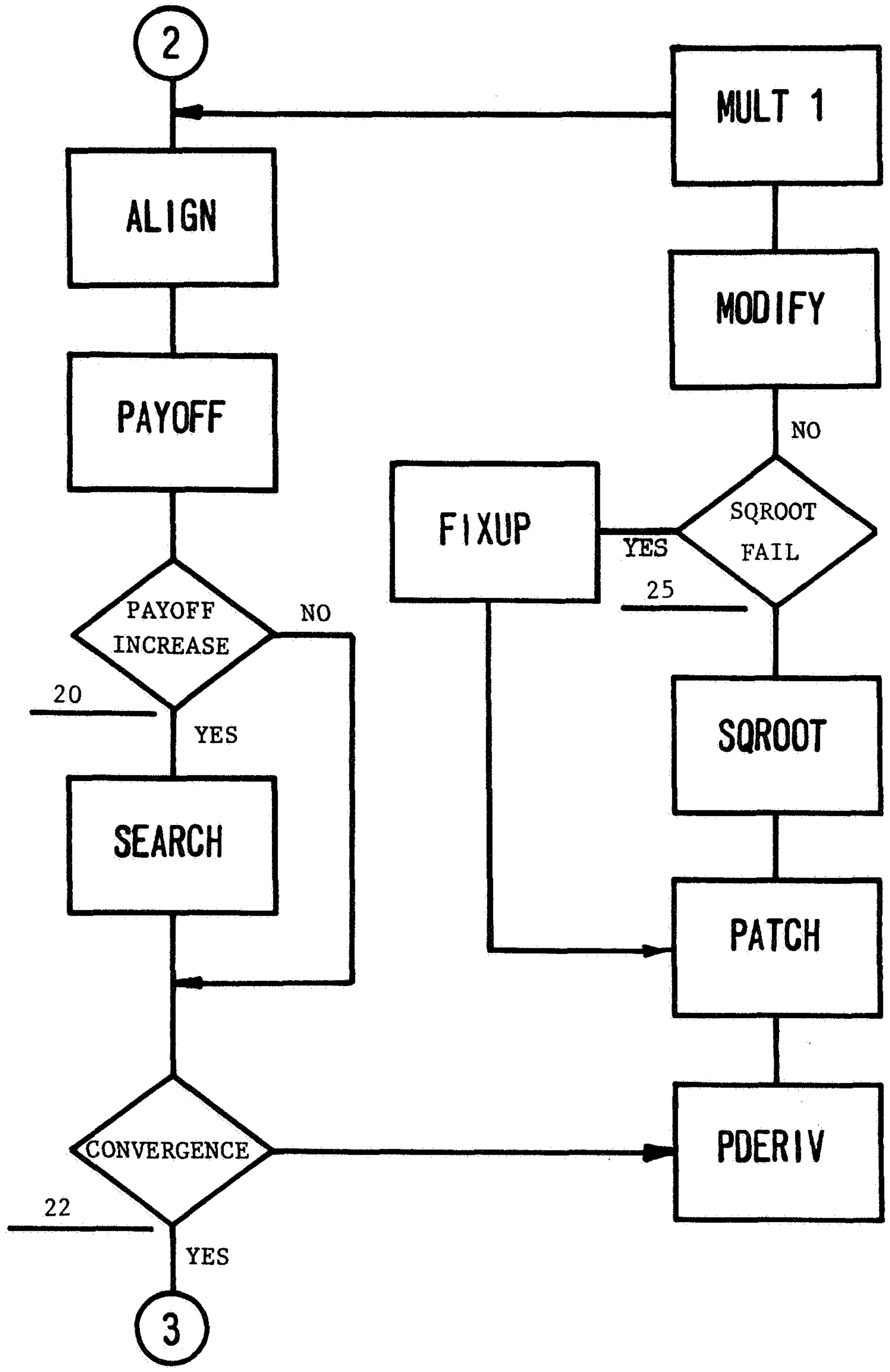
SUBROUTINE VTMODE

SECTION I



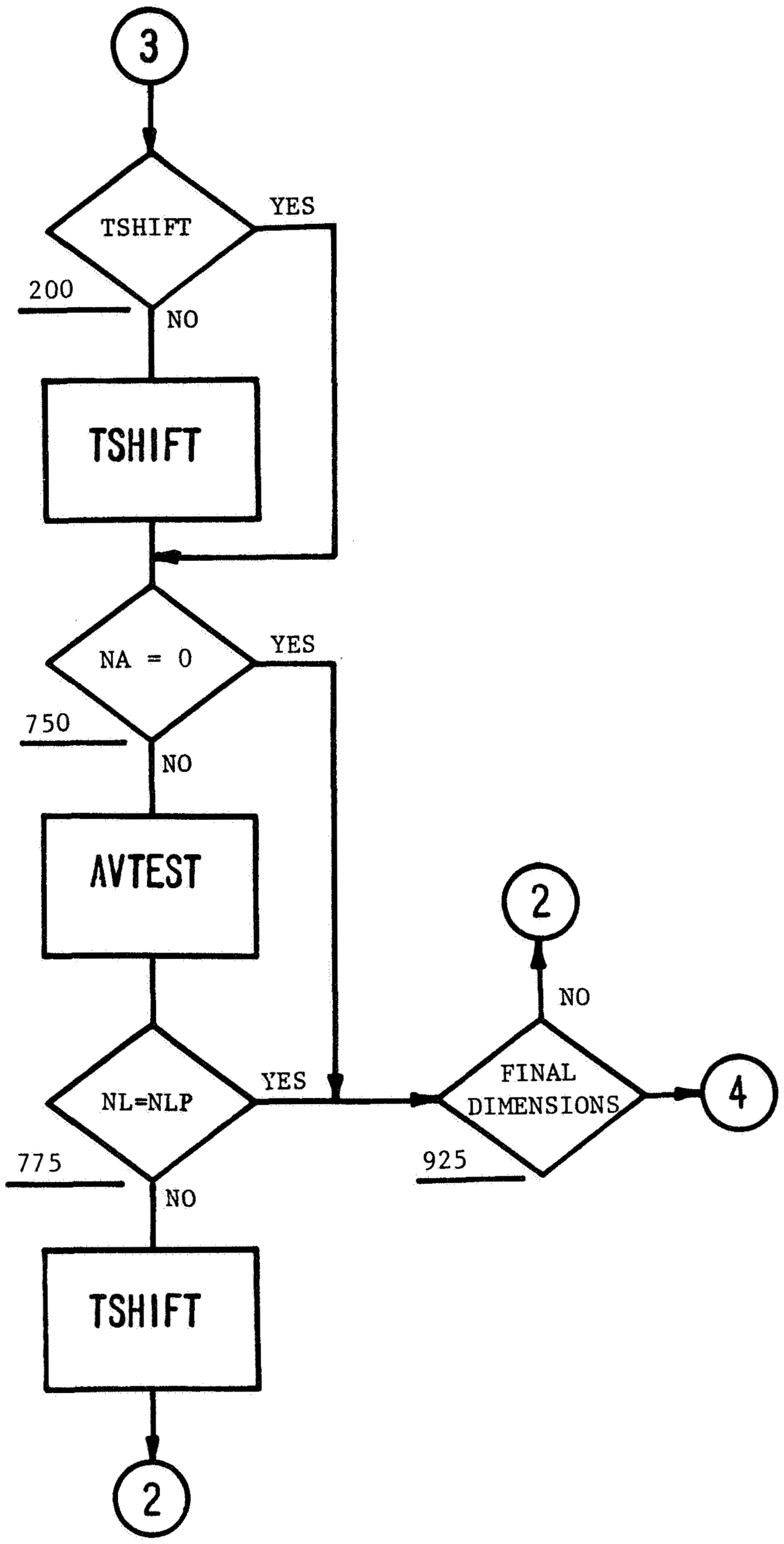
SUBROUTINE VTMODE (CONT.)

SECTION II



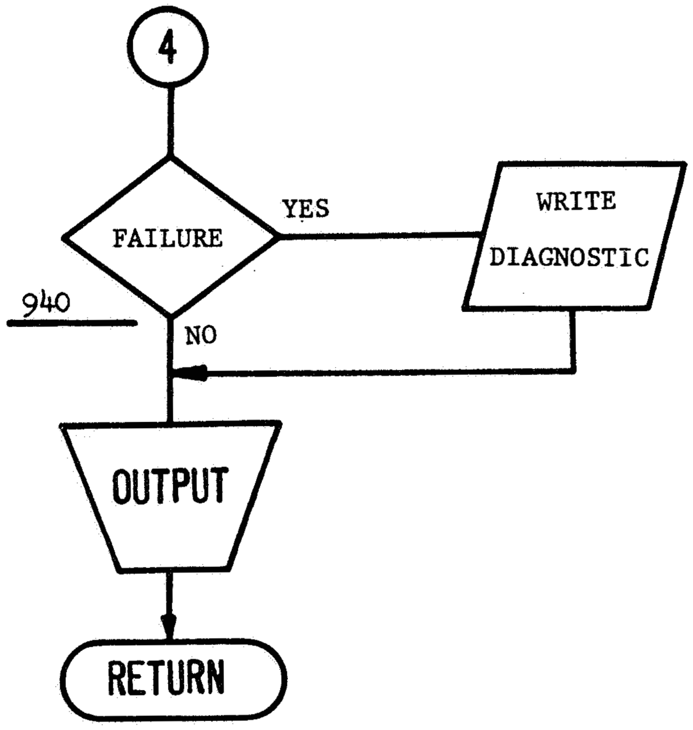
SUBROUTINE VTMODE (CONT.)

SECTION III



SUBROUTINE VTMODE (CONT.)

SECTION IV



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.

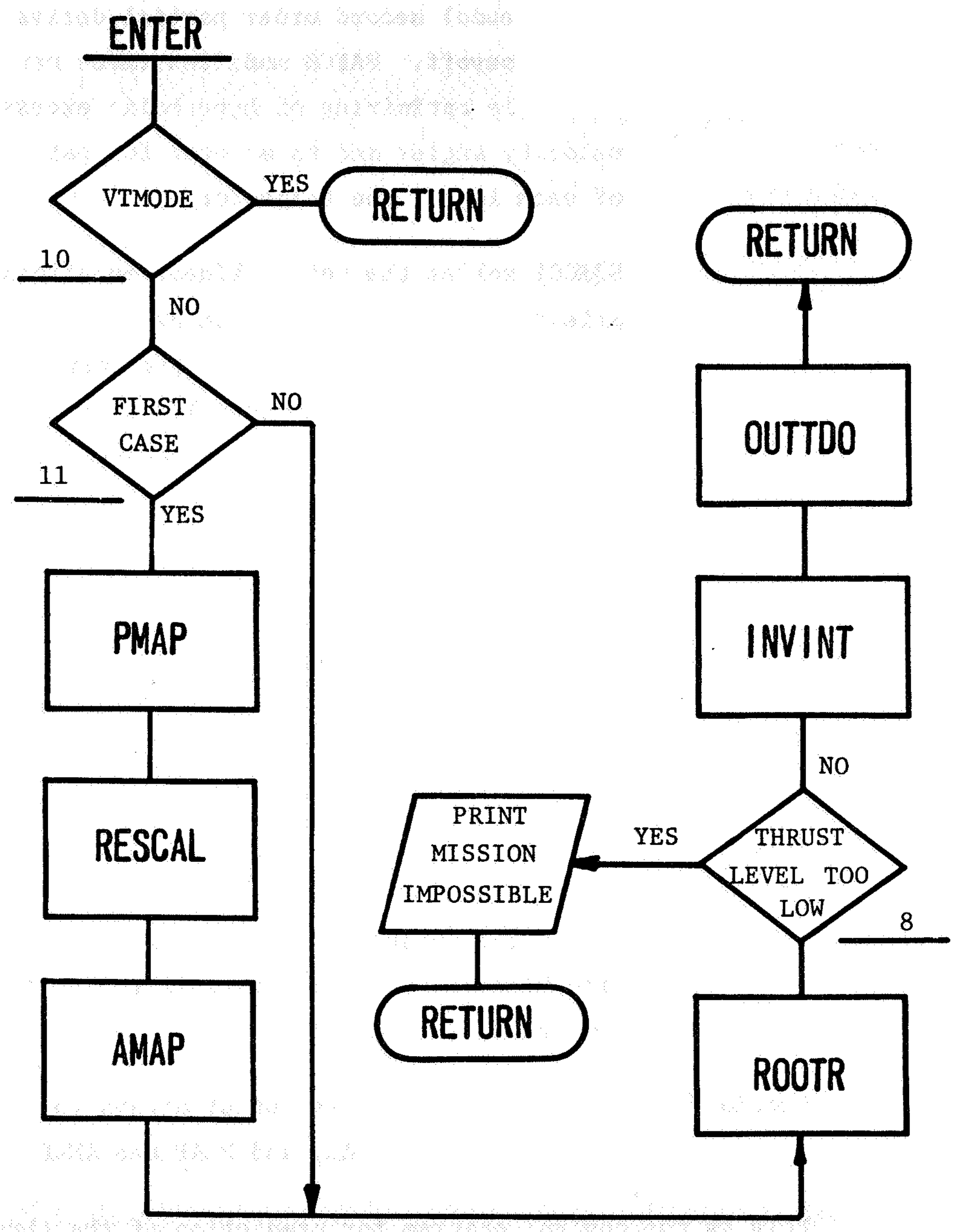
Section 3: This section performs two tasks. It shifts the trajectory patch times to achieve a better payoff and smoother acceleration magnitude time history. It converts from two to three dimensions (if NC = 3).

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.

SUBROUTINE TCTPRE



SI	is the specific impulse of the engine in seconds.

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

PMAP	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 PMAP 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 $\dot{x}$, $\dot{y}$, $\dot{z}$. 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

		semi-major axis (a.u.)	eccen-tricity	incli-nation (degrees)	longitude of node (degrees)	longitude of perihellion (degrees)	mean longitude of epoch (degrees)
1.	MERCURY	.387099	.205627	7.00399	47.85714	76.83309	222.6217
2.	VENUS	.723332	.006793	3.39423	76.31972	131.0083	174.2943
3.	EARTH	1.000000	.016726	0.	0.	102.2525	100.1581
4.	MARS	1.523691	.093368	1.84991	49.24903	335.3227	258.7673
5.	JUPITER	5.202803	.048435	1.30536	100.0444	13.67823	259.8311
6.	SATURN	9.538843	.055682	2.48991	113.3075	92.26447	280.6713
7.	URANUS	19.18195	.047209	7.73058	73.79630	170.0108	141.3050
8.	NEPTUNE	30.05779	.008575	1.77375	131.3398	44.27395	216.9409
9.	PLUTO	39.43871	.250236	17.1699	109.8856	224.1602	181.6463
10.		1.0	0.	45	270	0.	100.1581

Epoch = 2436935 (1 Jan. 1960)

Reference: Planetary Flight Handbook, NASA SP-35, Part 1, Vol. 3 (1963)

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 interplated. 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

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

$$m(t) = 1 - \beta_o \int_o^t \sigma(s)\, ds$$

$$K(t) = \frac{a_v(t)}{m(t)} - \frac{1}{c} \int_o^t \frac{c\beta_o \sigma(s)}{m(s)^2} a_v(s)\, ds - K$$

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

$\sigma(t)$ were constant. When K(t) is discovered to have changed sign, subroutine INTRDN or INTRUP is called to interpolate for the switch point and adjust accordingly the value of K(t).

In addition the program evaluates the auxilliary function

$$J_i = \int_o^t a_v(t) \frac{c\beta_o \sigma(t)}{m(t)} \, dt$$

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,..., NAV
NAV	is the number of elements in the array AV.
C,BZ, K	are the constants c, β_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

$$Y(m) = \sum_{k=1}^{2} U(k)^T A(k) U(k) X(m) \qquad m = 1,..,nd$$

if $L = 1$, or

$$Y(m) = \sum_{k=1}^{2} U(k)^T A(k)^T U(k) X(m) \qquad m = 1,..,nd$$

if $L = 2$

(See Reference 1 for interpretation).

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

$$C(k) = A(k)^T B(k) \qquad k = 1,2$$

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

$$B = \sum_{k=1}^{2} U(k)^T ((-1)^k \; A1(k), A2(k) \; \tilde{I})$$

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

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

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

If NS is 2, C is assumed to be symmetric and some savings reult.

OUTCAL: OUTCAL computes quantities related to the trajectory at 26 equally spaced times and stores them in the matrix 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

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

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

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., POWER (1) = 1. Note that all quantities in this subroutine are double precision.

The nominal power profile is as follows:

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

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

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

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 T2, T3, and TP are also evaluated anew.

RESCAL: This subprogram solves the differential equation:

$$\frac{d\tau}{dt} = p(t)$$

tabulating the solution:

$$t_i = g(\tau_i)$$

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 $g(\tau_i)$ tabulated at equal intervals in τ.

N is the number of points in the array P and G.

S is $\int_o^1$ pdt assuming $t_o = o$, $t_N = 1$

Simpson's Rule is applied to tabulated τ 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 $J_v - J_i = 0$ where:

$J_v = \int_o^T |a_v|^2 \, dt$ defined by VTMODE

$J_i = \int_o^T |a_v| |a_c| dt$ defined by KROOT since J_i is a montone 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)

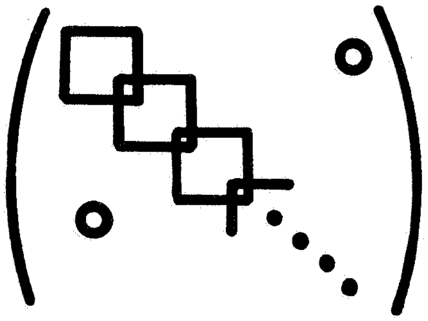
AV	is the array $a_v(t_i)$ $i = 1,2,...,NAV$
NAV	is the number of elements in the array AV
C, BZ	are constants passed on to KROOT
TSW	is the output array of length 30, tabulating each time that the constant thrust switches value.
LGF	is the error indicator signifying when a mission is impossible even 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 $\varepsilon(1,0)$,P(X+RODX) must have a minimum.

SECANT actually provdes 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:



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: $f(0) = 0$, $f(1) = 1$, $f'(0) = DZ$, $f'(1) = D1$. In addition $f'(s) > 0$, if $DZ > 0$ and $D1 > 0$. 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,D1,S) where:

DZ	is f'(o)
D1	is f'(1)
S	is any number in the interval [0,1]

The procedure is as follows:

if $DZ + D1 > 2$

$$f(S) = [(\frac{D1-DZ}{2})\ S^2 + (DZ)S]\ B + g(S) + h(S)$$

where

$$B = (\frac{2}{DZ+D1})/(1 + \sqrt{1 - \frac{2}{DZ+D1}}\ )$$

$$g(S) = \begin{cases} 0 & S \leq 1-B \\ \frac{D1}{2}\ (\frac{1-B}{B})\ (S+B-1)^2 & S > 1-B \end{cases}$$

$$h(S) = \begin{cases} \frac{DZ}{2}\ (1-B)S(2-\frac{S}{B}) & S < B \\ \frac{DZ}{2}\ (1-B)\quad B & S \geq B \end{cases}$$

and if $DZ + D1 \leq 2$

$$f(S) = S(DZ + S(3 - 2DZ - D1 + S(DZ + D1-2)))$$

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,..,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 vector (AV) and the power vector (WV), and thirdly it calculates quantities which are used in DERIV (B,Q,G,H).

The quantity G is calculated only if 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 adviseable 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

1. Johnson, F. T., "Approximate Finite-Thrust Trajectory Optimization," AAS Paper No. 68-080, AAS/AIAA Astrodynamics Specialists Conference, Jackson, Wyoming, September 3-5, 1968.

2. Johnson, F. T., and Hahn, D. W., "An Approximate Technique for Predicting Thrust-Limited Performance," Technical Papers on Mission Analysis Technology for Electric Propulsion, AIAA 7th Electric Propulsion Conference, Williamsburg, Virginia, March 3-5, 1969.

3. Strack, W. C., "Solar Electric Propulsion System Performance for a Close Solar Probe Mission," AIAA J. Spacecraft, Vol. 4, No. 4, April 1967.

APPENDIX A

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 au^2/yr^3 (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,..,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 α (and β if ND=3) are introduced, representing the heliocentric longitude (latitude) of the hyperbolic excess velocity. Now X(2,M,1) becomes a function of α (and β) (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 α (and β) 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 $\begin{cases} \text{2 for Gauss' method} \\ \text{3 for Newton's method} \end{cases}$
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
NPM1:	NP - 1
NPO:	Same as NPOW
NPOW:	Power number $\begin{cases} \text{1 for variable power} \\ \text{0 for constant power} \end{cases}$
NR:	Number of full revolutions around the sun (NR = 0 for travel angles less than 360^o).
NT:	Tracking number $\begin{cases} \text{0 for no tracking} \\ \text{1 for tracking} \end{cases}$
P:	Payoff in au^2/yr^3
PLANET:	A logical variable which is true if one of the planets is Mercury, Pluto or ficticious.
PS:	Payoff in au^2/yr^3.

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 3π.
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,..,ND; I = 1,..,ND-1; NB = 1,2) representing the first order partial derivatives of VR with respect to heliocentric longitude and/or latitude, that is, if

$$VR(\cdot,NB) = \begin{pmatrix} \cos\alpha \cos\beta \\ \sin\alpha \cos\beta \\ \sin\beta \end{pmatrix}, \qquad \text{then}$$

$$VRA(M,1) = \frac{\partial\, VR(M,NB}{\partial \alpha} \text{ and } VRA(M,2) = \frac{\partial\, VR(M,NB)}{\partial \beta}$$

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

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

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

WV: WV is a doubly subscripted array (WV(I,LN)) and is equal to $\sqrt{p_o/p}$ (p=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

$$X(2,M,LN) = T(LN) \left.\frac{dx(M,t)}{dt}\right|^{t=t_i}$$

$$X(NP-1,M,LN) = -T(LN) \left.\frac{dx(M,t)}{dt}\right|^{t=t_f}$$

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 IO 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 velocity 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 compiler 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.

```
*****************************************************************************************************

*****************************************************************************************************
    VARIABLE THRUST CONSTANT POWER TRAJECTORY IN 2 DIMENSIONS USING 1 LEGS OF 10 CHEBYCHEV POINTS
    DEPARTURE PLANET IS EARTH    AT JULIAN DAY 2440538.0  ARRIVAL PLANET IS MARS     AT JULIAN DAY 2446738.0
    EXCESS VELOCITY AT EARTH   IS        0.0              CONSTRAINED  , AT MARS    IS         0.0               CONSTRAINED
    PERFORMANCE INDEX IS          0.531113E 01    CONSTANT POWER
*****************************************************************************************************

*****************************************************************************************************
```

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 for NA = 0

JC = 12.7077 for NA = 1

The main program is as follows:

```
COMMON/BDYP/BDY(3,3,2),PV,PC

CALL VTMODE(3,0,3,5,2,0,2444140.,2444940.,0.,0.,1,1,0)

CALL OUTCAL

CALL TCTPRE(.00008,5000.)

STOP

END
```

VARIABLE THRUST VARIABLE POWER TRAJECTORY

3 DIMENSIONAL TRAJECTORY USING 3 PATCHED POLYNOMIALS WITH 10 CHEBYCHEV POINTS

PERFORMANCE INDEX (JV) 9.592

DEPARTURE PLANET EARTH ARRIVAL PLANET JUPITER

DEPARTURE DAY 2444140. ARRIVAL DAY 2444940.

EXCESS VELOCITY AT EARTH 0.0 CONSTRAINED

EXCESS VELOCITY AT JUPITER 7.633 UNCONSTRAINED

TIME	X	Y	Z	R	THETA	PHI	AX	AY	AZ	MAG. A	P/PO
0.0	1.003	-0.005	-0.000	1.003	-0.28	-0.00	0.56	7.84	-0.12	7.86	0.996
32.00	0.848	0.546	-0.000	1.009	32.78	-0.02	-2.47	6.83	0.01	7.27	0.991
64.00	0.433	0.995	-0.000	1.085	66.46	-0.02	-3.60	4.23	0.13	5.55	0.911
96.00	-0.113	1.251	0.000	1.256	95.18	0.02	-2.93	2.08	0.18	3.60	0.758
128.00	-0.672	1.335	0.003	1.494	116.72	0.10	-1.93	1.02	0.18	2.19	0.597
160.00	-1.187	1.393	0.006	1.763	132.33	0.20	-1.23	0.57	0.16	1.37	0.467
192.00	-1.648	1.203	0.010	2.040	143.87	0.29	-0.83	0.36	0.13	0.91	0.372
224.00	-2.056	1.061	0.015	2.314	152.70	0.38	-0.57	0.26	0.11	0.64	0.304
256.00	-2.418	0.895	0.021	2.576	159.68	0.46	-0.38	0.23	0.09	0.46	0.254
288.00	-2.740	0.715	0.026	2.832	165.38	0.54	-0.28	0.19	0.08	0.35	0.217
320.00	-3.028	0.526	0.032	3.073	170.15	0.61	-0.23	0.15	0.07	0.28	0.189
352.00	-3.286	0.333	0.039	3.303	174.22	0.67	-0.19	0.12	0.06	0.23	0.167
384.00	-3.517	0.138	0.045	3.520	177.76	0.73	-0.16	0.10	0.05	0.19	0.149
416.00	-3.725	-0.058	0.051	3.726	180.89	0.79	-0.12	0.09	0.04	0.16	0.135
448.00	-3.913	-0.252	0.058	3.921	183.68	0.84	-0.09	0.08	0.03	0.13	0.124
480.00	-4.081	-0.444	0.064	4.106	186.21	0.89	-0.07	0.08	0.03	0.11	0.114
512.00	-4.232	-0.634	0.070	4.280	188.52	0.94	-0.06	0.06	0.02	0.09	0.106
544.00	-4.367	-0.821	0.076	4.444	190.65	0.99	-0.05	0.05	0.02	0.08	0.099
576.00	-4.487	-1.005	0.083	4.599	192.62	1.03	-0.05	0.04	0.02	0.06	0.093
608.00	-4.594	-1.185	0.089	4.745	194.47	1.07	-0.04	0.03	0.01	0.05	0.088
640.00	-4.688	-1.362	0.094	4.883	196.23	1.11	-0.03	0.02	0.01	0.04	0.084
672.00	-4.770	-1.535	0.100	5.012	197.84	1.14	-0.02	0.02	0.01	0.03	0.080
704.00	-4.841	-1.705	0.105	5.133	199.40	1.18	-0.01	0.02	0.01	0.02	0.077
736.00	-4.901	-1.870	0.111	5.247	200.88	1.21	-0.01	0.01	0.00	0.01	0.074
768.00	-4.951	-2.031	0.116	5.352	202.31	1.24	-0.01	-0.00	0.00	0.01	0.071
800.00	-4.991	-2.186	0.121	5.451	203.68	1.27	0.01	0.01	-0.00	0.02	0.069

```
PREDICTED CONSTANT THRUST TRAJECTORY

THRUST TO WEIGHT RATIO     0.000040000

SPECIFIC IMPULSE         5000.0   SEC.

PERFORMANCE INDEX (JC)       0.127077E 02


     THRUST       THRUST
      OFF           ON
       AT           AT
      TIME         TIME
    346.522      800.000

***●*** *************●*************************●*************************************************●************************
*********************************************************************************************************************************
```

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 3 DIMENSIONS USING 3 LEGS OF 10 CHEBYCHEV POINTS
    DEPARTURE PLANET IS EARTH    AT JULIAN DAY 2446750.0  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         0.598081E 01    CONSTANT POWER
*********************************************************************************

*********************************************************************************
*********************************************************************************

*********************************************************************************
    VARIABLE THRUST CONSTANT POWER TRAJECTORY IN 3 DIMENSIONS USING 3 LEGS OF 10 CHEBYCHEV POINTS
    DEPARTURE PLANET IS EARTH    AT JULIAN DAY 2446750.0  ARRIVAL PLANET IS SATURN   AT JULIAN DAY 2448550.0
    EXCESS VELOCITY AT EARTH   IS          0.999999E 00     CONSTRAINED  , AT SATURN  IS         0.812348E 01    UNCONSTRAINED
    PERFORMANCE INDEX IS         0.476252E 01    CONSTANT POWER
*********************************************************************************

*********************************************************************************
*********************************************************************************

*********************************************************************************
    VARIABLE THRUST CONSTANT POWER TRAJECTORY IN 3 DIMENSIONS USING 3 LEGS OF 10 CHEBYCHEV POINTS
    DEPARTURE PLANET IS EARTH    AT JULIAN DAY 2446750.0  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         0.371380E 01    CONSTANT POWER
*********************************************************************************

*********************************************************************************
*********************************************************************************

*********************************************************************************
    VARIABLE THRUST CONSTANT POWER TRAJECTORY IN 3 DIMENSIONS USING 3 LEGS OF 10 CHEBYCHEV POINTS
    DEPARTURE PLANET IS EARTH    AT JULIAN DAY 2446750.0  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         0.284414E 01    CONSTANT POWER
*********************************************************************************

*********************************************************************************
*********************************************************************************

*********************************************************************************
    VARIABLE THRUST CONSTANT POWER TRAJECTORY IN 3 DIMENSIONS USING 3 LEGS OF 10 CHEBYCHEV POINTS
    DEPARTURE PLANET IS EARTH    AT JULIAN DAY 2446750.0  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         0.215888E 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 inclinded 45^{o} to the ecliptic. A tenth set of orbital elements were inserted in Subroutine EPHEM. They are

$a = 1.000000$

$e = 0.$

$i = .8$ radians

$\Omega = 4.7$ radians

$\varpi = 0$ radians

$L = 1.748089$

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

```
COMMON/BDYP/BDY(3,3,2),PV,PC

CALL VTMODE(1,1,3,10,2,2,2443958.,2444678.,0.,0.,1,0,0)

CALL OUTCAL

STOP

END
```

VARIABLE THRUST VARIABLE POWER TRAJECTORY

3 DIMENSIONAL TRAJECTORY USING 4 PATCHED POLYNUMIALS WITH 10 CHEBYCHEV POINTS

PERFORMANCE INDEX (JV) 17.840

DEPARTURE PLANET EARTH ARRIVAL PLANET TENTH

DEPARTURE DAY 2443958. ARRIVAL DAY 2444678.

EXCESS VELOCITY AT EARTH 0.0 CONSTRAINED

EXCESS VELOCITY AT TENTH 0.0 CONSTRAINED

TIME	X	Y	Z	R	THETA	PHI	AX	AY	AZ	MAG. A	P/PO
0.0	-0.995	-0.071	0.000	0.997	-175.90	0.00	-0.27	-0.15	0.58	0.66	1.003
28.80	-0.850	-0.539	0.004	1.006	-147.61	0.24	-0.22	0.02	2.81	2.82	0.993
57.60	-0.505	-0.880	0.024	1.015	-119.86	1.36	-0.20	0.15	4.35	4.36	0.984
86.40	-0.045	-1.017	0.065	1.021	-92.53	3.63	-0.15	0.26	4.88	4.89	0.978
115.20	0.425	-0.923	0.120	1.023	-65.29	6.71	-0.05	0.33	4.32	4.34	0.976
144.00	0.738	-0.613	0.174	1.025	-37.75	9.75	0.08	0.30	2.81	2.83	0.974
172.80	0.994	-0.174	0.206	1.030	-9.90	11.51	0.37	0.09	0.72	0.82	0.968
201.60	0.974	0.310	0.197	1.041	17.67	10.90	0.82	-0.17	-1.43	1.66	0.956
230.40	0.750	0.727	0.137	1.054	44.11	7.49	1.32	-0.33	-3.19	3.46	0.943
259.20	0.380	0.993	0.030	1.063	69.08	1.64	1.64	-0.37	-4.27	4.59	0.933
288.00	-0.058	1.057	-0.106	1.064	93.14	-5.85	1.69	-0.38	-4.56	4.88	0.933
316.80	-0.473	0.305	-0.253	1.053	117.59	-13.29	1.49	-0.36	-3.99	4.27	0.944
345.60	-0.781	0.565	-0.363	1.032	144.11	-20.92	1.06	-0.25	-2.60	2.82	0.966
374.40	-0.910	0.100	-0.417	1.007	173.71	-24.57	0.32	0.11	-0.60	0.69	0.993
403.20	-0.823	-0.396	-0.374	0.983	205.13	-22.34	-0.82	0.57	1.53	1.83	1.019
432.00	-0.534	-0.770	-0.226	0.964	235.28	-13.53	-2.07	0.76	3.24	3.92	1.040
460.80	-0.115	-0.943	0.002	0.950	263.07	0.10	-2.75	0.58	4.07	4.95	1.055
489.60	0.320	-0.850	0.252	0.942	290.61	15.53	-2.46	0.30	3.77	4.51	1.064
518.40	0.648	-0.511	0.453	0.941	321.70	28.75	-1.37	0.14	2.30	2.68	1.065
547.20	0.782	-0.025	0.537	0.949	356.12	34.47	0.12	-0.03	0.14	0.18	1.056
576.00	0.699	0.468	0.472	0.965	393.81	29.32	1.61	-0.20	-1.95	2.54	1.039
604.80	0.438	0.836	0.269	0.982	422.38	15.92	2.66	-0.29	-3.41	4.33	1.020
633.60	0.090	0.991	-0.023	0.995	445.36	-1.32	3.01	-0.37	-3.97	5.00	1.006
662.40	-0.279	0.901	-0.334	1.000	467.20	-19.48	2.63	-0.41	-3.59	4.47	1.000
691.20	-0.556	0.591	-0.586	1.001	493.21	-35.82	1.64	-0.38	-2.38	2.92	0.999
720.00	-0.639	0.138	-0.711	1.000	528.69	-45.34	0.16	-0.17	-0.70	0.74	1.000

APPENDIX D

LISTINGS

INDEX OF SUBROUTINES FOR APPENDIX D

*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)
      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),PXX,PX,G,VR,DVR,VRA,VRAA,DAL,TTP,DX
      DOUBLE PRECISION A,AA,BA,U,V,CHEB,TT,T,T2,T3,TP,PI2,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/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
      COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),PI2,TTP
      COMMON/COUNT/NCOUNT,NCOUN,NLP
      COMMON/VREL/VR(3,2),VRA(3,2,2),VRAA(3,2,2,2),DVR(3,2),DAL(3,2,2)
      COMMON/PUT/NP1,NP2,D1,D2,H1,H2,PS,V1(6),V2(6)
      NF=0
      IF(NT.EQ.1.AND.NP1.EQ.NP3.AND.NP2.EQ.NP4.AND.D1.EQ.D3.AND.
     CD2.EQ.D4) NF=1
      NCOUNT=NCOUNT+1
      PI2=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)
      PV=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   NPD=NP*ND
      ND2=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.+DELTA)*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=(VRA(3,2,1)*PX(ND2,1))**2+(FLOAT(2-ND)*PX(ND2-1,1))**2
      TEST2=(VRA(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.D0,1,ND)
      IF(NB2.EQ.1) CALL MULT1(VR(1,2),DVR(1,2),VR(1,2),1.D0,1,ND)
      DO 39 LN=1,NL
39    CALL MULT1(X(1,1,LN),DX(1,LN),X(1,1,LN),1.D0,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,ND,NP,N)
      GO TO 15
750   IF(NA.EQ.0) GO TO 925
      CALL        AVTEST(AV,T2,NLP,NL,ND,NP)
 775  IF(NLP.EQ.NL) GO TO 800
      CALL TSHIFT(X,XV,NL,NLP,ND,NP,N)
      GO TO 15
800   GO TO 925
925   IF(ND.EQ.NC.OR.ND.EQ.3) GO TO 950
      ND=NC
      DO 930 LN=1,NL
      DO 930 I=1,NP
```

```
930   X(I,3,LN)=0.
      GO TO 10
940   CONTINUE
      NCOUN=0
      NCOUNT=0
      WRITE(6,1009)
 950  CONTINUE
      H1=0.
      H2=0.
      DO 975 M=1,ND
      IF(NB3.EQ.2) VR(M,1)=AV(1,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(1,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,WV(1,LN),WV(1,LN),1,NP,N,1)
      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 PXX,PX,G
      DOUBLE PRECISION VR,DVR,VRA,VRAA,DAL
      COMMON/COUNT/NCOUNT,NCOUN,NLP
      COMMON/PXXPX/PXX(30,30,6),PX(30,6),G(90,6)
      COMMON/VREL/VR(3,2),VRA(3,2,2),VRAA(3,2,2,2),DVR(3,2),DAL(3,2,2)
      DATA PXX,PX,G /6120*0.D0/
      DATA NCOUNT,NCOUN,NLP/3*0/
      DATA VR,VRA,VRAA,DVR,DAL /60*0.D0/
      END


      REAL FUNCTION STARTF(DZ,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.+SQRT(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,U,V,CHEB,TT,T,T2,T3,TP,PI2,TTP,W
      COMMON/PUT/NP1,NP2,D1,D2,H1,H2,PS,V1(6),V2(6)
      COMMON/VREL/VR(3,2),VRA(3,2,2),VRAA(3,2,2,2),DVR(3,2),DAL(3,2,2)
      COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),PI2,TTP
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NO,NH
      COMMON/CONS1/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
      Q1=ATAN2(V1(2),V1(1))
      Q2=ATAN2(V2(2),V2(1))
      Q=Q2-Q1+6.283185
      Q=AMOD(Q,6.283185)
      Q=Q+6.283185*FLOAT(NR)
      IF(NT.EQ.1)  Q=TA+ATAN2(SIN(Q-TA),COS(Q-TA))
      TA=Q
      RO1=V1(1)**2+V1(2)**2
      RO2=V2(1)**2+V2(2)**2
      R1=SQRT(RO1+V1(3)**2)
      R2=SQRT(RO2+V2(3)**2)
      R=R2-R1
      O1=ATAN(V1(3)/SQRT(RO1))
      O2=ATAN(V2(3)/SQRT(RO2))
      O=O2-O1
      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=MINO(NLP,6)
      IF(NT.EQ.0) NL=NLP
      DO 5 LN=1,NL
      IF(NT.EQ.0) T(LN)=TT/FLOAT(NL)
      IF(NT.EQ.1) T(LN)=T(LN)*TT/TTP
      T2(LN)=1./T(LN)**2
      T3(LN)=T2(LN)/T(LN)
5     TP(LN)=(PI2*T(LN))**2
      IF(NT.EQ.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
      O1T=(R1*V1(6)-R1T*V1(3))/(R1*RO1)
      O2T=(R2*V2(6)-R2T*V2(3))/(R2*RO2)
      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=.0000001
      IF(Q.EQ.0.) Q=.0000001
      IF(O.EQ.0.) O=.0000001
      R1S=TT*R1T/R
      R2S=TT*R2T/R
      Q1S=TT*Q1T/Q
      Q2S=TT*Q2T/Q
      O1S=TT*O1T/O
      O2S=TT*O2T/O
      W=0.
      DO 10 LN=1,NL
      DO 8 I=1,NP
      S=(W+CHEB(I)*T(LN))/TT
      D=O1+STARTF(O1S,O2S,S)*O
      E=Q1+STARTF(Q1S,Q2S,S)*Q
      F=R1+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
79    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(10,3),F(10,3),AV(10,3),W(10),B(10,3),G(10,3,3),
     CQ(10,3),H(10,3,3),C(10,3),D(10,3),R3(10),R4(10),P,Z
      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,ND2,NPD2,NPD1,NPOW,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(NPOW.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)
```

```
19    D(I,M)=WR*AV(I,M)-3.*R3(I)*C(I,M)
      DO 29 M1=1,ND
      DO 29 M2=1,ND
      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)
39    CONTINUE
      CALL MULT4(AA,Q,B,ND,NP,N,1)
      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
49    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
75    R1=R1+Q(I,M)*C(I,M)
      DO 79 M1=1,ND
      D(I,M1)=2.5*C(I,M1)*R1-Q(I,M1)
      DO 79 M2=1,M1
      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,PI2
      DOUBLE PRECISION X(30,6),PXX,PX,P,PL,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),TP(6),PI2
      COMMON/CONS1/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,N,1)
      CALL WYDER(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)
      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/CONS1/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
      PRO=0.
      DO 5 LN=1,NL
      CALL MULT3(V,DX(1,LN),PXX(1,11,LN),ND,NP)
5     CALL MULT4(BA,PXX(1,11,LN),PXX(1,10,LN),ND,NP,N,1)
      CALL DDERIV(0.D0,P,PRO,0)
      RO=SECANT(P,PRO)
      RHO=RO+1.
      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 MULT1(X(1,LN),DX(1,LN),X(1,LN),RO,ND,NP)
      CALL ALIGN(X)
      RETURN
      END


      DOUBLE PRECISION FUNCTION SECANT(P,PRO)
      DOUBLE PRECISION RO,P,PRO,RO2,PRO2,P2
      RO2=-.5
      RO=0.
      DO 5 IT=1,20
      CALL DDERIV(RO2,P2,PRO2,1)
      IF(IT.EQ.20) GO TO 10
      IF(DABS((P2-P)/P).LT..0001.AND.DABS(PRO2/P2).LT.1.) GO TO 10
      P=P2
      P2=(RO*PRO2-RO2*PRO)/(PRO2-PRO)
      RO=RO2
      RO2=P2
5     PRO=PRO2
10    SECANT=RO2
      P=P2
      RETURN
      END


      SUBROUTINE DDERIV(RO,P,PRO,NS)
      DOUBLE PRECISION VR,DVR,VRA,VRAA,DAL
      COMMON/VREL/VR(3,2),VRA(3,2,2),VRAA(3,2,2,2),DVR(3,2),DAL(3,2,2)
      DOUBLE PRECISION P,PRO,RO,W,Z
      DOUBLE PRECISION DH(3,2),DHRO(3,2),H,RHO,U(3),V(3),DSQRT
      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/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),PI2
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NO,NH
      COMMON/OUT/XV(30,6),AV(30,6),WV(10,6)
      COMMON/PUT/NP1,NP2,D1,D2,H1,H2,PS,V1(6),V2(6)
      RHO=RO+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)
      H=T(LN)*(FLOAT(NB-1)*H2+FLOAT(2-NB)*H1)
      Z=0.
      W=0.
      DO 30 M=1,ND
      V(M)=DVR(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=Z+V(M)**2
30    W=W+U(M)*V(M)
      Z=1./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(XV(1,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 WYDER(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(I,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),TP(6),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),PX(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,B,Q,G,HH,W,T3,L1,L2)
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NO,NH
      DOUBLE PRECISION P(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/SCRACH/R(10,10),S(10,10)
      DOUBLE PRECISION A,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,NPM1)
      CALL MULT4(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(NPM1))*T3
      IF(NPOW.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(I2,I1)=F(I1,I2)
      CALL MULT9(BA,F,E,NP,N,1)
      CALL MULT9(BA,E,D,NP,N,2)
   81 DO 99 M1=1,ND
      DO 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,M3,M1)*H(I2,M3,M2)
      Z=H(I1,M2,M1)*E(I2,I1)+E(I1,I2)*H(I2,M1,M2)+Z*F(I1,I2)
      IF(NO.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.NPM1)) 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,NPM1)
      K1=(L1-2)*ND+M1
      DO 99 I1=L1,L2
      K1=K1+ND
      J2=L2
      IF(M1.EQ.M2) J2=I1
      K2=(L1-2)*ND+M2
      DO 99 I2=L1,J2
      K2=K2+ND
      PXX(K1,K2)=(R(I1,I2)+V(I1,1)*R(2,I2)+V(I1,2)*R(NPM1,I2))*T3
99    PXX(K2,K1)=PXX(K1,K2)
      RETURN
      END


      SUBROUTINE PATCH(T,H1,H2)
      DOUBLE PRECISION XV,AV,WV,VA(3),VB(3),VAA(3),VAB(3),VBB(3)
      DOUBLE PRECISION PXX,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/PXXPX/PXX(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    AV(I,LN)=PXX(I,I,LN)
      DO 10 NB=1,2
      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)+ND*(2-NB)
      M3=(NPD2+1)*(NB-1)+ND2*(2-NB)
      M4=(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     VAA(M)=P*VRAA(M,1,1,NB)
      DO 7 I=1,NPD
      Q=0.
      R=0.
      DO 6 M=1,ND
      R=R+PXX(I,M1+M,LN)*VB(M)
6     Q=Q+PXX(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+WV(M1+M,2)*VB(M)+FNO*PX(M1+M,LN)*VBB(M)
8     Q=Q+WV(M1+M,1)*VA(M)+FNO*PX(M1+M,LN)*VAA(M)
      PX(M3,LN)=P
      PX(M4,LN)=S
      PXX(M3,M3,LN)=Q
      PXX(M4,M4,LN)=R
      IF(NB.EQ.1) PXX(M3,M4,LN)=Z
      IF(NB.EQ.2) PXX(M4,M3,LN)=Z
      DO 9 I=1,NPD2
      K=ND2+I
      IF(NB.EQ.2) PXX(M3,I,LN)=WV(I,1)
      IF(NB.EQ.2) PXX(M4,I,LN)=WV(I,2)
      IF(NB.EQ.1) PXX(K,M4,LN)=WV(K,2)
9     IF(NB.EQ.1) PXX(K,M3,LN)=WV(K,1)
10    CONTINUE
      IF(NL.EQ.1) RETURN
      DO 2 LN=2,NL
      Q=T(LN-1)
      R=T(LN)
      DO 1 J=1,NPD2
      DO 1 I=1,ND
      PXX(ND2+J,ND+I,LN)=PXX(ND2+J,ND+I,LN)*R
      Z=PXX(NPD1+I,J,LN-1)
      PXX(NPD1+I,J,LN-1)=-PXX(NPD2+I,J,LN-1)*Q
1     PXX(NPD2+I,J,LN-1)=Z
      DO 4 I=1,ND
      Z=PX(NPD1+I,LN-1)+PX(I,LN)
      PX(NPD1+I,LN-1)=-PX(NPD2+I,LN-1)*Q+PX(ND+I,LN)*R
      PX(NPD2+I,LN-1)=Z
      DO 4 J=1,ND
      IF(J.GT.I) GO TO 4
      Z=PXX(NPD1+I,NPD1+J,LN-1)+PXX(I,J,LN)
      PXX(NPD1+I,NPD1+J,LN-1)=PXX(NPD2+I,NPD2+J,LN-1)*Q**2
     C+PXX(ND+I,ND+J,LN)*R**2
      PXX(NPD2+I,NPD2+J,LN-1)=Z
4     PXX(NPD1+I,NPD2+J,LN-1)=PXX(ND+I,J,LN)*R-PXX(NPD2+I,NPD1+J,LN-1)*Q
2     CONTINUE
      RETURN
      END


      SUBROUTINE MODIFY(PX,DX,H1,H2)
      DOUBLE PRECISION PX(30,6),DX(10,3,6),VR,VRA,VRAA,DAL,DVR
      DOUBLE PRECISION TT,T,T2,T3,TP,PI2,DAL1(2),DAL2(2)
      COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),PI2
      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)
      DAL1(1)=-PX(ND2,1)
      DAL1(2)=-PX(NPD2+1,NL)
      DAL2(1)=FLOAT(2-ND)*PX(ND2-1,1)
      DAL2(2)=FLOAT(2-ND)*PX(NPD2+2,NL)
      DO 50 NB=1,2
      IF((NB.EQ.1.AND.NB1.NE.1).OR.(NB.EQ.2.AND.NB2.NE.1)) GO TO 50
      DO 45 M=1,ND
      DAL(M,1,NB)=VRA(M,1,NB)*DAL1(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    DVR(M,NB)=DAL(M,1,NB)+.5*DAL(M,2,NB)
50    CONTINUE
      DO 5 M=1,ND
      DO 4 LN=1,NL
      DO 4 I=1,NP
      K=(I-1)*ND+M
4     DX(I,M,LN)=-PX(K,LN)
      IF(NB1.NE.0) DX(2,M,1)=0.
      IF(NB2.NE.0) DX(NPM1,M,NL)=0.
      DX(1,M,1)=0.
5     DX(NP,M,NL)=0.
      IF(NL.EQ.1) RETURN
      DO 10 LN=2,NL
      DO 10 M=1,ND
      DX(2,M,LN)=T(LN)*DX(2,M,LN)
      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,PI2,VR,DVR,VRA,VRAA,DAL
      COMMON/VREL/VR(3,2),VRA(3,2,2),VRAA(3,2,2,2),DVR(3,2),DAL(3,2,2)
      COMMON/PUT/NP1,NP2,D1,D2,H1,H2,PS,V1(6),V2(6)
      COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),PI2
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NO,NH
      DO 3 M=1,ND
      DVR(M,1)=VR(M,1)-DVR(M,1)
3     DVR(M,2)=VR(M,2)-DVR(M,2)
      IF(NB1.EQ.1) CALL VCAL(VR(1,1),VRA(1,1,1),VRAA(1,1,1,1),ND)
      IF(NB2.EQ.1) CALL VCAL(VR(1,2),VRA(1,1,2),VRAA(1,1,1,2),ND)
      DO 10 M=1,ND
      IF(NB1.NE.0) X(2,M,1)=T(1)*(V1(M+3)+H1*VR(M,1))
      IF(NB2.NE.0) X(NPM1,M,NL)=T(NL)*(-V2(M+3)+H2*VR(M,2))
      X(1,M,1)=V1(M)
10    X(NP,M,NL)=V2(M)
      IF(NL.EQ.1) RETURN
      DO 5 LN=2,NL
      DO 5 M=1,ND
      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
      DOUBLE PRECISICN PXX,PX,G,XV,AV,WV,R,S,Z,A,AA,BA,U,V,CHEB
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NO,NH
      DOUBLE PRECISION TT,T,T2,T3,TP,PI2
      COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),PI2
      COMMON/CONS1/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/PXXPX/PXX(30,30,6),PX(30,6),G(10,3,3,6)
      COMMON/OUT/XV(30,6),AV(30,6),WV(10,6)
      DO 20 LN=1,NL
      DO 20 I=1,NPD
      DO 15 J=1,I
15    PXX(I,J,LN)=PXX(J,I,LN)
      PX(I,LN)=XV(I,LN)
20    PXX(I,I,LN)=AV(I,LN)
      IF(NO.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
65    S(NP+1-I1,NP+1-I2)=Z
      DO 100 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)
      L3=MIN0(L1,2)
      L4=MAX0(L2,NPM1)
      DO 99 M1=1,ND
      DO 99 M2=1,M1
      K1=(L1-2)*ND+M1
      DO 99 I1=L1,L2
      K1=K1+ND
      J2=L2
      IF(M1.EQ.M2) J2=I1
      K2=(L1-2)*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(I1,M1,M2,LN)
      PXX(K1,K2,LN)=PXX(K1,K2,LN)-Z*T3(LN)
99    PXX(K2,K1,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,V,CHEB,TT,T,T2,T3,TP,PI2,TS
      COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),PI2
      COMMON/CONS1/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 MULT3(V,X(1,LN),XV(1,1,LN),ND,NP)
```

```
      I=1
      THET=0.
      TS=0.
      DO 3 II=1,NL
      DO 2 J=1,NP
      I=I+1
      XC=XV(J,1,II)
      YC=XV(J,2,II)
      THETA(I)=ATAN2(YC,XC)
      TIME(I)=TS+T(II)*CHEB(J)
      DO 1 K=1,6
      IF(THET.LT.THETA(I)+.01) GO TO 27
1     THETA(I)=THETA(I)+PI2
27    IF(THETA(I).GT.THET+1.57.AND.I.GT.2)GO TO 950
      THET=THETA(I)
      THETA(I)=THET*(XC**2+YC**2)**EX
    2 CONTINUE
    3 TS=T(II)+TS
      SUBARC=(THETA(I)-THETA(2))/FLOAT(NLP)
      II=1
      ARC=THETA(2)+SUBARC
      DO 4 J=4,I
      IF(THETA(J-1).LT.ARC) GO TO 4
      TP(II)= TIME(J-2)+(ARC-THETA(J-2))*(TIME(J-1)-TIME(J-2))/
     1(THETA(J-1)-THETA(J-2))
      ARC=ARC+SUBARC
      II=II+1
      IF(II.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,T,T2,T3,TP,PI2,S,W
      DIMENSION G(3)
      DOUBLE PRECISION X(30,6),Y(10,3,6)
      COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),PI2
      COMMON/CONS1/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 LNP=1,NLP
      DO 13 I=1,NP
      S=W+TP(LNP)*CHEB(I)
      CALL MODLEG(T,NL,S,LN)
      CALL POLEVL(X(1,LN),G,S,ND,NP,N)
      DO 13 M=1,ND
13    Y(I,M,LNP)=G(M)
```

```
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)=(PI2*T(LNP))**2
      NL=NLP
      RETURN
      END


      SUBROUTINE OUTCAL
      DOUBLE PRECISION TT,T,T2,T3,TP,PI2,S,XV,AV,WV
      COMMON/PUT/NP1,NP2,D1,D2,H1,H2,PS,V1(6),V2(6)
      COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),PI2
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NO,NH
      COMMON/OUT/XV(30,6),AV(30,6),WV(10,6)
      COMMON/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(WV(1,LN),Z,S,1,NP,N)
      B(I,12)=1./Z**2
      CALL POLEVL(XV(1,LN),G,S,ND,NP,N)
      CALL POLEVL(AV(1,LN),H,S,ND,NP,N)
      W=0.
      Z=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(B,NP1,NP2,D1,D2,H1,H2,PS,NL,ND,NP,NB1,NB2,NPOW)
      RETURN
      END


      SUBROUTINE OUTPUT(A,P1,P2,D1,D2,H1,H2,JV,NL,ND,NP,NB1,NB2,NPOW)
      DIMENSION A(26,12) ,UNCST(3)
      DOUBLE PRECISION PLANET(12),POWOPT(2)
      INTEGER P1,P2
      DATA UNCST/4H  UN,4H    ,4H     /
      DATA POWOPT/ 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) POWOPT(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 , UNCST(NB1+1), UNCST(3)
      WRITE(6,121) PLANET(P2), H2 , UNCST(NB2+1), UNCST(3)
      WRITE(6,109)
      DO 1 I=1,26
    1 WRITE(6,100) (A(I,J),J=1,12)
      RETURN
  100 FORMAT(F9.2,4F8.3,6F8.2,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   TIME        X        Y        Z
     1        ,25H   R      THETA     PHI
     4        ,23H    AX      AY      AZ
     2           ,30H  MAG. A    P/PO
     3 //)
  120 FORMAT(// 15X,I5,29H 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,
     111HCONSTRAINED , A4 )
      END


      SUBROUTINE OUTPT1(P1,P2,D1,D2,H1,H2,JV,NL,ND,NP,NB1,NB2,NPOW)
      DIMENSION UNCST(3)
      DIMENSION A(4)
      DATA A/4H**** , 4H**** , 4H****  , 4H****  /
      INTEGER P1,P2
      DOUBLE PRECISION PLANET(12),POWOPT(2)
      DATA UNCST/4H  UN,4H    ,4H     /
      DATA POWOPT/ 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,A
      WRITE(6,5432)
      WRITE(6,700) A,A,A,A,A,A,A
      WRITE(6,101) POWOPT(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 ,
     1UNCST(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,19HEXCESS VELOCITY AT ,A8,2HIS,E20.6,2X,A4,
     211HCONSTRAINED , 7H   , AT , A8,2HIS  ,    E20.6,2X,A4,
     111HCONSTRAINED , A4 )
  101 FORMAT(5X,16HVARIABLE THRUST ,A8,20H POWER TRAJECTORY IN , I2 ,1X,
     116HDIMENSIONS USING  ,I2 , 8H LEGS OF  ,I3,17H 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)=-VRA(2,2)
      VRAA(2,1,2)=VRA(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,BA,U,V,CHEB,DH(3),X(10,3),BAX(10,3)
      DOUBLE PRECISION VDH(2),VDHP,VDHM,V1,V2
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NO,NH
      COMMON/CONS1/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
      DO 50 M=1,ND
      VDH(NB)=(1.+V(2,1))*DH(M)
      VDH(3-NB)=V(2,2)*DH(M)
      X(2,M)=X(2,M)+VDH(1)
      X(NPM1,M)=X(NPM1,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)*VDHP
      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)
525   AVM(2,LN)=SQRT(Z2)*T2(LN)
      DO 10 LN=2,NL
      RATIO=(AVM(1,LN)-AVM(2,LN-1))/AMIN1(AVM(1,LN),AVM(2,LN-1))
      IF(ABS(RATIO).GT..1)NLP=MIN0(6,NL+1)
   10 CONTINUE
      RETURN
      END


      SUBROUTINE POLEVL(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     DO 10 M=1,ND
      Z=0.
      DO 5 I=1,N
      K=NP+1-I
5     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
      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 79
75    V1=V1+A(J,I,1)*U1(J)
      V2=V2+A(J,I,2)*U2(J)
79    CONTINUE
      Y(I,M)=V2-V1
99    Y(K,M)=V2+V1
      RETURN
      END


      SUBROUTINE MULT7(A,B,C,N,NS)
      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,NS)
      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(I,J)=A1(I,J,2)+A1(I,J,1)
      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)
```

```
2     CONTINUE
      RETURN
      END


      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+1-I
      DO 1 J=1,N
      L=NP+1-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)
1     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 FNPM1,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
      FNPM1=FLOAT(NPM1)
      DO 1 I=1,NP
      Z=3.14159265358979323846D0*FLOAT(I-1)/FNPM1
      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)=DCOS(Z)
      Q(1)=-(1.+2.*FNPM1**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+IABS(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+I-K)**2-1)
      F2=FLOAT(4*(J-I)**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(I)-1.
      V(I,1)=(Q(NPM1)*Q(K)-Q(2)*Q(I))/Z
      IF(I.EQ.2) V(I,1)=Q(2)/Z-1.
      IF(I.EQ.NPM1) V(I,1)=-Q(NPM1)/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.-FNPM1**2)/3.-(1.+3.*R(2*J-1))/P(2*J-1))/P(J)
      BA(J,J,1)=BA(J,J,2)-8./P(2*J-1)
      GO TO 40
35    BA(J,J,2)=(4.*(4.+FNPM1**2*(FNPM1**2-5.)))/15.
      BA(J,J,1)=BA(J,J,2)-(8.*(1.-FNPM1**2))/3.
40    A(1,J,2)=.5*A(1,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,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(30,6),X(30,6),W,Z,DSQRT
      IND=3
      NCP1=N1
      NCP3=NCP1
      NCP2=N
      DO 45 KK=1,M
      IF(KK.EQ.M) NCP2=N2
      IF(KK.EQ.1) GO TO 9
      DO 5 J=1,NCPL
      JM1=J-1
      X(J,KK)=X(N-NCPL+J,KK-1)
      DO 2 I=J,NCPL
2     A(I,J,KK)=A(N-NCPL+I,N-NCPL+J,KK-1)
      DO 5 I=NCP1,NCP2
      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=NCP1,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)
10    Z=Z-A(J,K,KK)**2
20    IF(Z.LT.0.) RETURN
      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,K,KK)*A(J,K,KK)
   40 A(I,J,KK)=Z*A(J,J,KK)
43    NCP3=1
45    NCP1=NCPL+1
      NCP1=N+1-N2
      NCP3=NCP1
      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,KKK)=X(J,KKK+1)
89    DO 90 J=NCP1,NCP2
      L=N+1-J
      JM1=J-1
      Z= X(L,KKK)
      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 NCP1=NCPL+1
      IND=2
      RETURN
      END

      SUBROUTINE EPHEM(D,N,V)
      DIMENSION V(6),E(6,10)
      DATA E /.387099,.205627,.1222427,.8352647,1.34099,3.885481,.723332
```

```
     C,.006793,.0592405,1.33203,2.286526,3.04201,1.,.016726,0.,0.,1.7846
     C43,1.748089,1.523691,.093368,.032287,.8595577,5.852485,4.516341,5.
     C202803,.048435,.0227828,1.746105,.2387301,4.534909,9.538843,.05568
     C2,.0434571,1.977588,1.610319,4.898639,19.18195,.047209,.0134924,1.
     C287988,2.967249,2.466237,30.05778,.008575,.0309578,2.292312,.77272
     C62,3.786333,39.43871,.250236,.2996713,1.917866,3.912334,3.170326,
     C1.,0.,.8,4.7,0.,1.748089/
      DATA EPOCH /2436935./
      AM=DMOD(.01720242*(D-EPOCH)/E(1,N)**1.5+E(6,N)-E(5,N),6.283185307)
      AE=AM
      DO 1 I=1,20
      B=AM+E(2,N)*SIN(AE)
      IF(ABS(B-AE).LT..000001) GO TO 2
1     AE=B
2     AT=2.*ATAN(SQRT((1.+E(2,N))/(1.-E(2,N)))*TAN(AE/2.))
      P=E(1,N)*(1.-E(2,N)**2)
      R=P/(1.+E(2,N)*COS(AT))
      RT=6.283185*E(2,N)*SIN(AT)/(R*SQRT(P))
      A=6.283185*SQRT(P)/R
      U=(E(5,N)-E(4,N))+AT
      V(1)=R*(COS(U)*COS(E(4,N))-SIN(U)*COS(E(3,N))*SIN(E(4,N)))
      V(2)=R*(COS(U)*SIN(E(4,N))+SIN(U)*COS(E(3,N))*COS(E(4,N)))
      V(3)=R*SIN(U)*SIN(E(3,N))
      V(4)=RT*V(1)-A*(SIN(U)*COS(E(4,N))+COS(U)*COS(E(3,N))*SIN(E(4,N)))
      V(5)=RT*V(2)-A*(SIN(U)*SIN(E(4,N))-COS(U)*COS(E(3,N))*COS(E(4,N)))
      V(6)=RT*V(3)+A*(COS(U)*SIN(E(3,N)))
      RETURN
      END


      DOUBLE PRECISION FUNCTION POWER(R)
      DOUBLE PRECISION R,DSQRT,A1,A2
      DATA A1,A2 /2.825D0  ,1.825D0  /
      IF(R.GT.(5.*A2/(4.*A1))**2) GO TO 1
      POWER=(A1/(5.*A2/(4.*A1))**4)/5.
      RETURN
1     POWER=(A1-A2/DSQRT(R))/R**2
      RETURN
      END


      DOUBLE PRECISION FUNCTION DPOWER(R)
      DOUBLE PRECISION R,DSQRT,A1,A2
      DATA A1,A2 /-5.65D0  ,4.5625D0 /
      DPOWER=(A1+A2/DSQRT(R))/R**3
      IF(DPOWER.GT.0.) DPOWER=0.
      RETURN
      END


      SUBROUTINE TCTPRE(TW,SI)
      COMMON/BDYP/BDY(3,3,2),PV,PC
      COMMON/COUNT/NCOUNT,NCOUN,NLP
      COMMON /TIMEQ/ TT
      DOUBLE PRECISION TT
      DIMENSION ST(30)
      COMMON/PXXPX/PR(301),TYME(301),VA(301)
10    IF(NCOUNT.EQ.0) RETURN
11    IF(NCOUN.EQ.NCOUNT)GO TO 950
      WRITE(6,4321)
4321  FORMAT(1H1)
      NCOUN=NCOUNT
      NNN=101
      DT= TT/(NNN-1)
      DO 1 I=1,NNN
    1 PR(I)= PMAP((I-1)*DT)
      CALL RESCAL  (PR,TYME,NNN,TAVSTR)
      DO 4 I=1,NNN
    4 VA(I)= AMAP(TYME(I)*TT)/PMAP(TYME(I)*TT)
      T=TT*TAVSTR
  950 CONTINUE
      BO=TW*31557600./SI
      C=SI*65280./31557600.
      BOT=BO*T
      CT=C/T
      CALL ROOTR(VA,NNN,CT,BOT,ST,LGF)
8     IF(LGF) 2,2,3
3     TB=ST(1)*T
      DO 199 I=1,30
      U=T*365.25*ST(I)
      K=I-2*(I/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*(BO*C)**2)/(1.-BO*TB)
      PS=.7121*PC
      CALL INVINT(ST,TYME,NNN)
      T=TT
      DO 201 I=1,30
  201 ST(I)= T*365.25*ST(I)
      CALL OUTTOO(TW,SI,PS,ST)
      RETURN
    2 CONTINUE
50    WRITE(6,1000)
1000  FORMAT(//4X46HMISSION IMPOSSIBLE EVEN WITH CONTINUOUS THRUST)
      ST(1)= TT
      ST(2)=0.
      CALL OUTTOO(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
      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.
      DO 1 I=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
      J=2
      DO 4 I=2,N
      S=(I-1)*SUM/(N-1)
      DO 2 K=1,N
      L=J+K-1
      IF(L.EQ.N) GO TO 3
      IF(S.LE.TAUS(L)) GO TO 3
2     CONTINUE
3     J=L
    4 TAU(I)= T(J-1)+((S-TAUS(J-1))/(TAUS(J)-TAUS(J-1)))*(T(J)-T(J-1))
      SCALE=SUM/(3*N-3)
      RETURN
      END


      SUBROUTINE ROOTR(AV,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,2
      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
9     JVTEST=BZ*C*AV(1)
      DO 10 I=2,NAV,2
```

```
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
      RETURN
   50 CONTINUE
      LGF=-1
      RETURN
      END


      SUBROUTINE OUTTOO(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    ,
     18H   OFF       ,8H    ON     ,8H   OFF     ,8H    ON     ,8H   OFF    ,
     28H    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=I
    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
      IF(KKK.EQ.J1) J= J2-(J1-1)*J
      WRITE(6,301) (THRUST(I),I=1,J)
      WRITE(6,301) (    OFF(I),I=1,J)
      WRITE(6,301) (     AT(I),I=1,J)
      WRITE(6,301) (   TIME(I),I=1,J)
      WRITE(6,302)    (T(I),  I=1,J)
      WRITE(6,5432)
      DO 3 I=J,29
      T(I+1-J)= T(I+1)
3     CONTINUE
      WRITE(6,700) A,A,A,A,A,A,A
      WRITE(6,4322)
      WRITE(6,700) A,A,A,A,A,A,A
      RETURN
5432  FORMAT(10X/10X)
 301  FORMAT( 4X,A8,4X,A8,4X,A8,4X,A8,4X,A8,4X,A8,4X,A8,4X,A8,4X,A8,
     14X,A8   )
 302  FORMAT( 10F12.3 )
 200  FORMAT(// 2X, 37HPREDICTED CONSTANT THRUST TRAJECTORY        )
 201  FORMAT(/  2X, 25HTHRUST TO WEIGHT RATIO         , F12.9      )
 202  FORMAT(/  2X, 17HSPECIFIC IMPULSE               , F12.1 , 6H  SEC. )
 203  FORMAT(/  2X, 24HPERFORMANCE INDEX (JC)         , E16.6          )
4322  FORMAT( 10H             )
 700  FORMAT(1H  ,28A4 )
      END


      SUBROUTINE KROOT(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(1)-K
      IGL=C*B*AV(1)
      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-K-(JV+DJ)/C
      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.E40
      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 MN,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
```

```
MN=ML-B*DTS
AV2=(AV(1)*(DT-DTS)+AV(2)*DTS)/DT
IGN=C*B*AV2/MN**2
DJ=0.5*(IGL+IGN)*DTS
JV=JV+DJ
JI=JI+0.5*(IGL*ML+IGN*MN)*DTS
KL=AV(2)/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,T,DT,ML,KL,IGL,JV,TSW,JI)
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=DT-DTS
MN=ML-B*DTS
IGN=C*B*AV1/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,PI2,S
COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NO
COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),PI2
COMMON/OUT/XV(30,6),AV(30,6),WV(10,6)
S=H
CALL MODLEG(T,NL,S,LN)
CALL POLEVL(WV(1,LN),G,S,1,NP,N)
PMAP=1./G**2
RETURN
END


REAL FUNCTION AMAP(H)
DOUBLE PRECISION XV,AV,WV,TT,T,T2,T3,TP,PI2,S
COMMON/OUT/XV(30,6),AV(30,6),WV(10,6)
COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NPOW,NB1,NB2,NO
COMMON/TIMEQ/TT,T(6),T2(6),T3(6),TP(6),PI2
```

```
      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
```