General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

CR 114354
AVAILABLE TO THE PUBLIC

BOEING

Final Report for
CHEBYCHEV TRAJECTORY OPTIMIZATION PROGRAM (CHEBYTOP II)

D180-12916-1

by

D. W. Hahn and F. T. Johnson

June 1971

Prepared Under Contract NAS2-5994

THE BOEING COMPANY
Research and Engineering Division
Seattle, Washington

for

ADVANCED CONCEPTS AND MISSIONS DIVISION/OART
MOFFETT FIELD, CALIFORNIA
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

TABLE OF CONTENTS

I. SUMMARY

CHEBYTOP II is an analysis tool that enables the user to conduct rapidly the parametric analysis and optimization of interplanetary missions employing electrically propelled spacecraft. As the name implies, this is the second version of CHEBYTOP to be released for distribution. Both versions were developed under contract with Advanced Concepts and Missions Division/OART. The analytical and numerical algorithm used is termed the Chebychev Optimization Method. The method avoids numerical integration and the usual variational calculus solution by employing polynomial representation and several other specialized techniques.

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 major planets are available, plus 8 additional sets of elements as listed in Table 1. Either rendezvous (matching the planet's orbital velocity about the sum), 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.

The program will, at the user's option, determine optimum departure or arrival dates between planets. It will also determine optimum in-plane travel angles between orbits or between planets or orbits.

CHEBYTOP II 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 effects of thrust angle constraints on performance can be determined. Either powerplant mass fraction or the exhaust velocity, or both can be optimized. A tankage factor can be included in the determination of the payload fraction.

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. Solar panel tilt and time dependent power degradation can be considered.

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

Coding was done in FORTRAN IV for the IBM 360 or CDC 6000 series computers. The core requirements are 170,000 bytes. No auxiliary tapes are needed.

II. INTRODUCTION

The success and comparatively wide distribution of the original CHEBYTOP program prompted a follow-on effort to expand the programs capabilities. Therefore, CHEBYTOP II has been developed and supersedes the original computer program. All of the capabilities of the first version have been retained and a number of new features and improvements have been added. CHEBYTOP II is constructed so that it can be used as a subroutine of a master program. In fact, it contains several features which make it especially suited to QUICKTOP, a mission analysis master program described in Reference 1.

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 an analytical and numerical optimization method, described in Reference 2, called by 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 II. Hence, the program is composed of two major subprograms. The first, VTMODE, solves the unconstrained continuous-thrust case (Variable Thrust Mode) using the Chebychev Optimization Method of Reference 2. The second subprogram, CTMODE, solves for the constant specific impulse, constrained thrust mode, using the prediction scheme described in Appendix D.

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 II 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.

III. ANALYSIS

DEFINITION OF PROBLEM

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

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

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

$$J = \int_o^T |a|^2 \frac{p_o}{p(r,t)} dt$$

subject to boundary conditions on x and thrusting constraints on a. The trip time, T, is always assumed to be fixed. The end points, x(o) and x (T), can be fixed or allowed to vary in a prescribed orbit so as to determine optimum launch or arrival times. 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).

The acceleration, a, may be completely unconstrained (variable thrust) or else it may be assumed to derive from a constant Isp engine with shut down and start up capability. In the latter case additional constraints on the thrusting angle may also be imposed. For a constant Isp mode we have

$$a(t) = \frac{a_o}{\mu} \frac{p(r,t)}{p_o} \nu(t), \quad \mu(t) = \frac{a_o}{c} \frac{p(r,t)}{p_o} \sigma(t)$$

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 ν is a unit direction vector.

$$\nu = \begin{cases} 1 \text{ during powered phase} \\ o \text{ during coast} \end{cases}$$

$$\sigma \cdot = \sqrt{\nu,\nu}$$

For constraints on thrusting angle we have in addition

$$(\nu_1 - \frac{x}{r}) - \sigma \cos \theta = 0, \quad \theta \in \Theta$$

Here Θ is a set of cone angles (we allow a maximum of four).

The parameters a_o, c and the set Θ may be optimized for maximum payload. Actually instead of a_o it is more convenient to optimize an equivalent quantity.

$$\mu_w = \frac{\alpha_w a_o c}{2\eta(c)}$$

where μ_w is the powerplant mass fraction, α_w the powerplant specific mass, and $\eta(c)$ the efficiency curve of the thrusters. The simplest definition of payload is $\mu_1 - \mu_w$, where μ_1 is the final vehicle mass fraction. More complicated definitions are allowed.

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 constrained thrust 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.

However in the three years of the program's existence in one form or another, the constrained thrust approximation has been found to be sufficiently accurate to permit reliable mission analysis.

MATHEMATICS OF VARIABLE THRUST SOLUTION

The basic mathematical foundation of this part of the program is contained in Reference 2. Several modifications to the development found there are necessary because of the following:

1. Polynomial patching capabilities have been included for more efficient calculation of long trajectories.
2. A variable power option has been incorporated.
3. Optimal launch date capability has been incorporated.
4. A better iteration procedure has been developed for use in early iterations when the trajectory is not in the neighborhood of an optimum.

These changes are discussed in the following four sections:

1. Polynomial Patching

   It was noted in Reference 2 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. The elapsed time along the trajectory covered by each polynomial is varied internally by the program to obtain a better representation of the trajectory. J then becomes a weighted sum of the performance index for each polynomial segment, or 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 2. All differentiation formulas of Reference 2 can be applied separately to each P_i.

2. Variable Power

   The variable thrust algorithm of Reference 2 must be modified to account for power which varies as a function of time and/or heliocentric radius from the sun. In particular Equations (16), (17), and (18) of the reference are affected. Let p(r,t) be the spacecraft power relative to 1 au and/or t = o. Let W be a column vector with elements

$$W^{\nu} = \frac{\sqrt{p_o}}{\sqrt{p(r(s^{\nu}),\ t(s^{\nu}))}}$$

   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

$$G(m) = WAV(m)$$

i.e.

$$G(m)^{\nu} = W^{\nu}\, AV(m)^{\nu}$$

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 $H(i,j) = Yx(i,j) + \frac{Wx(j)}{W} AV(i)$ and S(m) = WR(m), we obtain

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

$$Pxx(i,j) = (A^{-1}BA)^T\, W^T FW(A^{-1}BA)\, \delta_{i,j}$$

$$+ (A^{-1}BA)^T\, W^T(FWH(i,j)$$

$$+ H(j,i)^T\, W^T FW\, (A^{-1}BA)$$

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

$$+ R(m)^T\, Gxx(m,i,j)$$

The last term can be expanded and is equal to

$$I(j,i)BA + BA^T I(i,j)^T + ID(i,j)$$

where $I(i,j) = S(i)\, \frac{Wx(j)}{W}$

and $ID(i,j) = S(m)\, [Yxx(m,i,j) + \frac{Wx(i)}{W} Yx(m,j)$

$$+ \frac{Wx(j)}{W} Yx(m,i) + AV(m)\, \frac{Wxx(i,j)}{W}\ ]$$

A redefined interpolation procedure was used in Reference 2 to reduce execution time computing partial derivatives. This procedure matched the polynomial derivatives at end points rather than interpolating the second and next to last Chebychev points. The inclusion of variable power eliminates this savings, however, and Px, Pxx are computed according to Equation (29), Reference 2.

3. Optimal Launch Dates

In Reference 2 optimization of hyperbolic excess velocity directions at launch or arrival was accomplished by adding two new parameters, α and β, the heliocentric longitude and latitude of the excess velocity vector. These parameters were included with all the rest to be optimized, and appeared functionally through the endpoint velocity elements of the state vector. A new parameter λ, the Julian date of launch or arrival, is added when one of these dates is to be optimized. The endpoint position and velocity elements of the state vector are then dependent on λ through Kepler's equations, and derivatives are computed accordingly.

Two different parameters λ_1, and λ_2 are considered when both the launch and arrival dates are free. (See section on INPUT). However only λ_2 is considered to be independent when launch and arrival dates are constrained by fixed trip time. Then $\lambda_1 = \lambda_2 - \tau$, where τ is trip time. The latter causes a considerable complication in the calculations because one free parameter appears in both the first and last legs of the trajectory.

4. Better Iteration Procedure

Generally, Newton's method is employed when the iterations have progressed to the point that the trajectory is in the neighborhood of an optimum. Away from an optimum Gauss' method is used (See Reference 2). Whereas Newton's method is a true second order convergence technique, Gauss' method employs only first derivative of the nonlinear functions involved in the payoff. On this account Gauss' method occasionally bogs down before reaching the point where Newton's method can be employed. Hence it has been necessary to modify Gauss' method to include second order term as follows.

Let us assume our payoff P as a function of a vector of free parameters x can be expressed

$$P(x) = G(x)^T G(x) \tag{1}$$

where G(x) is a nonlinear vector function of x, and T denotes transpose.

A Gauss' iteration producing an increment in the trajectory x at a nominal state x_o is defined by

$$[Gx(x_o)^T Gx(x_o)]\Delta x = -fGx(x_o)^T G(x_o) \tag{2}$$

Here f is a scale factor limiting the step size of Δx to insure a decrease in payoff. If Gauss' iteration bogs down Δx will be small and we may consider the differential equation

$$[Gx(x_o)^T Gx(x_o)] \frac{dx}{d\rho} = -f(\rho)Gx(x_o)^T G(x_o) \tag{3}$$

Differentiating once more,

$$[Gx(x_o)^T Gx(x_o)] \frac{d^2x}{d\rho^2} =$$

$$- \frac{df(\rho)}{d\rho} Gx(x_o)^T G(x_o)$$

$$- f(\rho)[Gxx(x_o)^T G(x_o) \frac{dx}{d\rho} + Gx(x_o)^T Gx(x_o) \frac{dx}{d\rho}] \tag{4}$$

$$- Gx(x_o)^T Gxx(x_o) \frac{dx}{d\rho} \frac{dx}{d\rho} - \frac{dx}{d\rho} Gxx(x_o) Gx(x_o) \frac{dx}{d\rho}$$

Equation (3) is to be solved first and the result substituted into (4). The actual increment in the trajectory will then be a function of ρ defined by

$$\Delta x(\rho) = \frac{dx}{d\rho} \rho + \frac{1}{2} \frac{d^2x}{d\rho^2} \rho^2 \tag{5}$$

A linear search on ρ is to be performed to find the value minimizing

$$P(x_o + \Delta x(\rho))$$

$f(\rho)$ may be chosen in a variety of ways.

$f(\rho) = \frac{1}{1 - \rho}$ will insure that the iteration reduces to the ordinary Gauss' iteration when Gxx = 0.

Equation (5) is a second order correction to Gauss' method, and in general results in faster convergence in those cases when Gauss' method must take small steps to insure decreasing payoff.

MATHEMATICS OF THE CONSTRAINED THRUST SOLUTION

The basic mathematics of the program's constant Isp prediction scheme are contained in Appendix D, which is an extension of the work done in Reference 3. Only the first side condition has been employed in the present program.

The scheme has been modified to include a solar power option and constraints on thrusting direction. A spacecraft centered coordinate system is adopted as shown in Figure 1. Reference directions are towards the sun and one other star. The usual star reference is Canopus, however, the user is free input to whatever reference desired. As supplied, the program uses a fictitious star located in the southern celestial sphere, normal to the ecliptic plane. Thrust directions are defined by cone and clock angles. Only cone angles can be constrained. The approximation technique assumes the spacecraft is free to rotate about the spacecraft-sun line. The basic assumption in the derivation given in Appendix D is that the constrained thrust trajectory profile is closely approximated by the variable thrust profile. The implications of this assumption are that the acceleration levels, variable thrust compared to constrained, cannot be vastly different, and the power profiles, constrained versus variable, are close to one another.

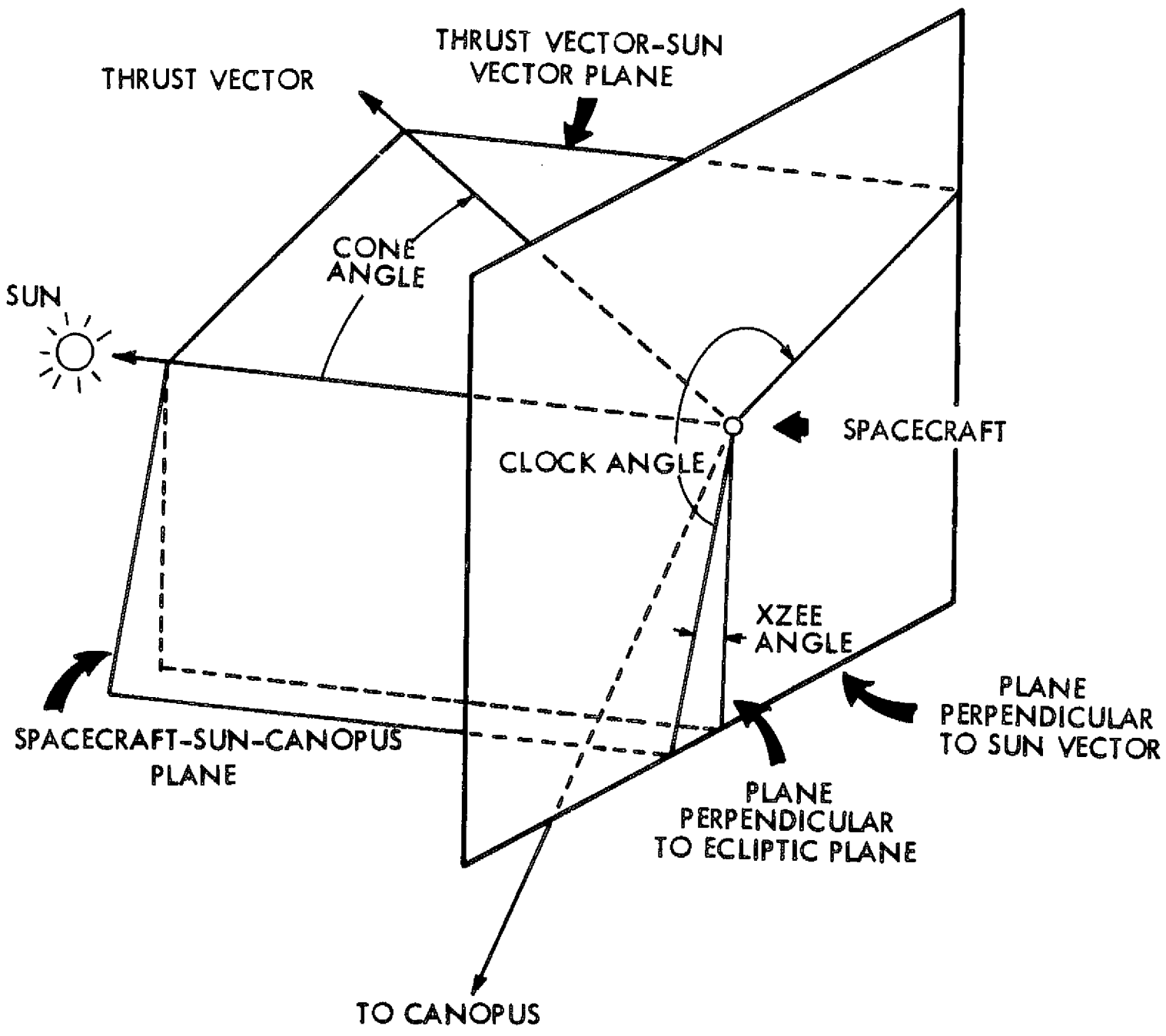


Figure 1- CONE & CLOCK COORDINATES

IV. PROGRAM ORRANIZATION

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, CTMODE. Perhaps the simplest mode of program operation is tc compile a MAIN program which contains one or more calls of subroutine VTMODE and CTMODE. The sample cases included in Appendix C use this form of input.

In addition to the calls to VTMODE and CTMODE, several optional parameters can be set in the MAIN program via COMMON statements. If these statements are omitted the parameters are given default values. The optional parameters are described following the VTMODE and CTMODE call list descriptions below.

Variable Thrust Mode Input

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

```
CALL VTMODE(NC,RN,NP1,NP2,NB1,NB2,NV1,NV2,D1,D2,HV1,HV2,NPOW,NT)
```

The arguments of the subroutine are:

- NC — an integer specifying the number of dimensions (2 or 3), except for out-of-the-ecliptic probes or similar trajectories which require a three-dimensional starting trajectory, in which case set NC = 1.
- RN — a floating point number which limits the number of revolutions about the sun, i.e., travel angle. The travel angle, measured at the sun, will fall between RN and RN + 2π. (Except in some instances where tracking is allowed, see NT description.) The units of RN are revolutions (RN = 1.5 is 3π radians).

See Comments on Input Constant Selection.

NP1 the integer number of the departure planet, varying from 1 for Mercury to 9 for Pluto*.

NP2 the analogous number of the arrival planet

NB1 an integer with values 0, 1, 2.

- 0 Launch position free in the orbit (not restricted to planet ephemeris).
- 1 Launch position tied to the launch planet, but variable in orbit as dictated by planet ephemeris. If NB1 = 1 then NB2 = 1 (program checks NB1 and sets NB2 if needed).
- 2 Launch position fixed.

NB2 an interger with values 0, 1, 2.

- 0 Arrival position free in orbit (not restricted to an ephemeris).
- 1 Arrival position tied to an ephemeris, but variable. If NB2 = 1 then NB1 must = 1.
- 2 Arrival position fixed.

NV1 Represents options for hyperbolic excess velocity at launch planet. NV1 = 0 flyby, 1 for specified hyperbolic excess speeds, and 2 for rendezvous.

NV2 Same as NV1, but for arrival planet.

D1 the modified Julian date of departure (Julian date minus 2440000).

D2 the modified Julian date of arrival.

H1 the initial hyperbolic excess velocity relative to the departure planet (if NV1 = 1) in km/sec.

H2 the arrival hyperbolic excess velocity relative to the arrival planet (if NV2 = 1) in km/sec.

NPOW the type of power supply

- 0 for constant
- 1 for variable

Eight additional sets of elements are contained in subroutine EPHEM. Hence, NP1 and NP2 are permitted values up to 16 (see Table 1 for the list of elements available).

NT a control which allows tracking from a previously computed trajectory

0 no tracking, uses standard starting solution

1 tracking

Constrained Thrust Mode Input

The call list to CTMODE to predict constrained thrust performance is

```
CALL CTMODE(WMU,C,ALPHAW,CONE(4),NCONE,NCONOP,NMUOP,NCOP)
```

The arguments of the subroutine are:

WMU the powerplant mass fraction (non-negative, floating point).

C the exhaust velocity (km/sec) (floating point).

ALPHAW the powerplant specific mass (kg/kw)

CONE a four vector of cone angles (deg.) (between 0 and 180).

NCONE an integer from 1 to 4 giving the number of permissible cone angles.

NCONOP an integer with values 0, 1, or 2.

0 all cone angles are permissible and NCONE and CONE are ignored.

1 the first NCONE elements of CONE are the only permissible cone angles.

2 the program automatically chooses the best NCONE cone angles and subsequently places them in CONE.

NMUOP an integer with values 0 or 1.

0 WMU is assumed to be input.

1 WMU is internally optimized and returned as output. The initial value of WMU is irrelevant in this case.

NCOP an integer with values 0 or 1.

0 C is input

1 C is optimized and returned as output.

All four combinations of NCOP and NMUOP are possible.

Optional COMMON Parameters

Most of these options have been inserted to make CHEBYTOP II more compatible with QUICKTOP (see Reference 1). They have been given the form of COMMON statements since they are infrequently adjusted parameters which would unduly lengthen the CALL lists.

COMMON/ARRAY/TILT,PLOSS

TILT The angle is degrees the plane of the solar panels make with the spacecraft-sun vector. Default value is 90°.

PLOSS a factor to cause a time variation in the power, p.

$$p = p_o \, e^{-(PLOSS)t}$$

where p_o is the initial power at 1 au and t is in years. Default value is 0.

COMMON/INERTS/TANKS

TANKS a factor which assigns tankage a percentage of the propellant weight (for 10% tankage factor input TANKS = .1). Default value is 0.

COMMON/THRUST/BB,DD

BB and DD are parameters in the expression for thruster efficiency, η, as a function of exhaust velocity. Default values are BB = 1, DD = 0.

$$\eta = \frac{BB}{1 + \frac{DD^2}{C}}$$

COMMON/ELEMNT/ELEM(7)

ELEM(7) An array of seven orbital elements to be inserted into the 14th position of the ephemeris array (see Table 1)(radians).

COMMON/ESTUFF/ANGL,AYOU

This common block is used to modify the ephemeris array of Table 1 for the purpose of computing out-of-the-ecliptic probes (element set 11) or solar probes (element set 13).

ANGL the orbital inclination of element set 11 (degrees)

AYOU the semi-major axis of element set 13(au's)

COMMON/BDYP/BDY(3,4,2),PV,PC,BT,PMT

This, and the following statement, are output common blocks which makes certain performance data available to a master program.

BDY 2 3 x 4 x 2 array containing the vectors, x, $\dot{x}$, a, $\dot{a}$ at each endpoint (e.g., BDY(M,3,1) = a(M),M = 1,3). BDY is computed for the variable thrust solution only.

PV the variable thrust value of $\int_0^T a^2 \frac{p_o}{p} dt$ in au^2/yr^3

PC its constant thrust equivalent

BT the burn time of the thrusters (constrained thrust mode) in years

PMF the payload mass fraction (constrained thrust mode)

COMMON/PANDR/VEL1,PRAT1,PRAT2,RRR1,RRR2,VEL2,D1,D2

VEL1 the magnitude of the excess velocity at the departure planet (km/sec)

PRAT1 the power ratio, p/p_o, at departure

PRAT2 the power ratio, p/p_o, at arrival

RRR1 the radius to the sun at departure (au)

RRR2 the radius to the sun at arrival (au)

VEL2 the magnitude of the excess velocity at the arrival planet (km/sec)

D1 the modified Julian date of departure

D2 the modified Julian date of arrival

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 plant) 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 minimum number of solar revolutions specified by RN in the call list. Subsequent iterations will not in general change this basic revolution number. The user is advised to either refer to planetary ephemerides, or to carefully compare input Julian dates with those of similar solutions he already has on hand, to avoid slip ups in setting the revolution counter, RN. In the earlier versions of CHEBYTOP the argument RN was an integer, therefore the starting solution was either less than 360° or greater than 360° (or less than/greater than 720°, etc.). Problems arose when the departure and arrival dates selected by the analyst resulted in travel angles close to 360°. If the actual travel angle was just over 360° but RN was set to zero (zero meant less than 360°) the resulting trajectory would have a ridiculously short travel angle. The only way to avoid the problem was trial and error or, heaven forbid, to look up the geometry in an ephemeris. The new floating point system solves the problem for the analyst without handy ephemerides by allowing him to input .5 for the near 360° example, then the program will restrict itself to answers between π and 3π. If the tracking option (NT = 1) is used the counter RN 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.

The nature of the boundary conditions must be input through NB1,NB2 and NV1,NV2. The departure and arrival positions (x, y, z) are determined by the input Julian dates, D1 and D2. The arguments NB1 and NB2 allow the analyst the option of optimizing the departure and/or arrival

positions. Therefore, if a 45° out-of-the ecliptic probe trajectory is desired, the user can set NB1 = 2 and NB2 = 0. The result will be a trajectory which leaves the earth's orbit at the date specified and achieves the desired 45° orbit with the optimum travel angle with the travel time specified by the input Julian dates. Travel times are always constant and equal to the difference between the input values of D1 and D2.

The user may also select the option NB1 = NB2 = 0 for the above example. In that case the result will be an optimized earth departure date and arrival position. Note that in either case the output values of the Julian date reflect orbital position only, and not real time, i.e., their difference is not trip time. The only NB options that give real time outputs are 1,1 and 2,2.

Some care must be exercised when using the optimum departure/arrival options (NB1/NB2). There do exist local optimum solutions, such as the ones shown in Figure 2, which the user may not be seeking but which will adequately satisfy the program's convergence criteria. Optimum Mars departure date searchs starting at points 1 and 2 in Figure 2 both converged to the local optimum at Julian Day 2444460. Very slow convergence is another danger when using the optimum departure/arrival option. In the out-of-the-ecliptic probe case mentioned above for example, the derivative of the payoff, J, with respect to departure and arrival dates (the NB1 = NB2 = 0 option) is very small. In other words, the mission cost is not strongly dependent on departure time, hence, successive iterations make small changes in J and, when the first guess is a long way from the optimum, many iterations will be used. The user might be ejected from the machine for exceeding maximum time or get a FOLLOWING TRAJECTORY FAILED TO CONVERGE comment in the printout. If either of these stops occur the case should be examined. The user may decide to accept the unconverged answer, if not, an improved first guess must be input.

The type of velocity boundary condition is set by NV1 and NV2. If these constants are set equal to either 0 or 2 any values input for H1 and H2 will be ignored. Similarly, if a rendezvous result is called for, it is

MARS TO EARTH

OPPOSITION [244 4294.8
25.3 FEB 1980

ZERO HYPERBOLIC EXCESS SPEEDS

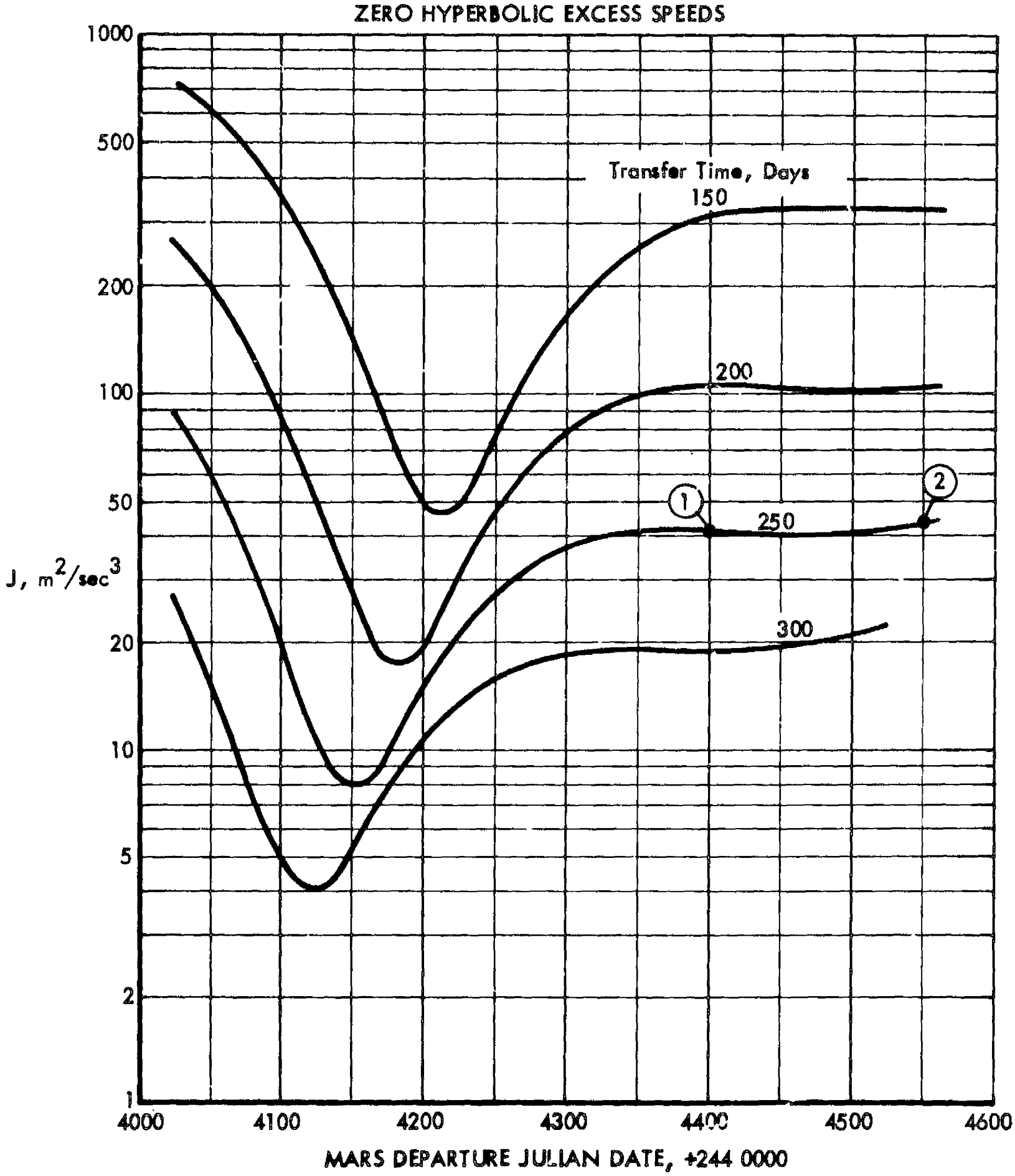


Figure 2 – EXAMPLE OF LOCAL OPTIMUM PITFALLS

not sufficient to merely set H1 and H2 equal to zero. The settings on NV1 and NV2 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.

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 4. The procedure for altering this profile is discussed in the following section.

OUTPUT

There are two forms of printed output available at the user's option, one for a description of the variable thrust trajectory and one for the constrained thrust mode. The variable thrust output is produced by following a call to VTMODE with

CALL VTOUT

Similarly the constrained thrust printout is obtained by following a CTMODE call with

CALL CTOUT

Note that neither of the output routines has a call list. If neither subroutine is called there will be no printed output. Samples of both forms of output are given in Appendix C.

The variable thrust output contains the case number, the input call lists, the order of polynomial fit used, the number of iterations needed and a selected list of output parameters. Following these is a trajectory time history. The times are given at unequal intervals which depend on the radius from the sun.

The following is a dictionary of output terms:

JV	the performance index (m^2/sec^3)
H1	excess velocity at departure (km/sec)
H2	excess velocity at arrival (km/sec)
D1	departure date(modified Julian day number). This date is different from the input D1 when NB1 is 0 or 1.

D2	arrival date (modified Julian day number). This date is different from the input D2 when BN2 is 0 or 1.
AL1,BE1	angles giving the direction of the departure excess velocity (degrees) measured as THETA and PHI below.
AL2,BE2	angles giving the direction of the arrival excess velocity (degrees).
TIME	tabulated at 26 points in days from departure.
X,Y,Z	heliocentric ecliptic cartesian coordinates of 1950.0 (au's).
XD,YD,ZD	coordinate velocities (au's/year)
R	distance from the sun (au's)
THETA	heliocentric longitude measured in the ecliptic plane from the positive X-axis (degrees).
PHI	heliocentric latitude measured from the ecliptic plane (degrees).
CONE	angle of the thrust vector measured from the spacecraft-sun line (degrees) see Figure 1.
CLOCK	angle of the thrust vector measured from a reference direction in a plane perpendicular to the spacecraft-sun line (degrees) see Figure 1.
XZEE	the angle by which the clock angle reference deviates from the normal to the ecliptic plane (degrees).
MAG.A	the magnitude of the thrust acceleration vector (au/yr^2)
P/PO	power level relative to that at 1 au.
NL	the number of patched polynomials used in the solution (maximum of 6)
NIT	the number of iterations used in the solution.

The constrained thrust output, as shown in Appendix C, needs no further explanation. Those output terms not contained in the above dictionary of terms are defined under Input call list description.

SUBROUTINE DESCRIPTIONS

This section contains brief descriptions of the program subroutines. The index below is an alphabetical listing giving page locations.

INDEX OF SUBROUTINES

*There are function subprograms

ALIGN: ALIGN serves two functions. It constructs the boundary elements of the vector X using the array of boundary parameters (AL) and specified hyperbolic excess speeds (H1, H2). To do this ALIGN calls VCAL which actually performs the necessary calculations.

Secondly, ALIGN 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.

AMAP: AMAP calculates $\emptyset$ defined by Equation (14d) of Appendix D at time H. In fact this value is returned as AMAP. However, other quantities are also calculated at time H, namely the cone (ACONE) and clock (ACLOCK) angles of the constrained thrust acceleration as well as the reference angle ZETA. (See Figure 1.) In addition the quantities (a_v, c_v) and $|a_v - (a_v, c_v)\ c_v|$ AB and ABP respectively, as defined in (14d) of Appendix D are also computed.

The loop 35 performs the actual selection of an optimal cone angle.

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.

BAKSUB: This routine solves the equation AX = B for X once the coefficient matrix A has been square-rooted by SQROOT. (See Reference 7 for a brief discussion of the square root method of solving a set of linear equations.) The loop 40 comprises the forward substitution and the loop 92 performs the backward substitution.

COEFF: This routine computes the coefficients of an 8th order polynomial approximating J. The coefficients are stored in the array CO for later use in a one-dimensional search.

CONOP: This subroutine computes optimum cone angles as explained in Appendix D, Part III. Inputs to this routine are outputs of the unconstrained cone angle case. STP is a vector of length NST containing normalized switch times. BoT is the normalized value of a_o/c, PHE the vector of length NNN containing $|a_v(\tau)|$, and ACONE the vector of length NNN containing the cone angle of a_v. PHE1 and PHE2 are vectors of length NNN containing (a_v,c_v), and $|a_v - (a_v,c_v)c_v|$, respectively. Finally NCONE is the number of desired optimal cone angles, and CONE a vector in which they are to be stored.

The following are internal quantities defined by CONOP. PHE5, PHE6, and PHE7 and $\frac{\sigma}{\mu} \cdot$ PHE, $\frac{\sigma}{\mu} \cdot$ PHE1, and $\frac{\sigma}{\mu} \cdot$ PHE2, respectively. PHE3 and PHE4 are $\int_{\tau_o}^{\tau} (a_v,c_v) \frac{\sigma}{\mu} d\tau$ and $\int_{\tau o}^{\tau} |a_v-(a_v,c_v)c_v| \frac{\sigma}{\mu} d\tau$, respectively.

P is the value of the right side of Equation (14b) (without the factor a_o), G1 and G2 vectors containing the cosine and sine of the cone angles, respectively, S1 and S2 vectors whose j^{th} element contains

$$\int_{\tau}^{\tau_o} (a_v, c_v) \frac{\sigma}{\mu} d\tau \text{ and } \int_{\tau_o}^{\tau} |a_v - (a_v, c_v) c_v| \frac{\sigma}{\mu} d\tau$$

where the integrals are evaluated only over that portion of the trajectory where the j^{th} cone angle is selected by (14d).

The loop 99 interates on cone angles to find the set maximizing P. The loop 89 updates the cone angles after each iteration and replaces "lost" cone angles with the cone angle of a_v at the most advantagous place (IMAX).

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.

CONST is called only once regardless of the number of cases run.

COPTR: This routine provides the logic for optimizing payload with respect to exhaust velocity assuming fixed powerplant mass fraction. COPTR repeatedly calls ROOTR with various values of c using Golden sectioning (Page 242, Reference 6) to isolate the optimum. The loop 15 steps off c from o to ∞ until a decrease in payload occurs. The loop 90 then narrows down the resulting interval to a point where parabolic interpolation (statements 95-98) should closely approximate the best c.

The call list is passed on the ROOTR except for RO3, which is the optimal exhaust velocity calculated by COPTR. One can check ROOTR and subsequently KROOT for explanation of quantities in the list. WMU, ALPHAW, and WMU1 are also used by the payload calculation routine WMUL, and these quantities are defined there.

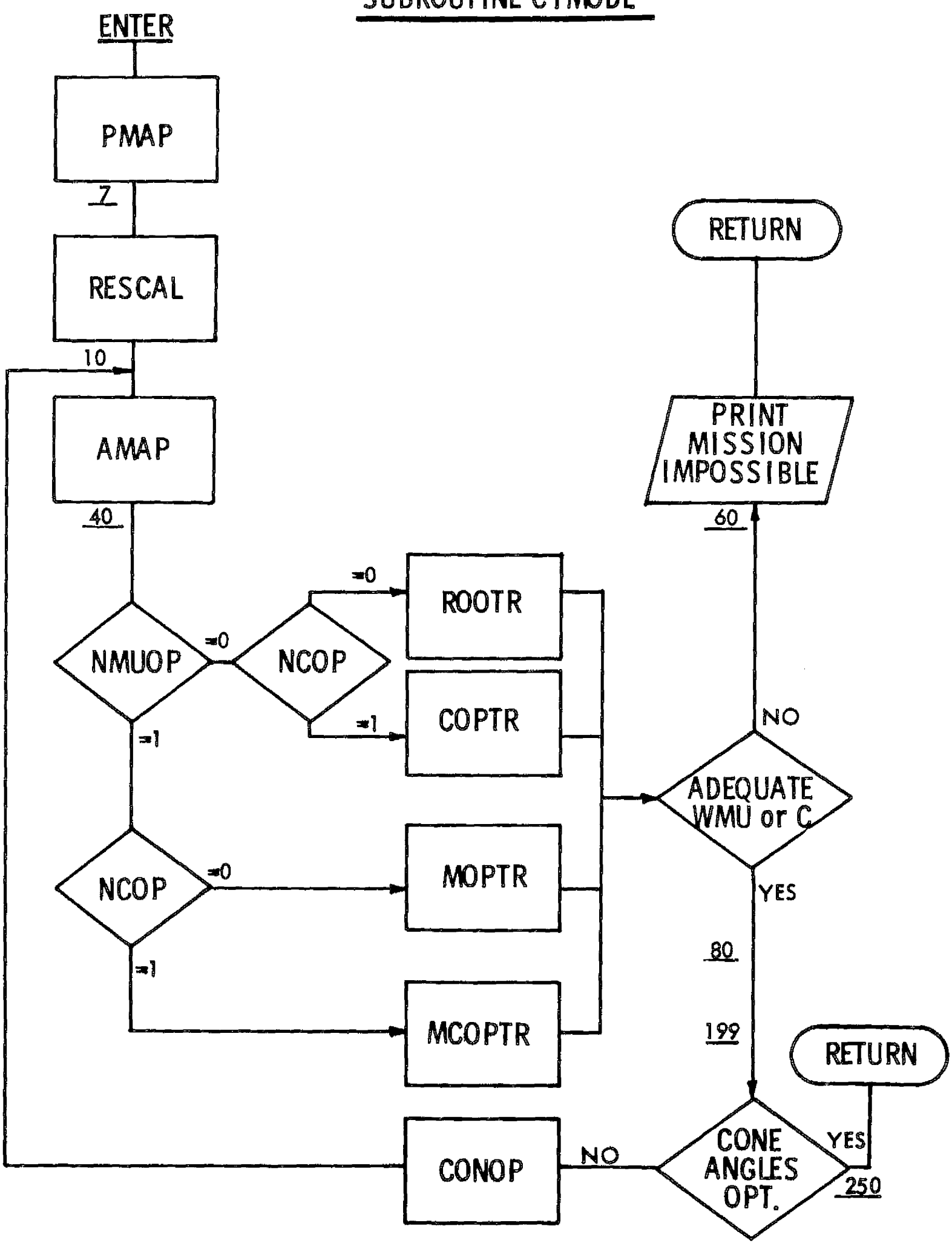
CTMODE: This is the control program for the constrained thrust prediction scheme. The call list is explained in the section on INPUT. CTMODE uses the following subroutines in order to construct the constrained thrust approximation.

PMAP	To tabulate power as a function of time
RESCAL	To construct time intervals during which equal amounts of power are consumed, thereby making possible the transformation of variables discussed in the last section of Part I, Appendix D.
AMAP	To calculate $\emptyset$ as defined by Equation (14d) of Part I, Appendix D.
ROOTR	To obtain the value of K in (14e) which leads to a solution of (14b)
COPTR, MOPTR, MCOPTR	To optimize on c, μ_w, and c and μ_w respectively as discussed in Appendix D, Part II.
CONOP	To determine optimum cone angles.

Subroutines PAMP and AMAP use results stored in COMMON by subroutine VTMODE. Hence, CTMODE operate on data stored by the most recent call to VTMODE.

CTOUT: This subroutine displays the results of the constrained thrust prediction scheme in the format shown in Appendix C. The number of time points printed is variable and is equal to 20 (NL) + 1, where NL is the final number of legs required by the variable thrust trajectory. Note that the time intervals are not necessarily equally spaced in the case of variable power.

SUBROUTINE CTMODE



DDERV1: This routine evaluates J and its directional derivative for use in the linear search routine LSERCH. DDERV1 uses the polynomial approximation to J calculated in COEFF. However, if either the first or last leg of the trajectory involves a hyperbolic excess velocity or variable Julian date, the contribution to J of this leg is computed exactly by DDERV2.

RO is the scalar parameter of the linear search, P is the total J. and PRO the derivative of J with respect to RO.

DDERV2: This routine performs the same function as DDERV1, but makes exact rather than approximate evaluations. DDERV2 calls WYDER to obtain J for each leg as well as necessary quantities to compute $\frac{dJ}{dRO}$.

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.

If NO = 2 (Gauss method) the pseudo second partial derivatives are stored symmetrically in the PXX array, with the diagonal partials stored in PDIAG as well. If NO = 3 (Newton's method) the true second partial derivatives are stored on the main diagonal and below, while the pseudo second partial derivatives are stored above the main diagonal and in PDIAG.

EPHEM: EPHEM is the subroutine which determines the heliocentric cartesain components of position and (its first three derivatives with respect to time) for the launch or arrival bodies. The integer N in the call list is the number of the desired body, and the quantity D is the Julian date (-2440000) at which its state is to be calculated. The output of EPHEM is the vector V whose components are $x, y, z, \dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z}, \dddot{x}, \dddot{y}, \dddot{z}$ in order. These units are au and yr.

TABLE 1 - ORIBITAL ELEMENTS

		Semimajor axis (au)	Eccentricity	Inclination (degrees)	Longitude of node (degrees)	Longitude of perihellion (degrees)	Mean longitude of epoch (degrees)	Epoch date (modified Julian days)
1.	MERCURY	.387099	.205627	7.00399	47.85714	76.83309	222.6217	-3065.0
2.	VENUS	.723332	.006793	3.39433	76.31972	131.0083	174.2943	-3065.0
3.	EARTH	1.000000	.016726	0.0	0.0	102.525	100.1581	-3065.0
4.	MARS	1.523691	.093368	1.84991	49.24903	335.3227	258.7673	-3065.0
5.	JUPITER	5.202803	.048435	1.30536	100.0444	13.67823	259.8311	-3065.0
6.	SATURN	9.538843	.055682	2.48991	113.3075	92.26447	280.6713	-3065.0
7.	URANUS	19.18195	.047209	.773058	73.79630	170.0108	141.3050	-3065.0
8.	NEPTUNE	30.05779	.008575	1.77375	131.3398	44.27395	216.9409	-3065.0
9.	PLUTO	39.43871	.250236	17.1699	109.8856	224.1602	181.6463	-3065.0
10.	DARREST	3.4477	.623	19.61	138.98	315.83	315.83	5230.0
11.	EXTRA	1.0	0.0	0.0	0.0	0.0	0.0	0.0
12.	CERES	2.7675	.07590	10.607	80.514	152.367	52.37098	952.5
13.	SOLAR	.1	0.0	0.0	0.0	0.0	0.0	0.0
14.	INPUT*	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15.	EROS	1.4581	.223	10.83	304.006	122.069	112.953	-9800.0
16.	ENCKE	2.2180	.847	11.95	334.189	160.170	160.170	4580.0
17.	HALLEY	17.929	.967	162.25	58.0200	169.760	169.760	6439.0

*This position filled via COMMON/ELEMENT/, see INPUT.

The matrix E contains the orbital elements of the bodies 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 lengitude of epoch (radians)
E(7,N) = Julian date at epoch (-2440000)

As originally delivered the program contains the elements given in Table 1.

The portion of the program from the loop 1 to statement 6 involves the handling of common input for special cases, e.g., out of ecliptic probe, solar probe, etc.

The loop 100 solves Kepler's equation using Newton's method, and the statements from 200 to the loop 300 compute position and velocity vectors. The loop 300 then calculates acceleration and its time derivative.

ETA: This function routine simply calculates efficiency of the thrustors (ETA) as a function of exhaust velocity (c). The formula in the subroutine listing of this document is a fairly good analytic expression for ion-propulsion engines and contains parameters BB,DD (of the common block THRUST) which can easily be adjusted. However, the user is free to substitute any function desired, even if defined by tables.

INDCAL: The purpose of this routine is the calculation of various indices used throughout the program. These particular indices depend upon the type of boundary conditions specified at each end, and the call list reflects this fact. Most of the indices are stored in the common block INDEX.

KOPTR: For fixed a_o/c KOPTR optimizes payload with respect to K of (14e), Appendix D. KOPTR is an inner loop for MCOPTR,

and one should check the explanation of that routine. KOPTR makes multiple calls to KROOT with different values of K. The optimization logic is identical to that of MOPTR and will not be discussed here.

The call list is passed on to ETA, KROOT or WMUL and is treated in these routines.

KROOT: This subroutine performs a forward integration of Equations (14 a, b, c) of Appendix D using the switching function of (14e) to determine σ. The function Ø of (14d) along with the number of fixed integration steps are the first two inputs in the call list. These are followed by the value of a_o/c (normalized if variable power is considered) and K. PV here is the output value of the right side of (14b) (divided by c), and WMU1 the final mass fraction. ST is the vector of resultant normalized switch times and NST its length. TEST is a floating point number which is positive or negative depending on whether the forward integration is considered to be valid or invalid, respectively. An invalid integration is assumed if the mass fraction goes negative or if K_{MIN} turns out to be greater than K. (See Part III of Appendix D.)

Integration in this subroutine is performed using the simple trapezoidal rule. Linear interpolation is used to find switch points.

LSERCH: This routine provides the logic for minimizing a function (P) of one variable (RO). It assumes P and the derivative of P with respect to RO (PRO) are available through the subroutine DDERIV. ROo, Po, PROo are brought in through the call list as initial values. RO,P,PRO are also brought in as an initial guess to the optimal values.

It is assumed that PROo is nonpositive. The loop 10 then steps off in the positive RO direction until a positive derivative is encountered. The loop 20 subsequently

interval halves and researches for zero derivative. If the resulting payoff P is greater than Po, LSERCH launches into the Golden sectioning loop 90. Golden sectioning (Reference 6) is a linear search algorithm which uses only function values and not derivatives to isolate a minimum.

MCOPTR: This routine is called when both exhaust velocity, c, and powerplant mass fraction, μ_w, are to be optimized for maximum payload. In accordance with the discussion of Part III, Appendix D. two equivalent parameters a_o/c and K are substituted for c and μ_w. (Acutally a normalized value of a_o/c is used in the case of variable power.) MCOPTR provides the logic for optimizing a_o/c in an outer loop and makes repeated calls to KOPTR, which optimizes K in an inner loop.

The operation of MCOPTR is conceptually the same as that of COPTR and is discussed in that routine's explanation.

MODIFY: MODIFY takes the solution of the SQROOT and BAKSUB routines (PX) and extracts quantities determining the actual change in the position vector (X). In the case of auxiliary parameters (hyperbolic excess velocity directions and Julian dates) the first and/or last few elements of PX are the changes in these parameters and are loaded into DAL. 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.

MOPTR: MOPTR is identical in concept to COPTR except that exhaust velocity is fixed and powerplant mass fraction optimized.

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 2. 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 be 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 2 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

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

If NS is 2, B is assumed to be symmetric and some savings result. See Reference 2 for definitions of U and 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 result.

PATCH: PATCH has two functions. In the loops 49 and 90 it modifies the derivative matrices for the first and last legs in case hyperbolic excess velocity directions or Julian dates 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 these auxiliary parameters. (See Reference 2.)

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.

PDERIV: The only purpose of PDERIV is to call WYDER and DERIV for each leg of the trajectory. In the process of doing so, PDERIV sums the payoff for each leg to form the total variable thrust J.

PMAP: PMAP furnishes CTMODE with the power level at time H. The vectors necessary to calculate this value are obtained from VTMODE through the common block OUT.

POLEVL: Roughly, POLEVL evaluates either the zero or first derivative of a polynomial with coefficients Y at a point S. The result is stored in G. The order of the derivative evaluated in NO. POLEVL is capable of evaluating three polynomials at a time, since the position, velocity, and acceleration vectors of the trajectory have three cartesian components.

POWER: The purpose of this subroutine is the calculation of the relative power level of the powerplant at heliocentric radius R and time T. Two options are available depending on the integer NPO. If NPO is o the power level is assumed to be independent of radius, whereas if NPO is 1 the power profile with respect to radius is assumed to be that of Reference 4. In either case the power may be diminished by the quantity FACTOR, which takes into account a percentage power loss per year (PLOSS), and a uniform loss (TILT). In the case of solar power the latter loss corresponds to an attitude anomaly of the solar array.

NO is an integer which specifies the desired order of derivatives of the power profile with respect to radius. If NO is o only Po, the actual power level is calculated. If NO is 1, P1 or $\frac{dPo}{dR}$ is also evaluated, and if NO is 2, P2 or $\frac{d^2Po}{dR^2}$ is calculated.

PXS: PXS calculates the quantities necessary for employment of thy modified Gauss' method (see Mathematics of Variable Thrust Solution). PXS is called for each leg of the trajectory. If auxiliary parameters are being optimized in the first and/or last legs, NA is set equal to one and additional computations are performed.

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

The patch time dependent quantities T,TACUM,T1,T2,T3, and TP are also evaluated anew.

RESCAL: This subprogram solves the differential equation

$$\frac{d\tau}{dt} = P(t),\ \tau(o) = o \qquad 0 \leq t \leq 1$$

using Simpson's rule with N equally spaced points, where N is odd and $N \leq 301$. The solution $\tau = \tau(t_i)$, $T_i = \frac{i-1}{N-1}$, $i = 1, 2, --N$ is loaded into the array TAUS. It is assumed that $P > 0$, hence the inverse function $t = t(\tau)$ exists. This function is tabulated at N equally spaced τ and loaded into TAU.

The quantity SCALE is $\int_o^1 P(t)dt$.

ROOTR: This subroutine provides the logic for finding the value of K in Equation (14e), Appendix D, such that (14b) is satisfied. The procedure is described at the beginning of Part III of Appendix D. ROOTR calls KROOT with various values of K and analyzes WJVTST, or $\frac{a_o}{c} \int_{t_o}^{t_1} \frac{\phi\sigma}{\mu} d\tau$ to determine better approximations of the root.

The loop 30 decreases K from K_{MAX} until TEST is less than zero, or WJVTST is greater than WJV, i.e., JV/c. The loop 5 then interval halves, either isolating the root or determining none exists. In the latter case ROOTR returns a negative final mass fraction whose magnitude is roughly proportional to the ratio of the left side of (14b) over the right. This insures correct functioning of outer search loops which call ROOTR.

Many quantities of the call list of ROOTR are simply passed on to KROOT and one can check KROOT for their definition. PV is the variable thrust J and PHEMAX is the maximum element of PHE. The remaining quantities are used to compute BoT, the normalized value of a_o/c.

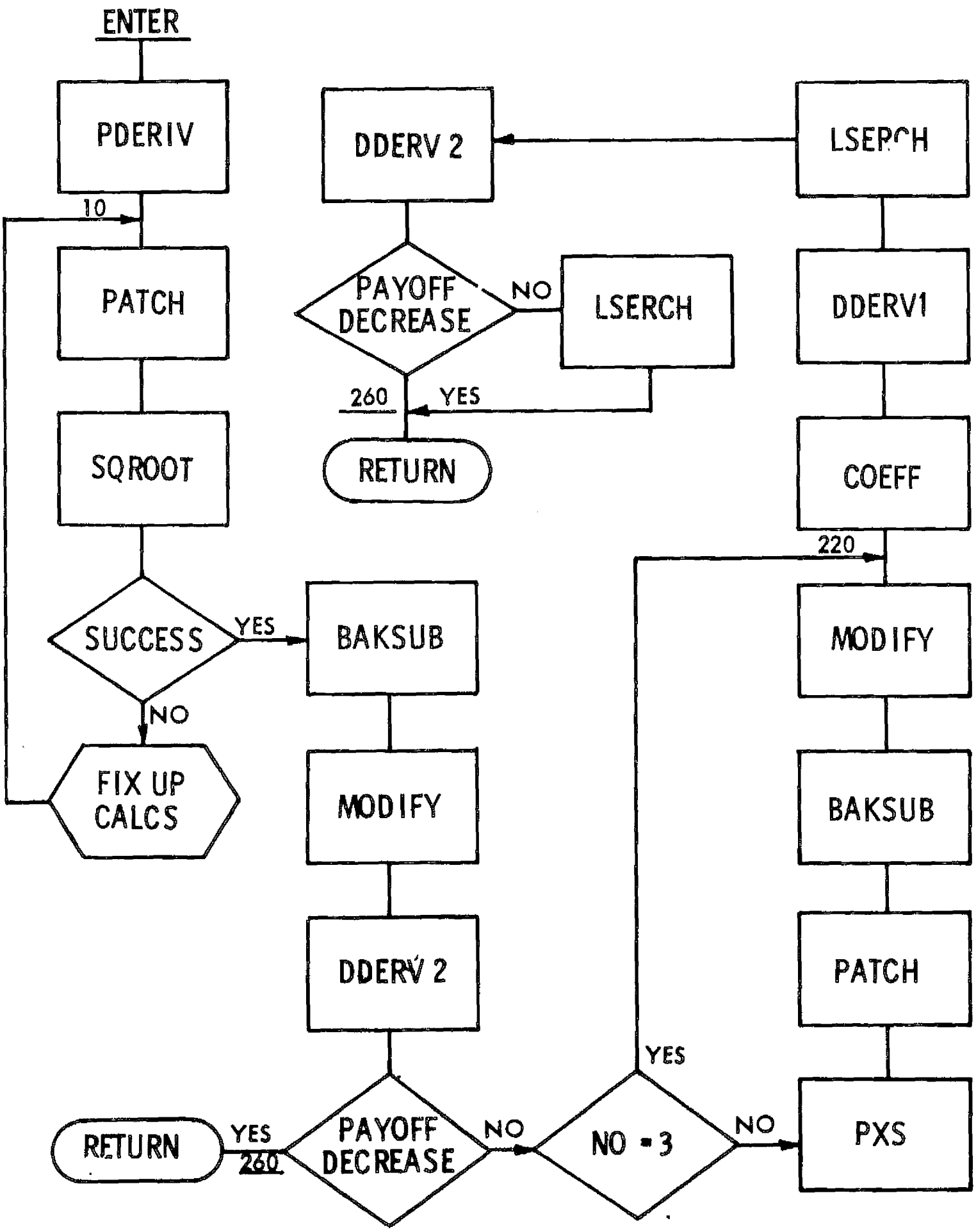
SEARCH: SEARCH executes an iteration as follows. PDERIV is called to calculate the payoff and first and (possibly pseudo) second partial derivatives of the payoff. PATCH modifies these partials to enable optimizing auxiliary parameters (hyperbolic excess velocity directions and Julian dates) if any, and to account for patching of each leg of the trajectory. SQROOT then extracts the square root of the second partial matrix (PXX).

If a negative square root is encountered (only possible when NO = 3 or NH = 3), PXX is appropriately modified (NO = 2 or NH = 2) and SQROOT is called again.

Upon successful completion of SQROOT, BAKSUB is called to complete the solution to the set of linear equations (see Mathematics of Variable Thrust Solution). Note that the solution is loaded into the input array PX. MODIFY then takes the solution delivered by BAKSUB and converts it to actual changes in trajectory parameters which are loaded into DX1 and DAL1.

DDERV2 is called to see if these changes result in a trajectory with smaller J. If so a RETURN is executed. If not a linear search is performed as follows. The loop 180 calls PXS for each leg to calculate the quantities necessary for execution of a second order Gauss iteration. (See Mathematics of the Variable Thrust Solution.) PATCH, BAKSUB, and MODIFY are called to complete the computation of the second order changes in the trajectory.

SUBROUTINE SEARCH

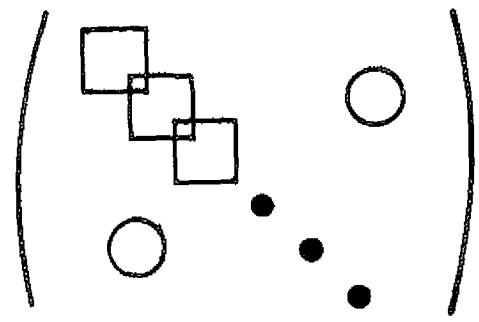


At this point COEFF computes a polynomial approximation to J for the first and second order changes just calculated LSERCH minimizes this polynomial. If the resultant payoff is larger than the original, it is assumed that the polynomial approximation is invalid. LSERCH is again called, and this time J is computed exactly.

The trajectory is updated and a RETURN executed.

SQROOT: SQROOT takes the square root of a matrix A which is symmetric and positive definite. (See Reference 7.)

In our case A is the matrix of (psuedo) 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. This is done in the loop 45.

In the case NB1 = 1 and NB2 = 1, the last row and column are nonzero due to the fact launch and arrival dates are coupled. The portion of the program between statements 45 and 78 account for this anomaly.

At the beginning of the subroutine NSUCC is set o. Upon successful completion of the routine NSUCC is set equal to 1. If the routine encounters a negative square root (only

possible when NO = 3 or NH = 3, and A is not positive definite) a return is executed with NSUCC still 0 - 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 to 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.

TSHIFT: This routine computes the function $P(\tau) = r(\tau)^{-\lambda}$ where r is the heliocentric radius of the spacecraft. Hence $\lambda = 3/2$ if $r < 1$ and $\lambda = 1$ if $r < 1$. TSHIFT then calls RESCAL to tabulate times corresponding to equal intervals of $\int_0^t P(\tau)d\tau$. The trip time is then divided into NLP parts each possessing equal portions of the above integral.

These become the new leg times and are fed into REDIST which redistributes the legs of the trajectory accordingly.

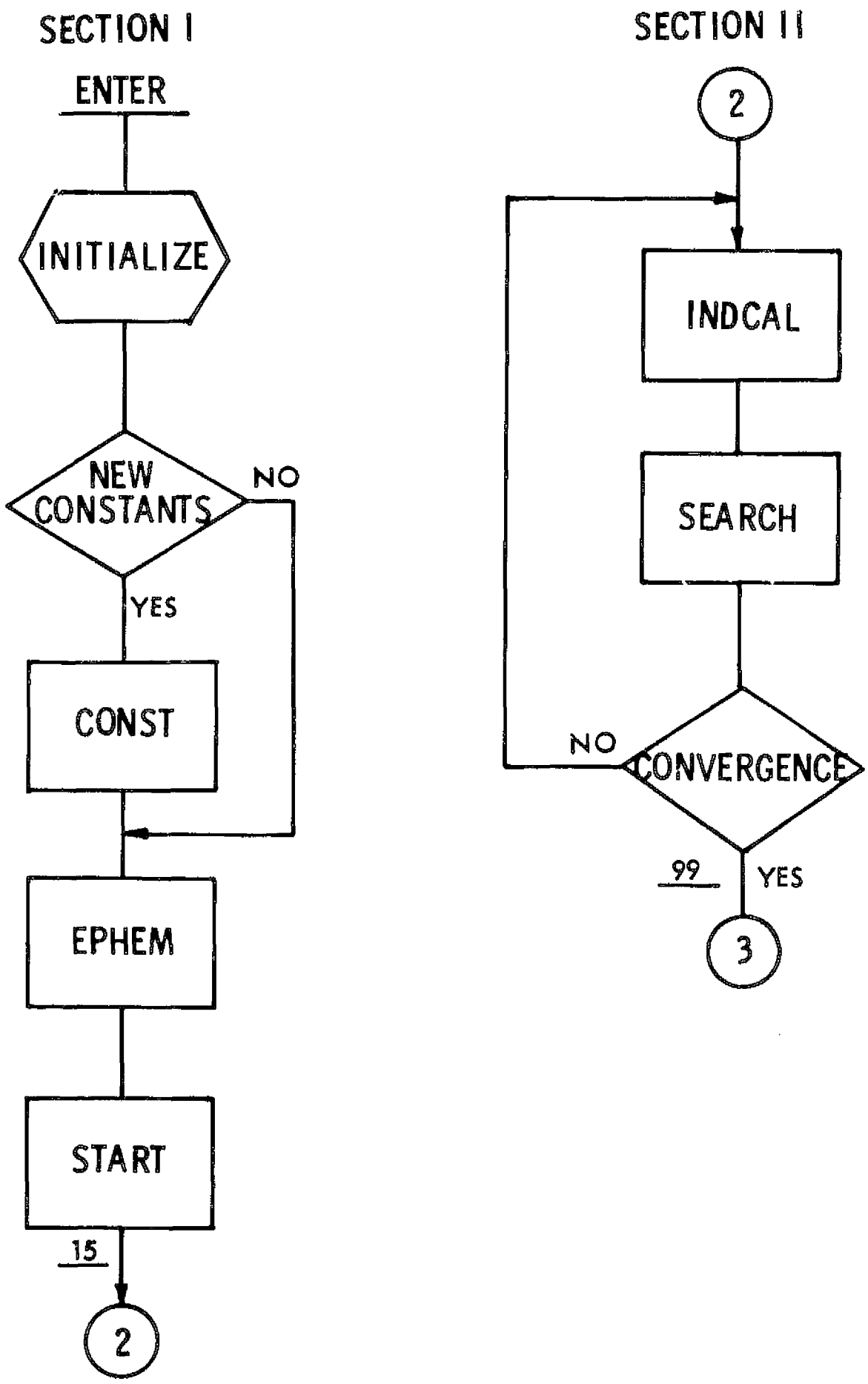
VCAL: This routine computes the heliocentric position and velocity of the spacecraft at either launch or arrival. In addition VCAL computes the first two derivatives of these quantities with respect to the vector AL containing the Julian date and hyperbolic excess velocity pointing angles. The zero, first, and second derivatives are stored in VX,VXA and VXAA respectively. The first three components of VX are position and the last three are velocity.

VTMODE: VTMODE is the control subroutine of the variable thrust portion of CHEBYTOP. As such VTMODE is primarily a bookkeeping program. It keeps track of the progress of the iterations, redistributes trajectory legs, converts from two dimension to three, computes output, provides logic for tracking, etc. The call list is discussed under INPUT. The program can be divided into four sections. The first extends to the loop 99. The second consists of this loop. The third extends from statement 100 to 950, and the last from 950 to the end. 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 dete' 'e the state of the launch and arrival planets, and finally START is called to determine a starting trajectory or tracking parameters.

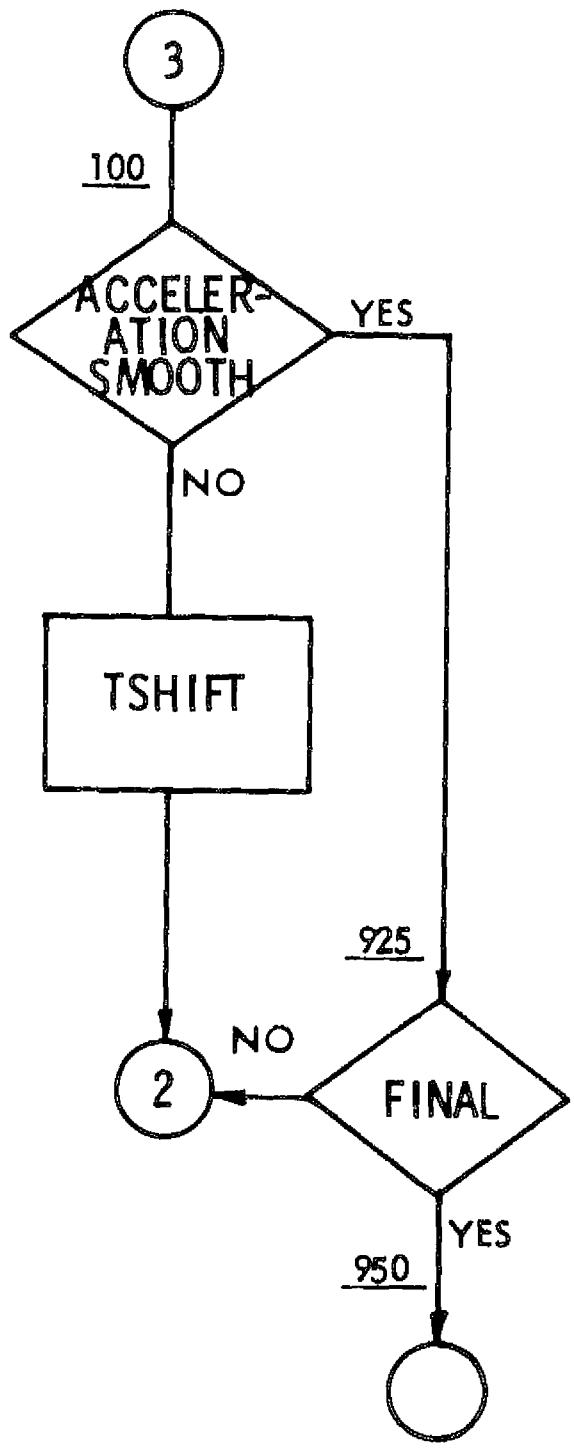
Section 2: This section is the iteration loop minimizing J. The logic for each iteration is actually

SUBROUTINE VTMODE

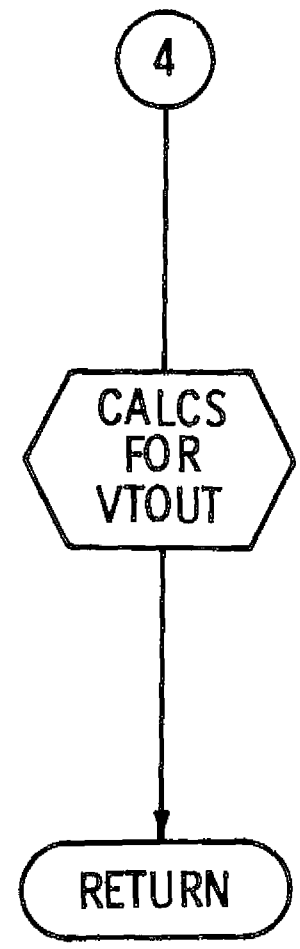


SUBROUTINE VTMODE (CONT.)

SECTION III



SECTION IV



contained in the roucine SEARCH. However, VTMODE determines which kind of iteration to take and when convergence is achieved. For example, VTMODE requires that a Newton iteration succeed before a trajectory is considered to be optimum.

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, VTOUT, and PMAP and AMAP.

VTOUT: VTOUT calculates the prints quantities describing the variable thrust trajectory. The print format is displayed in Appendix C. The trajectory quantities are calculated at twenty-six representative (not necessarily equally spaced) times.

WMUL: This function routine calculates payload (WMUL) as a function of powerplant mass fraction (WMU), exhaust velocity (c), powerplant specific mass (ALPHAW), and final mass fraction (WMU1). The formula in the subroutine listing is a relatively simple one. It implies that payload is final mass minus powerplant mass (except for a tank factor which is input through common). The user is free to replace this formula with one of his own devising.

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 (AV) and the power vector (WV), and thirdly it calculates quantities which are used in DERIV, namely O,Q,G,GD, and H. The integer NORDER in the call list indicates (roughly) the

order of derivatives desired, and not all calculations ar:
actually performed if NORDER is less than three.

V. PROGRAM LIMITATIONS

Generally, trajectories should not be attempted which require more than three revolutions about the sun. The order of polynomial representation is not sufficient, in most cases, to maintain accuracy beyond three revs. Although one of the sample cases in Appendix C, the Mercury case, does reach 1265° of in-plane transfer angle it must be regarded as an exception.

It is possible to increase the order of representation by adding more polynomial "legs." In fact, successful trial runs have been made with up to 10 legs of ninth order each. CHEBYTOP II, however, has been limited to six ninth order legs because that is a sufficient number for most applications and because additional legs significantly increase computer storage requirements.

VI. REFERENCES

1. Mascy, A. C., "QUICKTOP, A User Oriented Multi-Option Computer Program for Quick Trajectory and Mass Optimization of Electric Propulsion Missions," OART/Advanced Concepts and Missions Division Working Paper MS-71-1, June 1971.

2. Johnson, F. T., "Approximate Finite-Thrust Trajectory Optimization," AIAA Journal, Vol. 7, No. 6, June 1969.

3. 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.

4. Ross, R. E., et al., "Measured Performance of Silicon Solar Cell Assemblies Designed for Use at High Solar Intensity," JPL CM 33-473, March 1971.

5. Berkovitz, Leonard D., "Variational Methods in Problems of Control and Programming," Journal of Mathematical Analysis and Applications 3, 145-169 (1961).

6. Wilde, D. J., and Beightler, C. S., Foundations of Optimization, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1967.

7. Fadeeva, V. N., Computational Methods of Linear Algebra, Dover Publications, Inc., N.Y., 1959.

APPENDIX A
DICTIONARY OF TERMS

A:	Equation 22, Reference 2.
AA:	Same as C(k), Equation 25, Reference 2
AB:	Page 24
ABP:	Page 24
ACLOCK:	Page 24
ACONE:	Page 24
AL:	Page 24
AL1:	Page 22
AL2:	Page 22
AMAG:	Magnitude of the constrained thrust acceleration.
ALPHAP:	Internal variable equal to ALPHAW
ALPHAW:	Page 15
ANGL:	Page 16
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.
AYOU:	Page 16
BoT:	Page 25
BA:	Equation 23, Reference 2.
BB:	Page 16
BDY:	Page 17
BE1:	Page 22
BE2:	Page 22
BT:	Page 17
C:	Same as F, Equation 16, Reference 2, when used in the variable thrust portion of the program. See Page 15 for use in the constrained thrust part of the program.
CHEB:	A singly subscripted array containing the Chebychev points of [0, 1] of order NP.
CLOCK:	Page 22
CO:	Page 25
CONE:	Page 15

CONEP: Internal array set equal to CONE.

CONESC: A doubly subscripted array which is filled with the sine and cosine of the cone angles in CONE.

CP: Internal variable set equal to C.

CT: Exhaust velocity in au/yr. divided by T.

CVRG1: A convergence test which roughly checks to see if the payoff no longer decreases.

CVRG2: A convergence test which checks to see if a Newton's iteration leads to a decrease in payoff without the aid of a linear search.

CVRG3: A convergence test which basically checks to see if a Newton's iteration was performed.

CVRG4: A convergence test which checks to see if the payoff has settled down enough to try a Newton's iteration.

D: Either Equation 16, Reference 2, or Page 29.

D1: Page 14

D2: Page 14

DA1: Equal to D1

DA2: Equal to D2

DAL1: Page 38

DAL2: Page 38

DD: Page 16

DD1: Equal to D1

DD2: Equal to D2

DX1: Page 38

DX2: Page 38

E: Either Equation 16, Reference 2, or Page 31

ETA: Page 31

ETAP: Efficiency of the thrusters (ETA(C)).

ELEM: Page 16

F: Equal to $W^T FW$, Page 8.

GRAV: Square root of the gravitational constant of the sun in au, yr. units.

H1: Page 14

H2: Page 14

HH1: Equal to H1

HH2: Equal to H2

HV1: Equal to H1

HV2: Equal to H2

JV: Page 26

MAG A: Page 22

N: NP/2

NA: Page 36

NB1: Page 14

NB2: Page 14

NBD1: Equal to NB1

NBD2: Equal to NB2

NBFIX:
- 0 if auxiliary parameters free (hyperbolic excess velocity directions and Julian dates)
- 1 if auxiliary parameters fixed.

NC: Page 13

NCONE: Page 15

NCONEP: Equal to 0 before cone angles optimized, and equal to NCONE afterwards.

NCONOP: Page 15

NCONP: Equal to NCONOP

NCOP: Page 15

NCOPP: Equal to NCOP

NCOUNT: Counts number of VTMODE calls.

ND: Number of dimensions of trajectory X.

NDER12: Tells DDERV2 that it is being called from DDERV1.

NF: Convergence parameter which roughly keeps track of how many times TSHIFT has been called.

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 vectory 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.

NHFIX: Determines whether NH is set equal to 2 or 3.

NIT: Maximum number of iterations allowed for achieving an optimum on any given step.

NL: Page 22

NLP: Equal to NL

NMUOP: Page 15

NMUOPP: Internal variable set equal to NMUOP

NNN: Page 25

NO: Page 29

NP: Page 26

NP1: Page 14

NP2: Page 14

NPL1: Internal parameter set equal to NP1

NPL2: Internal parameter set equal to NP2

NPO: Internal parameter set equal to NPOW

NPOW: Page 14

NNN: Page 25

NST: Page 25

NSUCC: Page 40

NT: Page 15

NTT: Internal parameter set equal NT, unless tracking breaks down, in which case it is set to 0.

NV1: Page 14

NV2: Page 14

NVL1: Internal parameter set equal to NV1

NVL2: Internal parameter set equal to NV2

P: Variable thrust payoff in au^2/yr^3

PC: Page 17

PCON: Multiplier which converts from au^2/yr^3 to m^2/sec^3.

PCP: Internal variable set equal to PC.

PDIAG: Used initially in SEARCH to save the diagonal elements of PXX. Used later in SEARCH to store auxiliary parameter derivatives (if NH=3) when applying the second order version of Gauss' method.

PHE: Page 25

PHE1: Page 25

PHE2: Page 25

PHE3: Page 25

PHE4: Page 25

PHE5: Page 25

PHE6: Page 25

PHE7: Page 25

PHEMAX: Page 38

PHI: Page 22

PI: π

PI2: 2π

PIF: $180/\pi$

PLOSS: Page 16

PMF: Page 17

PMFP: Internal variable set equal to PMF

Po: Initial value of payoff

POWCON: Multiplier which converts from yr^3/au^2 to kg/kw

PRAT: Page 17

PRAT2: Page 17

PS: Variable thrust payoff

PV: Page 17 or Page 32

PVTEST: Equal to PV, Page 32.

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.

PXSAV: Used initially in SEARCH to save the array PX. Used later in SEARCH to store auxiliary parameter derivatives when applying the second order version of Gauss' method.

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.

R: Page 22

RN: Page 13

RRR1: Page 17

RRR2: Page 17

SCALE: Page 37

ST: Page 32

STP: Page 25

T: In the variable thrust portion of the program T = T(LN) is a singly subscripted array representing the duration in years of the LN^{th} leg of the trajectory. In the constrained thrust portion of the program,

$$T = \int_0^{TT} \frac{P(t)}{Po}\, dt ,$$

P(t) the power level of the powerplant

T1: $T1(LN) = T(LN)^{-1}$

T2: $T2(LN) = T(LN)^{-2}$

T3: $(T3(LN) = T(LN)^{-3}$

TACUM: TACUM(LN) = Accumulated trip time to beginning of LN^{th} leg, with TACUM(NL+1) = TT.

TANKS: Page 16

TAU: Page 37

TAUS: Page 37

TAVSTR: $\frac{1}{TT} \int_0^{TT} \frac{P(t)}{Po}\, dt$, p(t) the power level of the powerplant.

TB: Ratio of thrust-on time to trip time

TBF: Same as TB, but for ficticious time. (See Appendix D, end of Part 1).

TCON: Multiplier which converts from years to days.

TEST: In the variable thrust portion of the program, TEST in the relative change in payoff from one iteration to the next. See Page 32 for its use in the constrained thrust part of the program.

THETA: Page 22

TILT: Page 16

TIME: Page 22

TP: $TP(LN) = (GRAV \cdot T(LN))^2$

TT: Total trip time in years.

TTDAY: Total trip time in days.

TTP: Trip time, in case tracking on trip time is employed.

V1: Position, velocity, acceleration and derivative of acceleration of launch planet (See EPHEM subroutine for details.)

V2: Same as V1 for arrival planet

VEL1: Page 17

VEL2: Page 17

VELCØW: Multiplier which converts from km./sec. to au./yr.

VX: Page 42

VXA: Page 42

VXAA: Page 42

WJVTST: Page 37

WMU: Page 15

WMU1: Page 32

WMUL: Page 45

WMU1P: Equal to WMU1

WMUP: Internal variable equal to WMU

WV: WV is a triply subscripted array (WV(I, M, LN)). WV(I,1,LN) is equal to $\sqrt{po/p}$ (p= power level) at the I^{th} Chebychev point of the LN^{th} leg. WV(I, 2, LN) is roughly $\frac{\partial}{\partial R}$ WV(I,2,LN) and WV(I, 3, LN) is roughly $\frac{\partial}{\partial R}$ WV(I, 2, LN).

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.

X,Y,Z: Page 22

XD,YD,ZD: Page 22

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.

XVA: A 10x3x2 array containing the effect of first order changes in the auxiliary parameters (hyperbolic excess velocity directions and Julian dates) on the cartesian components of the trajectory. XVA is needed when the second order version of Gauss' method is employed.

X2EE: Page 22

ZETA: Page 24

APPENDIX B
OPERATING INSTRUCTIONS

CHEBYTOP II is coded to run on either the IBM 360 series computers or the CDC 6000 series computers with minor coding changes. The coding changes convert the program from double precision to single precision operation. 360 computers, with their shorter word length, necessitate the use of double precision arithmetic. A single precision deck should operate on any machine accepting standard FORTRAN IV coding, hence, there should be no difficulty with running on the IBM 7094, SRU 1108, or some others, as well as the CDC 6000 series.

Converting to single precision is accomplished by removing one card from each subroutine. The cards to be removed are designated "360 IN" in card-column 73 through 80. There are 10 function subroutines at the end of the deck which need two cards removed and one card inserted. The cards to be added are marked "360 OUT" in the listing in Appendix D. Copies of program decks obtained from NASA will probably contain both the "360 IN" and the "360 OUT" cards. The user is cautioned to remove one set or the other before running the program.

The program was validated on an IBM 360/65 using OS Release 19 MVT. The deck was compiled using Release 19 FORTRAN G and Release 19 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 170,000 bytes available core using the overlay suggested in Appendix E. Validation on the CDC 6000 series was under SCOPE 3.1 and the standard FORTRAN compiler.

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 be 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 360/65 was 1 second. 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. Two 10 point polynominals are used for the 200 day trip. A power degradation with time was applied by setting PLOSS = .05 in the common block ARRAY.

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

```
COMMON/ARRAY/TILT,PLOSS
PLOSS = .05
CALL VTMODE(2,0.,3,4,2,2,2,2,6538.,6738.,0.,0.,0,0)
CALL VTOUT
STOP
END
```

The output appears on the following page.

CASE 4

CALL LIST INPUT

NC	RN	NP1	NP2	NB1	NB2	NV1	NV2	D1	D2	H1	H2	NPO	NT
2	0.0	3	4	2	2	2	2	6538.0	6738.0	0.0	0.0	0	0

COMMON BLOCK INPUT

TILT	PLOSS	ANGL	AYOU
90.0	0.05	0.0	1.0

COMMON BLOCK INPUT

E1	E2	E3	E4	E5	E6	E7
0.0	0.0	0.0	0.0	0.0	0.0	0.0

INTERNAL PARAMETERS

NL	NIT
2	4

OUTPUT

D1	H1	AL1	BE1	D2	H2	AL2	BE2	JV
6538.0	0.0	-96.8	0.0	6738.0	0.0	119.2	0.0	5.387

TIME	X	Y	Z	R	THETA	PHI	CONE	CLOCK	X2EE	MAG. A	P/P0	XO	YO	ZO
0.0	-0.895	-0.458	0.0	1.004	-153.01	0.0	123.77	270.00	0.0	7.22	1.000	2.749	-5.622	0.0
7.18	-0.834	-0.564	0.0	1.007	-145.95	0.0	127.06	270.00	0.0	6.41	0.999	3.399	-5.366	0.0
14.36	-0.761	-0.666	0.0	1.012	-138.82	0.0	130.70	270.00	0.0	5.63	0.998	4.000	-5.019	0.0
21.55	-0.677	-0.761	0.0	1.018	-131.68	0.0	134.78	270.00	0.0	4.89	0.997	4.539	-4.592	0.0
28.73	-0.583	-0.846	0.0	1.028	-124.58	0.0	139.16	270.00	0.0	4.21	0.996	5.004	-4.098	0.0
35.91	-0.481	-0.921	0.0	1.039	-117.56	0.0	143.89	270.00	0.0	3.57	0.995	5.389	-3.551	0.0
43.09	-0.372	-0.986	0.0	1.053	-110.67	0.0	148.89	270.00	0.0	2.98	0.994	5.689	-2.968	0.0
50.28	-0.258	-1.038	0.0	1.070	-103.95	0.0	154.11	270.00	0.0	2.45	0.993	5.905	-2.363	0.0
57.46	-0.140	-1.078	0.0	1.088	-97.41	0.0	159.42	270.00	0.0	1.96	0.992	6.038	-1.749	0.0
64.64	-0.021	-1.107	0.0	1.107	-91.08	0.0	164.63	270.00	0.0	1.51	0.991	6.093	-1.148	0.0
71.82	0.099	-1.123	0.0	1.128	-84.97	0.0	169.39	270.00	0.0	1.10	0.990	6.079	-0.563	0.0
79.01	0.218	-1.128	0.0	1.149	-79.07	0.0	172.98	270.00	0.0	0.70	0.989	6.001	0.033	0.0
86.19	0.335	-1.122	0.0	1.171	-73.40	0.0	175.60	270.00	0.0	0.33	0.988	5.867	0.584	0.0
94.19	0.461	-1.103	0.0	1.195	-67.32	0.0	49.27	270.00	0.0	0.12	0.987	5.661	1.163	0.0
103.01	0.594	-1.068	0.0	1.222	-60.90	0.0	24.97	270.00	0.0	0.56	0.986	5.377	1.758	0.0
111.92	0.728	-1.019	0.0	1.247	-54.74	0.0	27.18	270.00	0.0	1.02	0.985	5.042	2.305	0.0
120.64	0.837	-0.957	0.0	1.271	-48.81	0.0	31.08	270.00	0.0	1.51	0.984	4.666	2.807	0.0
129.46	0.945	-0.893	0.0	1.294	-43.07	0.0	35.15	270.00	0.0	2.01	0.982	4.255	3.265	0.0
138.24	1.043	-0.799	0.0	1.314	-37.48	0.0	39.07	270.00	0.0	2.54	0.981	3.815	3.681	0.0
147.09	1.129	-0.706	0.0	1.332	-32.02	0.0	42.75	270.00	0.0	3.08	0.980	3.351	4.057	0.0
155.91	1.204	-0.604	0.0	1.347	-26.63	0.0	46.19	270.00	0.0	3.65	0.979	2.866	4.395	0.0
164.73	1.267	-0.494	0.0	1.359	-21.30	0.0	49.42	270.00	0.0	4.23	0.978	2.361	4.696	0.0
173.55	1.318	-0.377	0.0	1.371	-15.98	0.0	52.47	270.00	0.0	4.83	0.977	1.839	4.962	0.0
182.36	1.356	-0.255	0.0	1.380	-10.64	0.0	55.40	270.00	0.0	5.45	0.975	1.301	5.193	0.0
191.18	1.381	-0.127	0.0	1.387	-5.26	0.0	58.24	270.00	0.0	6.07	0.974	0.748	5.387	0.0
200.00	1.392	0.005	0.0	1.392	0.21	0.0	61.03	270.00	0.0	6.69	0.973	0.184	5.545	0.0

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 constrained thrust output. Note that a propellant tankage factor was inserted, thrust direction constrained to two cone angles (optimized), and exhaust velocity and powerplant mass fraction optimized.

The main program is as follows:

```
      COMMON/THRUST/BB,DD
      COMMON/INERTS/TANKS
      DIMENSION CONE(4)
      CALL VTMODE(3,0.,3,5,2,2,2,0,4140.,4940.,0.,0.,1,0)
      CALL VTOUT
      TANKS = .05
      ALPHAW = 30.
      BB = .76
      DD = 13.
      NCONE = 2
      NCONOP = 2
      NMUOP = 1
      NCOP = 1
      CALL CTMODE(WMU,C,ALPHAW,CONE,NCONE,NCONOP,NMUOP,NCOP)
      CALL CTOUT
      STOP
      END
```

CASE 1

CALL LIST INPUT

NC	MN	NP1	NP2	NB1	NB2	NV1	NV2	D1	D2	H1	H2	NPC	AT
3	0.0	3	5	2	2	2	0	4140.0	4940.0	0.0	0.0	1	0

CEPPCN BLOCK INPUT

TILT	PLOSS	ANGL	AYOU
90.0	0.0	0.0	1.0

CEPPCP BLOCK INPUT

E1	E2	E3	E4	E5	E6	E7
0.0	0.0	0.0	0.0	0.0	0.0	0.0

INTERNAL PARAMETERS

NL	NIT
3	9

OUTPUT

D1	H1	AL1	BE1	D2	H2	AL2	BE2	JV
4140.0	0.3	86.9	-1.0	4940.6	7.0	157.4	2.4	9.788

TIME	X	Y	Z	R	THETA	PHI	GAMA	CLOCK	AZEL	MAG. A	P/P0	XD	YD	ZD
0.0	1.063	-0.000	-0.000	1.003	-0.26	0.00	92.00	270.90	0.0	7.93	0.996	-0.072	0.282	0.000
13.71	0.973	0.254	-0.000	1.001	15.50	-0.00	96.39	270.52	0.0	7.85	0.999	-1.545	6.389	-0.004
27.41	0.868	0.470	-0.000	1.005	27.96	-0.01	100.72	270.32	0.0	7.54	0.994	-2.984	6.153	-0.005
41.12	0.791	0.691	-0.000	1.021	42.61	-0.02	105.08	269.46	0.0	6.93	0.974	-4.250	5.573	-0.003
54.82	0.711	0.885	-0.000	1.053	57.17	-0.03	111.63	268.81	0.0	6.22	0.935	-5.283	4.731	0.001
68.53	0.359	1.044	-0.000	1.104	71.04	-0.02	117.71	268.01	0.0	5.31	0.877	-5.977	3.752	0.007
82.24	0.120	1.108	0.000	1.173	83.62	0.00	124.91	267.01	0.0	4.38	0.835	-6.358	2.760	0.013
95.94	-0.115	1.253	0.001	1.258	95.26	0.03	130.63	265.77	0.0	3.56	0.727	-6.464	1.847	0.021
109.65	-0.358	1.337	0.002	1.385	105.33	0.07	136.21	264.14	0.0	2.84	0.650	-6.429	1.057	0.027
123.36	-0.592	1.399	0.004	1.515	119.31	0.13	143.79	262.31	0.0	1.96	0.533	-6.138	0.029	0.038
137.06	-1.191	1.307	0.007	1.769	132.34	0.22	155.61	252.45	0.0	1.28	0.423	-5.577	-0.806	0.048
150.77	-1.590	1.225	0.011	2.007	142.36	0.31	164.21	235.95	0.0	0.88	0.341	-5.035	-1.339	0.055
164.48	-1.798	1.111	0.015	2.244	150.33	0.39	168.73	264.15	0.0	0.63	0.281	-4.537	-1.680	0.061
178.18	-2.274	0.976	0.020	2.474	156.78	0.46	167.94	164.30	0.0	0.47	0.236	-4.095	-1.901	0.065
191.89	-2.567	0.827	0.025	2.697	162.14	0.53	163.25	139.28	0.0	0.36	0.201	-3.706	-2.042	0.068
205.60	-2.832	0.670	0.030	2.910	166.66	0.59	158.43	129.74	0.0	0.29	0.175	-3.363	-2.132	0.070
219.30	-3.073	0.507	0.035	3.115	170.63	0.65	152.95	122.73	0.0	0.24	0.154	-3.058	-2.187	0.071
233.02	-3.301	0.280	0.043	3.374	175.13	0.72	147.15	118.57	0.0	0.18	0.133	-2.702	-2.222	0.072
246.73	-3.532	0.140	0.053	3.632	178.74	0.82	143.44	114.73	0.0	0.13	0.110	-2.250	-2.229	0.072
260.44	-4.040	-0.101	0.064	4.039	185.36	0.91	133.34	112.07	0.0	0.09	0.093	-1.873	-2.196	0.071
274.14	-4.297	-0.737	0.075	4.395	189.34	0.98	128.36	111.32	0.0	0.07	0.081	-1.549	-2.145	0.069
287.85	-4.560	-1.332	0.085	4.744	192.96	1.05	124.67	110.37	0.0	0.05	0.072	-1.267	-2.083	0.067
301.56	-4.079	-1.932	0.095	4.866	195.89	1.11	120.67	109.76	0.0	0.03	0.065	-1.016	-2.014	0.063
315.26	-4.814	-1.629	0.104	5.083	198.69	1.17	117.59	109.24	0.0	0.02	0.060	-0.791	-1.941	0.060
328.97	-4.917	-1.914	0.113	5.278	201.27	1.22	114.87	108.86	0.0	0.01	0.056	-0.585	-1.864	0.056
342.68	-4.961	-2.188	0.121	5.451	203.68	1.27	87.65	90.18	0.0	0.00	0.052	-0.395	-1.785	0.052

PRECICTED CONSTANT ISP TRAJECTORY

COMMON BLOCK INPUT BB 0.76 CC 13.00 TANKS 0.05

EFFICIENCY (ETA) 0.571 POWERPLANT SPECIFIC MASS (ALPHAM) 50.000 KG/KW

OPTIMIZED POWERPLANT FRACTION (MUM) 0.483 OPTIMIZED EXHAUST VELOCITY (C) 22.626 KM/SEC

OPTIMIZED CONE ANGLES 138.06 104.96

CONSTANT ISP 3 (UG) 12.695 MB/SEC) THRUST DURATION 179.698 DAYS

FINAL MASS FRACTION (MU1) 0.590 PAYLOAD MASS FRACTION (MUL) 0.087

TIME	MAGNITUDE	CONE	CLOCK	TIME	MAGNITUDE	CONE	CLOCK
0.0	5.97	104.96	270.96				
3.43	5.93	104.96	270.86	123.50	4.65	138.06	262.03
6.87	5.56	104.96	270.75	129.60	4.48	138.06	260.89
10.30	5.56	104.96	270.63	136.00	4.31	138.06	259.56
13.74	5.63	104.96	270.52	142.65	4.13	138.06	257.93
17.17	5.69	104.96	270.40	150.06	3.95	138.06	256.92
20.61	5.74	104.96	270.28	157.72	3.77	138.06	255.39
24.05	5.80	104.96	270.15	166.62	3.59	138.06	249.83
27.49	5.86	104.96	270.02	174.71	3.40	138.06	245.18
30.92	5.90	104.96	269.87	184.21	0.0	138.06	238.06
34.44	5.94	104.96	269.75	194.45	0.0	138.06	229.44
37.91	5.98	104.96	269.60	205.44	0.0	138.06	216.95
41.62	6.01	104.96	269.45	217.33	0.0	138.06	200.82
44.59	6.03	104.96	269.29	230.26	0.0	138.06	181.09
48.36	6.04	104.96	269.12	244.36	0.0	138.06	161.48
51.83	6.05	104.96	268.94	259.75	0.0	138.06	145.93
55.83	6.04	104.96	268.75	276.56	0.0	138.06	136.05
59.58	6.02	104.96	268.55	294.59	0.0	138.06	130.30
63.33	6.00	104.96	268.33	313.34	0.0	138.06	125.18
67.11	5.97	104.96	268.10	337.70	0.0	138.06	121.06
71.13	5.91	104.96	267.83	362.33	0.0	138.06	118.48
75.13	5.85	104.96	267.55	387.59	0.0	138.06	116.29
79.17	5.78	138.06	267.25	413.71	0.0	138.06	114.56
83.49	5.70	138.06	266.91	452.94	0.0	138.06	113.23
87.89	5.60	138.06	266.53	484.77	0.0	138.06	112.16
92.29	5.50	138.06	266.13	530.37	0.0	138.06	111.22
97.04	5.38	138.06	265.66	575.11	0.0	138.06	110.44
101.91	5.25	138.06	265.14	624.27	0.0	104.96	109.87
106.79	5.11	138.06	264.51	678.07	0.0	104.96	109.36
111.24	4.96	138.06	263.80	736.65	0.0	104.96	108.82
117.00	4.81	138.06	263.01	800.80	0.0	104.96	90.18

Mercury Rendezvous

This case illustrates the tracking option, NT = 1. It is a 500 day, 1265° in-plane transfer angle, solar power trajectory. First a zero excess velocity departure from earth was asked for, then the departure velocity was increased in steps of .5 km/sec to an H2 of 2.5 km/sec.

The main program is:

```
CALL VTMODE(2,3.,3,1,2,2,2,2,4300.,4800.,0.,0.,1,0)
DO 5 I=1,5
HV1 = FLOAT(I)/2.
5 CALL VTMODE(2,3.,3,1,2,2,2,2,4300.,4800.,HV1,0.,1,1)
CALL VTOUT
STOP
END
```

CASE 8

CALL LIST INPUT

NC	KN	NP1	NP2	NB1	NB2	NV1	NV2	D1	D2	H1	H2	NPC	NT
2	3.0	3	1	2	2	1	2	4300.0	4000.0	2.5	0.0	1	1

COMPCA BLOCK INPUT

TILT	PLOSS	ANGL	AYCL
0.0	3.0	0.0	1.0

COMPCA BLOCK INPUT

E1	E2	E3	E4	E5	E6	E7
0.0	3.0	3.0	3.0	0.0	0.0	0.0

INTERNAL PARAMETERS

NL	NIT
0	12

OUTPUT

D1	H1	AL1	BE1	D2	H2	AL2	BE2	JV
4300.0	2.5	62.2	0.0	4000.0	0.0	120.0	0.0	5.344

TIME	X	Y	Z	R	THETA	PHI	GLNE	GLOCK	XZEE	MAGV A	P/P0	X0	Y0	Z0
0.0	6.937	0.324	0.0	0.791	166.92	0.0	75.14	90.00	0.0	0.71	1.011	-1.911	-5.498	0.0
30.35	-0.429	-0.245	0.0	0.940	194.77	0.0	103.23	90.00	0.0	1.38	1.052	2.110	-5.525	0.0
72.76	-0.926	-0.687	0.0	0.826	232.56	0.0	114.42	90.00	0.0	2.37	1.182	5.786	-2.855	0.0
109.17	-0.194	-0.749	0.0	0.716	281.36	0.0	99.96	90.00	0.0	3.09	1.371	6.835	-1.051	0.0
145.58	0.563	-0.602	0.0	0.506	333.71	0.0	88.56	90.00	0.0	3.82	1.395	-0.206	8.857	0.0
169.75	0.279	0.449	0.0	0.524	417.85	0.0	76.69	90.00	0.0	3.53	1.396	-7.828	4.867	0.0
194.51	-0.257	0.493	0.0	0.547	435.64	0.0	73.93	90.00	0.0	2.89	1.396	-8.121	-3.144	0.0
218.46	-0.578	0.146	0.0	0.588	525.62	0.0	89.14	90.00	0.0	1.87	1.396	-2.564	-7.562	0.0
255.02	-0.536	-0.306	0.0	0.665	565.52	0.0	97.35	90.00	0.0	2.19	1.396	3.700	-6.622	0.0
291.54	-0.224	-0.556	0.0	0.696	627.93	0.0	93.30	90.00	0.0	2.60	1.396	7.324	-2.297	0.0
272.94	0.165	-0.521	0.0	0.547	647.57	0.0	102.88	90.00	0.0	3.41	1.396	7.397	3.620	0.0
254.34	0.436	-0.192	0.0	0.477	656.21	0.0	92.52	90.00	0.0	3.86	1.396	2.357	8.983	0.0
344.74	0.318	0.203	0.0	0.416	700.05	0.0	87.22	90.00	0.0	2.58	1.396	-7.189	7.438	0.0
327.16	-0.138	0.342	0.0	0.416	629.36	0.0	81.41	90.00	0.0	2.02	1.396	-9.857	-2.764	0.0
343.55	-0.447	0.104	0.0	0.456	566.91	0.0	60.92	90.00	0.0	1.38	1.396	-3.114	-8.711	0.0
360.00	-0.446	0.278	0.0	0.467	541.96	0.0	100.48	90.00	0.0	1.94	1.396	4.241	-7.274	0.0
376.45	-0.119	-0.488	0.0	0.502	576.24	0.0	93.35	90.00	0.0	2.56	1.396	8.072	-1.610	0.0
391.66	0.214	-0.418	0.0	0.469	1017.11	0.0	79.61	90.00	0.0	2.31	1.396	7.182	5.040	0.0
406.21	0.367	-0.195	0.0	0.419	1057.65	0.0	91.12	90.00	0.0	1.71	1.396	2.145	9.530	0.0
420.82	0.317	0.162	0.0	0.366	1109.80	0.0	80.25	90.00	0.0	3.35	1.396	-6.494	8.865	0.0
427.42	-0.020	0.340	0.0	0.348	1173.21	0.0	36.08	90.00	0.0	1.39	1.396	-11.410	-0.318	0.0
444.13	-0.346	0.163	0.0	0.388	1144.35	0.0	91.37	90.00	0.0	1.62	1.396	-5.721	-8.731	0.0
455.73	-0.587	-0.293	0.0	0.437	1287.67	0.0	55.45	90.00	0.0	2.14	1.396	3.231	-8.490	0.0
471.17	-0.156	-0.433	0.0	0.461	1330.96	0.0	97.02	90.00	0.0	2.87	1.396	7.968	-2.789	0.0
486.58	0.175	-0.401	0.0	0.441	1411.54	0.0	89.66	90.00	0.0	2.82	1.396	7.561	4.619	0.0
560.00	0.359	-0.000	0.0	0.373	1426.17	0.0	73.65	90.00	0.0	2.67	1.396	0.470	10.462	0.0

D180-12916-1

Encke Rendezvous

The purpose of this case is to illustrate the optimal departure date option.

The main program is:

```
      CALL VTMODE(1,0.,3,16,1,1,2,2,3576.,4520.,0.,0.,1,0)
      CALL VTOUT
      STOP
      END
```

CASE 3

CALL LIST INPUT	NC	RN	NP1	NP2	NB1	NB2	NV1	NV2	D1	D2	H1	H2	NPO	NT
	1	0.0	3	14	1	1	2	2	3576.0	4520.0	0.0	0.0	1	0

COMMON BLOCK INPUT	TILT	PLOSS	ANGL	AYOU
	90.0	0.0	0.0	1.0

COMMON BLOCK INPUT	E1	E2	E3	E4	E5	E6	E7
	0.0	0.0	0.0	0.0	0.0	0.0	0.0

INTERNAL PARAMETERS	NL	NIT
	3	16

OUTPUT	D1	H1	AL1	BE1	D2	H2	AL2	BE2	JV
	3533.4	0.0	-126.2	-7.7	4477.4	0.0	-174.4	-4.0	10.833

TIME	X	Y	Z	R	THETA	PHI	CONE	CLOCK	MACH	MAG. A	P/P0	X0	Y0	I0
0.0	-0.556	0.812	0.000	0.994	124.42	0.00	70.73	278.13	0.0	4.43	1.020	-5.278	-1.974	0.300
20.18	-0.812	0.562	-0.001	0.987	145.29	-0.05	79.72	276.20	0.0	5.10	1.016	-3.845	-5.383	-0.031
40.37	-0.973	0.229	-0.003	0.999	166.75	-0.19	90.40	274.58	0.0	5.25	1.001	-1.924	-6.554	-0.053
60.55	-1.021	-0.147	-0.007	1.032	188.21	-0.36	100.94	272.84	0.0	4.93	0.960	0.145	-6.928	-0.062
80.74	-0.961	-0.523	-0.010	1.094	209.58	-0.51	111.14	270.67	0.0	4.24	0.888	1.989	-6.575	-0.055
100.92	-0.811	-0.865	-0.012	1.186	226.87	-0.60	122.88	267.54	0.0	3.27	0.793	3.360	-5.752	-0.037
121.11	-0.599	-1.156	-0.014	1.302	242.60	-0.61	136.59	262.88	0.0	2.42	0.690	4.215	-4.749	-0.014
141.29	-0.352	-1.391	-0.014	1.435	255.79	-0.55	149.02	252.13	0.0	1.83	0.594	4.662	-3.769	0.012
161.48	-0.089	-1.574	-0.013	1.577	265.77	-0.46	162.99	217.13	0.0	1.36	0.511	4.826	-2.900	0.037
181.11	0.303	-1.806	-0.007	1.842	282.32	-0.21	154.57	136.82	0.0	1.02	0.395	4.723	-1.658	0.076
202.97	0.949	-1.935	0.005	2.155	296.12	0.14	130.44	116.24	0.0	0.87	0.301	4.297	-0.613	0.115
287.84	1.445	-1.964	0.021	2.439	306.34	0.50	115.86	110.82	0.0	0.81	0.242	3.781	0.084	0.144
332.73	1.877	-1.923	0.040	2.699	314.32	0.86	105.58	108.41	0.0	0.76	0.203	3.260	0.565	0.166
377.56	2.247	-1.831	0.062	2.899	320.82	1.22	98.56	106.98	0.0	0.72	0.176	2.759	0.909	0.182
422.42	2.556	-1.703	0.085	3.073	326.32	1.58	93.39	106.01	0.0	0.69	0.158	2.280	1.160	0.193
467.28	2.808	-1.549	0.104	3.203	331.12	1.95	88.02	105.45	0.0	0.65	0.146	1.826	1.340	0.201
512.14	3.004	-1.374	0.134	3.306	335.41	2.32	83.72	104.92	0.0	0.63	0.138	1.373	1.487	0.204
559.13	3.148	-1.180	0.160	3.386	339.45	2.72	78.44	104.42	0.0	0.61	0.133	0.921	1.594	0.204
606.36	3.238	-0.964	0.180	3.434	343.42	3.15	72.98	104.75	0.0	0.59	0.132	0.446	1.675	0.198
654.60	3.265	-0.739	0.282	3.454	347.25	3.62	66.05	104.33	0.0	0.59	0.134	-0.042	1.729	0.198
702.83	3.226	-0.508	0.236	3.274	351.04	4.12	57.90	104.71	0.0	0.60	0.140	-0.556	1.759	0.171
751.36	3.116	-0.275	0.257	3.130	354.95	4.69	47.61	104.37	0.0	0.63	0.152	-1.113	1.765	0.145
799.30	2.929	-0.044	0.273	2.942	359.15	5.33	35.16	107.34	0.0	0.73	0.172	-1.744	1.741	0.107
847.53	2.650	0.182	0.284	2.672	363.94	6.10	20.55	113.78	0.0	0.93	0.205	-2.496	1.673	0.049
895.73	2.268	0.395	0.285	2.312	369.01	7.07	6.77	152.92	0.0	1.40	0.266	-3.464	1.528	-0.046
944.00	1.716	0.578	0.268	1.831	374.61	8.43	13.62	251.87	0.0	2.66	0.399	-4.885	1.191	-0.223

180-12916-1

APPENDIX D

AN APPROXIMATION METHOD FOR PREDICTING CONSTRAINED THRUST PERFORMANCE

In part one of this appendix we develop an equation for approximating the optimum value of J associated with constrained thrust (C-T) propulsion. In part two we consider optimization of propulsion parameters as well. In part three we show how the program actually performs the optimization.

PART I

The method for approximating the C-T optimum value of J starts with the calculation of the corresponding variable thrust (V-T) optimum trajectory, and by means of a mathematically well defined series of approximations converts the C-T optimum problem into a relatively simple search problem. The initial acceleration (a_o) and exhaust velocity (c) of the vehicle are assumed to be fixed inputs, since their optimization will be treated in part two. For the moment we shall assume constant power, the case of variable power involving only a transformation of variables at the end of the development.

Preliminary Development

Our first task is to recast the C-T optimum problem into one involving only the difference between the C-T and V-T modes. Let a_c and a_v be the optimum C-T and V-T acceleration vectors respectively.

Let $J_v = \int_{t_o}^{t_1}(a_v, a_v)dt$ and $J_c = \int_{t_o}^{t_1}(a_c, a_c)dt$

where (.,.) is the regular dot (inner) product.

The limits t_o and t_1 are the departure and arrival times. Set

$\Delta J = J_c - J_v$ and $\Delta a = a_c - a_v$. Then

$$\Delta J = \int_{t_o}^{t_1} (\Delta a, \Delta a)dt + 2 \int_{t_o}^{t_1}(\Delta a, a_v)dt.$$

Let x_c and x_v be the position vectors of the C-T and V-T trajectories and set $\Delta x = x_c - x_v$. Let $y_c = kx_c/r_c^3$, $y_v = kx_v/r_v^3$ and set $\Delta y = y_c - y_v$. Here k represents the gravitational constant of the sun and r_c , r_v are

$|x_c|$, $|x_v|$ respectively. The two-body equations of motion $\ddot{x} + kx/r^3 = a$ yield

$$\int_{t_o}^{t_1}(\Delta a,\ a_v)dt = \int_{t_o}^{t_1}(\Delta\ddot{x} + \Delta y,\ a_v)dt.$$

Let W be the symmetric matrix $\partial y_v/\partial x_v$. Then

$$\begin{aligned}\int_{t_o}^{t_1}(\Delta\ddot{x} + \Delta y,\ a_v)dt &= \int_{t_o}^{t_1}(\Delta\ddot{x} + W\Delta x, a_v)dt \\ &+ \int_{t_o}^{t_1}(\Delta y - W\Delta x, a_v)dt \\ &= \int_{t_o}^{t_1}(\Delta x,\ \ddot{a}_v + Wa_v)dt \\ &- \left[(\Delta x, \dot{a}_v) - (\Delta\dot{x}, a_v)\right]_{t_o}^{t_1} \\ &+ \int_{t_o}^{t_1}(\Delta y - W\Delta x,\ a_v)dt.\end{aligned}$$

But $\ddot{a}_v + Wa_v = 0$ owing to the fact that a_v is the optimum V-T acceleration. We have finally

$$\begin{aligned}\int_{t_o}^{t_1}(\Delta a,\ a_v)dt &= \left[(\Delta x,\ \dot{a}_v) - (\Delta\dot{x},\ a_v)\right]_{t_o}^{t_1} \\ &+ \int_{t_o}^{t_1}(\Delta y - W\Delta x,\ a_v)dt\end{aligned} \qquad (1)$$

In the case of rendezvous at one of the endpoints, the corresponding values of Δx and $\Delta\dot{x}$ are zero. In the case of flyby, Δx and a_v vanish. In either event the first term on the right of (1) is zero. An approximation must be made if a hyperbolic excess velocity with arbitrary direction but fixed magnitude is specified. Then Δx and $2(\Delta\dot{x},\ \dot{x}_y) + (\Delta\dot{x},\ \Delta\dot{x}) = 0$. Assuming the optimum direction of $\dot{x}_c$ is near that of $\dot{x}_v$, the last equation shows that $(\Delta\dot{x},\ \dot{x}_v)$ vanishes to second order. Hence so does $(\Delta\dot{x},\ a_v)$, as $\dot{x}_v$ is parallel to a_v. Thus

$$\left[(\Delta x,\ \dot{a}_v) - (\Delta\dot{x},\ a_v)\right]_{t_o}^{t_1}$$

can be set equal to zero again. Equation (1) becomes

$$\int_{t_o}^{t_1}(\Delta a,\ a_v)dt = \int_{t_o}^{t_1}(\Delta y + W\Delta x,\ a_v)dt \qquad (2)$$

Now $\Delta y - W\Delta x$ is of order $|\Delta x|^2$. In the case of low initial accelerations the C-T mode will try to imitate the more optimum V-T mode as best it can and $|\Delta x|$ should be small over much of the interval $[t_o, t_1]$. Note that even for modest values of $|\Delta x|$, $\Delta y - W\Delta x$ is identically zero in the field-free rectilinear case and relatively small for trajectories to the outer planets. Also note that $|\Delta x|$ is zero at t_o and t_1 where $|a_v|$ often achieves its maximum values. Hence it is reasonable to assume that

$$\int_{t_o}^{t_1} (\Delta y - W\Delta x, a_v)dt \simeq 0$$

We finally obtain

$$\Delta J = \int_{t_o}^{t_1} (\Delta a, \Delta a)dt \tag{3}$$

$$0 = \int_{t_o}^{t_1} (a, a_v)dt \tag{4}$$

Modified Optimum Problem with One Side Condition

We can define an optimization problem from these two equations, namely minimizing ΔJ as defined by Equation (3) subject to the side condition (4). Minimizing ΔJ is the same as minimizing J_c and Equation (4) is equivalent to the boundary condition.

$$[(\dot{\Delta x}, a_v) - (\Delta x, \dot{a}_v)]_{t_o}^{t_1} = 0 \tag{5}$$

The quantity $J_v + \Delta J$ will then be a lower bound for J_c since the other boundary conditions have been omitted; however, it will be an improvement over J_v as Equation (3) implies $\Delta J \geq 0$ regardless of Δa.

Modified Optimum Problem with Two Side Conditions

Checking Equation (5) it is appraent that Equation (4) is a sufficient side condition in the case of rectilinear flyby. We are motivated, therefore, to obtain one more side condition so that our approximation will be exact for rectilinear rendezvous (and hyperbolic excess velocities) as well. Our method will then be precise for the retilinear case, and take into account some effects of central force behavior as well.

Let $b_v = t\, a_v - 2\dot{x}_v$. One can check that $\ddot{b}_v + Wb_v = 0$. Hence equation (1) holds with a_v replaced by b_v. In the same manner as before the two right hand terms can be set equal to zero and we have the additional side condition.

$$0 = \int_{t_o}^{t_1} (\Delta a, b_v)\, dt \tag{6}$$

Equation (6) is equivalent to the boundary condition

$$[(\Delta x,\ \dot{b}_v) - (\dot{x},\ b_v)]_{t_o}^{t_1} = 0 \tag{7}$$

It has not been found necessary yet to include the side condition (6). Apparently the side condition (4) is of overwhelming importance as far as the minimization of J is concerned and accounts for a large share of the difference between J_c and J_v even for the case of rendezvous.

Solution of Modified Optimum Problem with One Side Condition

Let us rewrite (3) and (4) as follows:

$$J_c = \int_{t_o}^{t_1} (a_c,\ a_c)\, dt \qquad \text{(minimize)} \tag{8}$$

$$J_v = \int_{t_o}^{t_1} (a_c,\ a_v)\, dt \qquad \text{(side condition)} \tag{9}$$

The following are constraints imposed on a_c by hardware limitations.

Constant Isp Constraint:

$a_c = \frac{a_o}{\mu}\nu$, where ν is a unit direction vector (except during coast when $\nu = 0$). Here $\mu = \mu(t)$ is the spacecraft mass fraction and satisfies $\dot{\mu} = -\frac{a_o}{c}\sigma$, $\mu(t_o) = 1$ where $\sigma = \sqrt{(\nu,\nu)}$. Note that $\sigma(1-\sigma) = 0$.

Cone Angle Constraint:

Let $c_v = \frac{-x_v}{r_v}$. Then, if θ is the cone angle of the acceleration, we have $(\nu,\ c_v) - \gamma\sigma = 0$, where $\gamma = \cos\theta$. (We are again assuming $x_v \simeq x_c$)

Let $\Theta = \{\theta_i,\ i = 1, m$ be the set of permissible cone angles with $\alpha_i = \cos\theta_i$. Then $\prod_{i=1}^{m} (\gamma-\alpha_i) = 0$.

We are now prepared to solve the optimization problem by the development of Reference 5. We shall assume the α_i are free because the result can easily be specialized to the case where the α_i are fixed. (Note m is fixed)

Let y(t) be defined by $\dot{y} = (a_c, a_v)$, $y(t_o) = 0$. Then (9) $\Rightarrow y(t_1) = J_v$.

Let $x_1 = y$, $x_2 = \mu$, $x_3 = \alpha$, $u_1 = \nu$, $u_2 = \gamma$. Here x_3 and α are vector functions with components α_i, and of course u_1 and ν are three elements vector functions. Apparently

$$f = \frac{a_o^2}{\mu^2}\sigma\ ,\quad G_1 = \frac{a_o}{\mu}(\nu,\ a_v),\quad G_2 = -\frac{a_o}{c}\sigma,\quad G_3 = 0$$

$$\text{and } H = \lambda_o \frac{a_o^2}{\mu^2}\sigma + \lambda_1 \frac{a_o}{\mu}(\nu,\ a_v) - \lambda_2 \frac{a_o}{c}\sigma$$

We set

$$R_1 = \sigma(1-\sigma) \qquad R_4 = -R_1$$

$$R_2 = (\nu,\ c_v) - \gamma\sigma \qquad R_5 = -R_2$$

$$R_3 = \prod_{i=1}^{m} (\gamma-\alpha_1) \qquad R_6 = -R_3$$

From Theorem 2 (Reference 5) we obtain the following necessary conditions

$$\text{(a)}\quad \dot{y} = \frac{a_o}{\mu}(\nu,\ a_v) \qquad y(t_o) = 0,\ y(t_1) = J_v$$

$$\text{(b)}\quad \dot{\mu} = \frac{-a_o}{c}\sigma \qquad \mu(t_o) = 1$$

$$\text{(c)}\quad \dot{\lambda}_1 = 0 \qquad (10)$$

$$\text{(d)}\quad \dot{\lambda}_2 = \lambda_o \frac{2a_o^2}{\mu^3}\sigma + \lambda_1 \frac{a_o}{\mu^2}(\nu,\ a_v)\ ,\ \lambda_2(t_1) = 0$$

(e) $\lambda_3 = \eta_3 \prod_{j\neq i} (\gamma - \alpha_j)$ $\qquad \lambda_3(t_o) = \lambda_3(t_1) = 0$

(f) $[\lambda_o \frac{a_o^2}{\mu 2} - \lambda_2 \frac{a_o}{c} + \eta_1 (1-2\sigma) - \gamma\eta_2]\rho + \lambda_1 \frac{a_o}{\mu} a_v + \eta_2 c_v = 0$

(g) $-\eta_2\sigma + \eta_3 \prod_{\sigma_i \neq \gamma} (\gamma - \alpha_i) = 0$

(h) H, λ_1, λ_2, λ_3 continuous $\lambda_o \geq 0$

(i) γ, ν satisfy $\min_{\gamma\nu}$ (H) (Maximum Principle)

(j) $[\frac{\lambda_o a_o^2}{\mu^2} - \lambda_2 \frac{a_o}{c} + (1-2\sigma)\eta_1 - \gamma\eta_2] \geq 0$ (Clebsch)

Here $\eta_1 = \mu_1 - \mu_4$, $\eta_2 = \mu_2 - \mu_5$, $\eta_3 = \mu_3 - \mu_6$ and $\nu = \sigma\rho$

From these necessary conditions we can derive more refined necessary conditions which lead to the solution of the minimization problem.

Without loss of generality we can normalize the multipliers by setting $\lambda_o = 1$.

Then $H = \sigma[\frac{a_o^2}{\mu^2} + \lambda_1 \frac{a_o}{\mu}(\rho, a_v) - \lambda_2 \frac{a_o}{c}]$

Let $\phi = \frac{1}{\lambda_1} \min_{\gamma\nu} \lambda_1 (\rho, a_v)$. This determines γ and ρ. In fact

$$\rho = \gamma c_v + \sqrt{1-\gamma^2}\, h, \text{ where } h = \frac{1}{r}[a_v - (a_v, c_v)c_v]$$

and $r = -\operatorname{sign}(\lambda_1)\,|a_v - (a_v\, c_v)c_v|$. Also

$$\lambda_1 \leq 0 \rightarrow \phi = \max_{\gamma=\alpha_i} \{\gamma(a_v, c_v) + \sqrt{1-\gamma^2}\,|a_v - (a_v, c_v)c_v|\}$$

$$\lambda_1 > 0 \rightarrow \phi = \min_{\gamma=\alpha_i} \{\gamma(a_v, c_v) - \sqrt{1-\gamma^2}\,|a_v - (a_v, c_v)c_v|\} \qquad (11)$$

It is possible (with considerable difficulty) to show that if a minimum exists with $\lambda_1 > 0$, there also is a better minimum with $\lambda_1 < 0$. Hence we assume $\lambda_1 < 0$.

After determining ϕ we obtain

$$H = -a_o \sigma k, \text{ where } k = -[\frac{a_o}{\mu^2} + \lambda_1 \frac{\phi}{\mu} - \lambda_2 \frac{1}{c}]$$

Using (10d) one can show that

$$\dot{k} = -\lambda_1 \frac{\dot{\phi}}{\mu} \text{ with } k(t_1) = -\frac{a_o}{\mu(t_1)} 2 - \lambda_1 \frac{\phi(t_1)}{\mu(t_1)}$$

$$\text{Hence } k(t) = -\frac{a_o}{\mu(t_1)} 2 - \lambda_1 [\frac{\phi}{\mu} + \frac{a_o}{c} \int_{t_o}^{t_1} \frac{\sigma\phi}{\mu 2} d\tau]$$

$$= -\lambda_1 \{[\frac{1}{\lambda_1} \frac{a_o}{\mu(t_1)} 2 + \frac{a_o}{c} \int_{t_o}^{t_1} \frac{\sigma\phi}{\mu} d\tau]$$

$$+ [\frac{\phi}{\mu} - \frac{a_o}{c} \int_{t_o}^{t_1} \frac{\sigma\phi}{\mu 2} d\tau]\}$$

Since H is to be minimized, $k < 0 \rightarrow \sigma = 0$ and $k > 0 \rightarrow \sigma = 1$.

Since $\lambda_1 < 0$ we may replace $k(t)$ by $k(t)/(-\lambda_1)$ without loss of generality.

Hence,

$$k(t) = \frac{\phi}{\mu} - \frac{a_o}{c} \int_{t_o}^{t} \frac{\sigma \phi}{\mu^2} d\tau - K \qquad (12)$$

$$\text{where } K = -[\frac{1}{\lambda_1} \frac{a_o}{\mu(t_1)^2} + \frac{a_o}{c} \int_{t_o}^{t_1} \frac{\sigma\phi}{\mu 2} d\tau]$$

If the α_i, $i = 1, m$ are to be optimized we can obtain additional equations for this purpose.

From (10g) $$\eta_3 = \frac{\eta_2 \sigma}{\prod_{\alpha_i \neq \gamma} (\gamma - \alpha_i)}$$

(10e) implies

$$\dot{\lambda}_{3_i} = 0 \text{ unless } \gamma = \alpha_i \text{ in which case}$$

$$\dot{\lambda}_{3_i} = \eta_3 \prod_{\alpha_i \neq \gamma} (\gamma - \alpha_i) = \eta_2 \beta$$

(3f) implies $\lambda_1 \frac{a_o}{\mu} (a_v 0 \rho) + \eta_2 (c_v 0 \rho) = 0$

$$\text{or } \{\frac{\lambda_1 a_o}{\mu} (\gamma - (a_v, c_v) \frac{1}{r}\sqrt{1-\gamma^2}) - \eta \frac{1}{2r} \sqrt{1-\gamma^2}\} a_v 0 c_v = 0$$

Hence

$$\eta_2 = \frac{\lambda_1 a_o}{\mu} (r \frac{\gamma}{\sqrt{1-\gamma^2}} - (a_v, c_v)), \text{ and}$$

$$\lambda_{3_i} = \frac{\lambda_1 a_o \sigma}{\mu \sqrt{1-\gamma^2}} (\gamma |a_v - (a_v, c_v) c_v| - \sqrt{1-\gamma^2} (a_v, c_v)) \quad \text{if } \gamma = \alpha_i$$

$$0 \quad \text{if } \gamma \neq \alpha_i$$

$$\text{Therefore, } 0 = \alpha_i \int_{t_o}^{t_1} |a_v - (a_v, c_v) c_v| \frac{\sigma}{\mu} dt - \sqrt{1-\alpha_i^2} \int_{t_o}^{t_1} (a_v, c_v) \frac{\sigma}{\mu} dt \qquad (13)$$
$$\gamma = \alpha_i \qquad\qquad \gamma = \alpha_i$$

This completes the derivation of refined necessary conditions. In summary we have the following equations:

(a) $J_c = a_o^2 \int_{t_o}^{t_1} \frac{\sigma}{\mu^2} dt$ (minimize) (14)

(b) $J_v = a_o \int_{t_o}^{t_1} \frac{\phi\sigma}{\mu} dt$ (side condition)

(c) $\dot{\mu} = -\frac{a_o}{c} \sigma, \ \mu(t_o) = 1$

(d) $\phi = \gamma \overset{\max}{=} \alpha_i \{\gamma(a_v, c_v) + \sqrt{1-\gamma^2} |a_v - (a_v, c_v) c_v|\}$

(e) $k = \frac{\phi}{\mu} - \frac{a_o}{c} \int_{t_o}^{t_1} \frac{\sigma\phi}{\mu^2} d\tau - K$ (switching function)

$k > 0 \Rightarrow 1, \ k < 0 \Rightarrow \sigma = 0$

(f) $0 = \alpha_i \int_{t_o}^{t_1} |a_v - (a_v, c_v) c_v| \frac{\sigma}{\mu} dt - \sqrt{1-\alpha_i^2} \int_{t_o}^{t_1} (a_v, c_v) \frac{\sigma}{\mu} dt \quad , i = 1, m$
$\gamma = \alpha_i \qquad\qquad \gamma = \alpha_i$ (Defines optimal α_i)

Integration is to be performed over sub intervals of $[t_o, t_1]$ where (d) chooses i^{th} cone angle.

Variable Power

The previous derivation has assumed constant power, but can be modified to include variable power. One more assumption has to be made if the power varies as a function of radius from the sun, namely that the power level time history is roughly the same for both the variable thrust and constrained thrust modes. This hypothesis is actually only an extension of our previous assumption that the position vector time histories of the two modes are similar. 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 assumptions (8) and (9) become

$$J_c = \int_{t_o}^{t_1} (a_c, a_c) \frac{p_o}{p(t)} \, dt \tag{15}$$

$$J_v = \int_{t_o}^{t_1} (a_c, a_v) \frac{p_o}{p(t)} \, dt \tag{16}$$

where

$$a_c = \frac{a_o}{\mu} \frac{p(t)}{p_o} \nu \text{ and } \dot{\mu} = -\frac{a_o}{c} \frac{p(t)}{p_o}$$

Here a_o and p_o are the initial acceleration and power respectively, assuming the spacecraft to be at a distance of 1 au from the sun if solar power is considered.

Now set $h_v(t) = a_v(t) \left(\frac{p_o}{p(t)}\right)$, $h_c(t) = a_c(t) \left(\frac{p_o}{p(t)}\right)$,

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

The above equations now become

$$J_c = \int_{\tau_o}^{\tau_1} (h_c, h_c) d\tau$$

$$J_v = \int_{\tau_o}^{\tau_1} (h_c, h_v) d\tau$$

$$h_c(\tau) = \frac{a_o}{\mu} \nu(\tau), \quad \frac{d\mu}{d\tau} = -\frac{a_o}{c} \sigma(\tau)$$

where

$$\tau_o = \tau(t_o) \text{ and } \tau_1 = \tau(t_1)$$

These equations are the same as the previous ones except for a change in variable names, and we have reduced the variable power case to that of constant power. (Note that the cone angle constraint equations remain unaffected by variable power. Also note that the case of constant power is just a special case of variable power with p = 1.)

PART II

In the previous section we considered the problem of minimizing

$$J_c = \int_{t_o}^{t_1} (a_c, a_c) \frac{P_o}{P} \, dt$$

for fixed a_o and c. J_c can also be optimized with respect to these parameters as well, however this is not done because payload, while formally increasing as J_c decreases, also depends explicitly on a_o and c. To be more specific the rocket equations yield

$$\mu_1 = \frac{1}{1 + \frac{Jc}{a_o c}}$$

where μ_1 is the final mass fraction. If μ_L is the payload mass fraction, we have

$$\mu_L = f(\mu_1, a_o, c),$$

where f is a nonlinear function of μ_1, a_o, and c. It is f that should be maximized with respect to a_o and c.

Actually it is more convenient to use powerplant mass fraction μ_w as a parameter instead of a_o. If m_w is the powerplant mass and m_o the initial spacecraft mass, then $\mu_w = \frac{m_w}{m_o}$. Defining the powerplant specific mass by $\alpha_w = \frac{m_w}{P_o}$, we have

$$a_o = \frac{2\eta\mu_w}{\alpha_w c} \quad \text{or} \quad \mu_w = \frac{\alpha_w a_o c}{2\eta}$$

where $\eta = \eta(c)$ is efficiency as a function of exhaust velocity. It follows that

$$\mu_1 = \frac{1}{1 + \frac{\alpha_w J_c}{2\eta\mu_w}} \text{ , and}$$

$$\mu_L = f\left(\frac{1}{1 + \frac{\alpha_w J_c}{2\eta\mu_w}} , \frac{2\eta\mu_w}{\alpha_w c} , c\right) \tag{17}$$

μ_L can now be optimized with respect to either μ_w or c or both. α_w is to be specified along with $\eta(c)$. Often $\eta(c)$ is specified analytically by the formula

$$\eta(c) = \frac{B}{1 + (D/c)^2} \qquad \text{for some constants B and D.}$$

f is sometime taken as $\mu_1 - \mu_w$ although more complicated definitions are acceptable.

PART III

In this part we shall discuss how the program solves Equations (14), and later how it optimizes (17) with respect to μ_w and/or c. Hence let us fix μ_w and c (and consequently a_o) and look at (14). Initially we assume the set Θ of permissible cone angles is infinite and contains every value between 0° and 180°. Then $\emptyset = |a_v|$ and the only problem is choosing K of Equation (14e) so that (14b) is satisfied. Let $K_{max} = t\varepsilon^{max}_{\{t_o,t_1\}}\{\emptyset(t)\}$ Clearly we must have $K \leq K_{max}$, otherwise $k(t)<0$ and no thrusting will occur. Let $K_{min} = \frac{a_o}{c}\int_{t_o}^{t_1} \frac{\sigma\phi}{\mu^2}\, d\tau$. One can shcw that if $K = K_{min}$, the right hand side of (14b) is maximized. The procedure for solving (14b) is to start K at K_{max} where the right side of (14b) as well as J_c are zero. If K is then gradually decreased, and (14 a, b, c) are integrated forward using (14e) to determine σ, both these quantities will increase. K should be decreased until the right side of (14b) is greater than J_v or else $K<K_{min}$. In the latter case the propulsion parameters are insufficient for a solution. In the former case interval halving will produce a root of

(14b). The case where Θ is a finite set of specified cone angles is handled in a similar fashion. Before K is selected, $\emptyset$ is evaluated from (14d) and the procedure is then identical.

If Θ is a finite set of m unspecified cone angles the solution becomes rather complicated. The most logical attack is to choose m nominal cone angles, run through the above procedure for fixed cone angles, and then determine how to adjust these angles to better satisfy (14f). This is an iterative procedure, and quite time consuming. Furthermore pitfalls abound. If two or more cone angles at some time happen to coincide, (14f) becomes undefined. Also, if the nominal values are not chosen correctly, one or more angles will never be picked by (14d) leading to a solution for less than m angles. This phenomenon seems to be the rule rather than the exception, and a procedure must be devised for reinserting "lost" angles back into the iterations.

Our approach is to make a slight approximation. First we run the unconstrained cone angle case to get an idea of where thrusting is likely to occur. In fact we use the resultant μ and σ of this case to evaluate (14f). Next we note that (14f) is really telling us that we should choose our cone angles in such a manner that the right side of (14b) is maximized. This is logical since it costs fuel to satisfy (14b). Therefore we need only consider (14d) and (14b). First we choose m nominal cone angles at equally spaced intervals between 0° and 180°. Then we run through (14d) to define the time intervals over which each cone angle is optimal. The right side of (14b) is evaluated for each of these intervals and better cone angles found by setting derivatives zero. The procedure is then repeated. If some cone angle is optimal in the sense of (14d) at no time in $[t_o, t_1]$ it is reinserted by checking to see where $\emptyset$ differs most from $|a_v|$. The angle is then given the value making $\emptyset$ equal to $|a_v|$ at this time. Note that such a solution will insure the angle gets picked by (14d) on the next iteration. The above procedure is assumed to have converged when the cone angles and the right side of (14b) no longer change.

Now we consider the options of optimizing μ_w and/or c. If μ_w is fixed, c is optimized by repeatedly applying the above algorithms for fixed μ_w and c. The method of search on c is "Golden Sectioning" (Page 242 Ref. 6) using $\mu_L = f$ as a performance criterion. The situation is identical if c is fixed and μ_w is optimized.

If μ_w and c are both to be optimized we switch to two equivalent parameters, namely a_o/c and K (see 14d). Two Golden Sectioning loops are executed, the outer on a_o/c, the inner one K for each value of a_o/c. The technique of Golden Sectioning is fairly reliable regardless of the nature of f unless more than one local optimum exists, in which case the global optimum may not necessarily be the one found.

APPENDIX E
LISTINGS

INDEX OF SUBROUTINES FOR APPENDIX E

*These are function subprograms

```
      SUBROUTINE ALIGN(X,AL,VX,VXA,VXAA)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NO,NH,NAP
      COMMON/BVL/D1,D2,H1,H2,V1(12),V2(12)
      COMMON/TIMEQ/TT,T(6),TACUM(7),T1(6),T2(6),T3(6),TP(6),TTP
      COMMON/OPTION/NP1,NP2,NB1,NB2,NV1,NV2,NPO
      DIMENSION X(10,3,6),AL(3,2),VX(6,2),VXA(18,2),VXAA(54,2)
      CALL VCAL
     C(NP1,NB1,NV1,H1,T(1),AL(1,1),V1,VX(1,1),VXA(1,1),VXAA(1,1))
      CALL VCAL
     C(NP2,NB2,NV2,-H2,-T(NL),AL(1,2),V2,VX(1,2),VXA(1,2),VXAA(1,2))
      DO 10 M=1,ND
      IF(NV1.NE.0) X(2,M,1)=VX(M+3,1)
      IF(NV2.NE.0) X(NPM1,M,NL)=VX(M+3,2)
      X(1,M,1)=VX(M,1)
10    X(NP,M,NL)=VX(M,2)
      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

      FUNCTION AMAP(H,CONESC,NCONE,ZETA,ACONE,ACLOCK,AB,ABP)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      COMMON/OUT/XV(30,6),AV(30,6),WV(30,6)
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NO,NH,NAP
      COMMON/TIMEQ/TT,T(6),TACUM(7),T1(6),T2(6),T3(6),TP(6),TTP
      COMMON/NCONS/PI,PI2,PIF,GRAV,TCON,VELCON,POWCON,PCON
      DIMENSION CONESC(4,2),A(3),B(3),C(3)
      DATA C /0.,0.,-1./
      A(3)=0.
      B(3)=0.
      S=H
      CALL MODLEG(T,NL,S,LN)
      CALL POLEVL(AV(1,LN),A,S,ND,NP,N,0)
      CALL POLEVL(XV(1,LN),B,S,ND,NP,N,0)
```

```
      Z=0.
      DO 5 M=1,3
5     Z=Z+B(M)**2
      Z=1./SQT(Z)
      DO 10 M=1,3
10    B(M)=-Z*B(M)
      AA=0.
      AB=0.
      AC=0.
      BC=0.
      DO 15 M=1,3
      AA=AA+A(M)**2
      AB=AB+A(M)*B(M)
      AC=AC+A(M)*C(M)
15    BC=BC+B(M)*C(M)
      ABP=SQT(AA-AB**2)
      Z=-B(1)*C(2)+B(2)*C(1)
      ZETA=PIF*ATN2(Z,-C(3)+BC*B(3))
      Z=B(1)*(C(2)*A(3)-C(3)*A(2))+B(2)*(C(3)*A(1)-C(1)*A(3))+
     CB(3)*(C(1)*A(2)-C(2)*A(1))
      ACLOCK=PIF*ATN2(Z,AC-BC*AB)
      IF(ACLOCK.LT.0.) ACLOCK=ACLOCK+360.
      Z=SQT(AA)
      CT=AB/Z
      ST=SQT(1.-CT**2)
      IF(NCONE.EQ.0) GO TO 100
      Z=-2.*Z
      DO 35 I=1,NCONE
      W=CONESC(I,1)*AB+CONESC(I,2)*ABP
      IF(W.LT.Z) GO TO 35
      Z=W
      CT=CONESC(I,1)
      ST=CONESC(I,2)
35    CONTINUE
100   ACONE=PIF*ATN2(ST,CT)
      APAP=Z
      RETURN
      END
```

```
      SUBROUTINE AVTEST(AV,T2,NLP,NL,ND,NP)
      IMPLICIT REAL*8(A-H,C-Z)                                          360 IN
      DIMENSION AV(10,3,6),T2(6),AVM(2,6)
      DATA CONTIN /0.1/
      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)=SQT(Z1)*T2(LN)
  525 AVM(2,LN)=SQT(Z2)*T2(LN)
      DO 10 LN=2,NL
      RATIO=(AVM(1,LN)-AVM(2,LN-1))/AMN(AVM(1,LN),AVM(2,LN-1))
      IF(AB(RATIO).GT.CONTIN) NLP=NL+1
   10 CONTINUE
      RETURN
      END
```



```
      SUBROUTINE BAKSUB(A,B,X,NL,N1,N2,NCP,NAP)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      DIMENSION A(31,31,6),B(31,6),X(31,6),N1(6),N2(6),NCP(6)
      DO 45 LN=1,NL
      NCP3=N1(LN)
      NCP1=NCP3
      NCP2=N2(LN)
      IF(LN.EQ.1) GO TO 9
      NCPL=NCP(LN)
      NCP1=NCP1+NCPL
      DO 8 J=1,NCPL
      JLN=NCP3-1+J
      JLNM1=N2(LN-1)-NCPL+J
    8 X(JLN,LN)=X(JLNM1,LN-1)
    9 DO 40 J=NCP1,NCP2
      JM1=J-1
      H=B(J,LN)
      IF(J.EQ.NCP3) GO TO 20
```

```
      DC 10 K=NCP3,JM1
10    W=W-A(J,K,LN)*X(K,LN)
20    X(J,LN)=W*A(J,J,LN)
40    CONTINUE
45    CCNTINUE
      IF(NAP.EQ.0) GO TO 78
      JP=N2(NL)+1
      W=B(JP,NL)
      DC 50 LN=1,NL
      NCP3=N1(LN)
      IP=N2(LN)+1
      IF(LN.LT.NL) NCP2=N2(LN)-NCP(LN+1)
      IF(LN.EQ.NL) NCP2=N2(LN)
      DC 50 K=NCP3,NCP2
50    W=W-A(IP,K,LN)*X(K,LN)
      X(JP,NL)=W*A(JP,JP,NL)
      X(JP,NL)=X(JP,NL)*A(JP,JP,NL)
78    DC 92 LN=1,NL
      LNP=NL+1-LN
      NCP1=N1(LNP)
      NCP3=N2(LNP)
      NCP2=NCP3
      IP=NCP2+1
      IF(LN.EQ.1) GO TO 89
      NCPL=NCP(LNP+1)
      NCP2=NCP2-NCPL
      DO 85 J=1,NCPL
      JLN=NCP2+J
      JLNM1=N1(LNP+1)+J-1
85    X(JLN,LNP)=X(JLNM1,LNP+1)
89    DC 90 J=NCP1,NCP2
      L=NCP2+NCP1-J
      LP1=L+1
      W=X(L,LNP)
      IF(L.EQ.NCP3) GO TO 87
      DC 80 K=LP1,NCP3
80    W=W-A(K,L,LNP)*X(K,LNP)
87    IF(NAP.GT.0) W=W-A(IP,L,LNP)*X(JP,NL)
90    X(L,LNP)=W*A(L,L,LNP)
92    CONTINUE
```

```
      RETURN
      END


      SUBROUTINE COEFF(XO,DX1,DX2,AV,WV,CO,TP)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      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,NO,NH,NAP
      COMMON/SKRCH1/GO(10,3),G1(10,3),G2(10,3),G3(10,3),G4(10,3),
     CG(10,3),X1(10,3),X2(10,3),CO(3),C1(3),C2(3),DUM1(159)
      DIMENSION XO(10,3),DX1(10,3),DX2(10,3),AV(10,3),WV(10,3),CO(9)
      CALL MULT3(V,DX1,X1,ND,NP)
      CALL MULT4(BA,X1,G1,ND,NP,N,1)
      CALL MULT3(V,DX2,X2,ND,NP)
      CALL MULT4(BA,X2,G2,ND,NP,N,1)
      DO 39 I=1,NP
      Z=0.
      DO 8 M=1,ND
    8 Z=Z+XO(I,M)**2
      R2=1./Z
      R1=SQT(R2)
      R2=TP*R2
      CCO=0.
      CO1=0.
      CO2=0.
      C11=0.
      C12=0.
      C22=0.
      DO 19 M=1,NC
      CC(M)=R1*XO(I,M)
      C1(M)=R1*X1(I,M)
      C2(M)=R1*X2(I,M)
      CCO=CCO+CO(M)*CO(M)
      CO1=CO1+CC(M)*C1(M)
      CO2=CO2+CO(M)*C2(M)
      C11=C11+C1(M)*C1(M)
      C12=C12+C1(M)*C2(M)
      C22=C22+C2(M)*C2(M)
   19 CONTINUE
      CS=C11+CO2
```

```
      W0=WV(I,1)
      W1=W0*WV(I,2)*C01
      W2=W0*(WV(I,2)*CS+WV(I,3)*C01**2)
      DO 39 L=1,ND
      G0(I,L)=AV(I,L)
      G1(I,L)=G1(I,L)+R2*(C1(L)+C0(L)*(-3.*C01))
      G2(I,L)=G2(I,L)+R2*(C2(L)+C1(L)*(-6.*C01)
     C+C0(L)*(15.*C01**2-3.*CS))
      G3(I,L)=R2*(C2(L)*(-9.*C01)+C1(L)*(45.*C01**2-9.*CS)
     C+C0(L)*(45.*C01*CS-105.*C01**3-9.*C12))
      G4(I,L)=R2*(C2(L)*(90.*C01**2-18.*CS)+C1(L)*(180.*C01*CS-36.*C12
     C-420.*C01**3)+C0(L)*(945.*C01**4-630.*CS*C01**2+180.*C12*C01
     C+45.*CS**2-9.*C22))
      G4(I,L)=W0*G4(I,L)+4.*W1*G3(I,L)+6.*W2*G2(I,L)
      G3(I,L)=W0*G3(I,L)+3.*W1*G2(I,L)+3.*W2*G1(I,L)
      G2(I,L)=W0*G2(I,L)+2.*W1*G1(I,L)+W2*G0(I,L)
      G1(I,L)=W0*G1(I,L)+W1*G0(I,L)
      GC(I,L)=W0*G0(I,L)
   39 CONTINUE
      CALL MULT4(AA,G0,G,ND,NP,N,1)
      CALL MULT2(G,G0,P00,ND,NP)
      CALL MULT2(G,G1,P01,ND,NP)
      CALL MULT2(G,G2,P02,ND,NP)
      CALL MULT2(G,G3,P03,ND,NP)
      CALL MULT2(G,G4,P04,ND,NP)
      CALL MULT4(AA,G1,G,ND,NP,N,1)
      CALL MULT2(G,G1,P11,ND,NP)
      CALL MULT2(G,G2,P12,ND,NP)
      CALL MULT2(G,G3,P13,ND,NP)
      CALL MULT2(G,G4,P14,ND,NP)
      CALL MULT4(AA,G2,G,ND,NP,N,1)
      CALL MULT2(G,G2,P22,ND,NP)
      CALL MULT2(G,G3,P23,ND,NP)
      CALL MULT2(G,G4,P24,ND,NP)
      CALL MULT4(AA,G3,G,ND,NP,N,1)
      CALL MULT2(G,G3,P33,ND,NP)
      CALL MULT2(G,G4,P34,ND,NP)
      CALL MULT4(AA,G4,G,ND,NP,N,1)
      CALL MULT2(G,G4,P44,ND,NP)
      CC(1)=P00
```

```
      CC(2)=2.*P01
      CC(3)=P11+P02
      CC(4)=P12+P03/3.
      CC(5)=P22/4.+P13/3.+P04/12.
      CC(6)=P14/12.+P23/6.
      CC(7)=P33/36.+P24/24.
      CC(8)=P34/72.
      CC(9)=P44/576.
      RETURN
      END


      SUBROUTINE CONOP(CONE,NCONE,STP,NST,BOT,PHE,PHE1,PHE2,ACONE,NNN)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      COMMON/NCONS/PI,PI2,PIF,GRAV,TCON,VELCON,POWCON,PCON
      DIMENSION CONE(4),STP(30,2),PHE(301),PHE1(301),PHE2(301),
     CACCNE(301)
      COMMON/PXXPX/DUM1(4803),PHE3(301),PHE4(301),PHE5(301),PHE6(301),
     CPHE7(301),G1(4),G2(4),S1(4),S2(4)
      DATA DELTA,NIT /.001,20/
      L=1
      FLTNM1=FLT(NNN-1)
      DO 29 I=1,NNN
      S=FLT(I-1)/FLTNM1
      IF((S.GE.STP(L,1)).AND.(L.LT.NST)) L=L+1
      K=L-2*(L/2)
      IF(K.EQ.0) BETA=0.
      IF(K.EQ.1) BETA=1./(1.-(STP(L,2)+S-STP(L,1))*BOT)
      PHE7(I)=BETA*PHE(I)
      PHE5(I)=BETA*PHE1(I)
   29 PHE6(I)=BETA*PHE2(I)
      SUM1=0.
      SUM2=0.
      PHE3(1)=0.
      PHE4(1)=0.
      DO 39 I=2,NNN,2
      PHE3(I)=SUM1+(5.*PHE5(I-1)+8.*PHE5(I)-PHE5(I+1))/4.
      PHE4(I)=SUM2+(5.*PHE6(I-1)+8.*PHE6(I)-PHE6(I+1))/4.
      SUM1=SUM1+PHE5(I-1)+4.*PHE5(I)+PHE5(I+1)
      SUM2=SUM2+PHE6(I-1)+4.*PHE6(I)+PHE6(I+1)
```

```
      PHE3(I+1)=SUM1
   39 PHE4(I+1)=SUM2
      P=0.
      DO 45 J=1,NCONE
      X=PI*FLT(2*J-1)/FLT(2*NCONE)
      G1(J)=CN(X)
   45 G2(J)=SN(X)
      DO 99 IT=1,NIT
      DO 49 J=1,NCONE
      S1(J)=0.
   49 S2(J)=0.
      PHEMAX=0.
      DO 79 I=1,NAN
      Z=-AB(PHE1(I))-AB(PHE2(I))
      DO 59 J=1,NCONE
      W=G1(J)*PHE1(I)+G2(J)*PHE2(I)
      IF(W.LT.Z) GO TO 59
      Z=W
      K=J
   59 CONTINUE
      W=PHE7(I)-Z
      IF(W.LT.PHEMAX) GO TO 65
      PHEMAX=W
      IMAX=I
   65 IF(I.EQ.NAN) GO TO 79
      S1(K)=S1(K)+PHE3(I+1)-PHE3(I)
      S2(K)=S2(K)+PHE4(I+1)-PHE4(I)
   79 CONTINUE
      PSAVE=P
      P=0.
      TEST=0.
      K=0
      DO 89 J=1,NCONE
      GSAVE=G1(J)
      W=SQT(S1(J)**2+S2(J)**2)
      IF(W.GT.0.) GO TO 85
      IF(K.EQ.1) GO TO 88
      K=1
      X=ACONE(IMAX)/PIF
      G1(J)=CN(X)
```

```
      G2(J)=SN(X)
      GO TO 88
   85 G1(J)=S1(J)/W
      G2(J)=ABS(S2(J))/W
   88 TEST=TEST+(G1(J)-GSAVE)**2
   89 P=P+G1(J)*S1(J)+G2(J)*S2(J)
      TEST=SQT(TEST+(1.-PSAVE/P)**2)
      IF((TEST.LT.DELTA).OR.(NCONE.EQ.1)) GO TO 100
   99 CONTINUE
  100 CONTINUE
      DO 199 J=1,NCONE
  199 CONE(J)=PIF*ATN2(G2(J),G1(J))
      RETURN
      END


      SUBROUTINE CONST(NP,N)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      DIMENSION P(10),Q(10),R(10)
      COMMON/NCONS/PI,PI2,PIF,GRAV,TCON,VELCON,POWCON,PCON
      COMMON/CONS1/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
      COMMON/CONS2/B(10,10),C(10,10),D(10,10),E(10,10),F(10,10)
      NPM1=NP-1
      FNPM1=FLT(NPM1)
      DO 1 I=1,NP
      Z=PI*FLT(I-1)/FNPM1
      P(I)=SN(Z)**2
      CHEB(I)=SN(Z/2.)**2
      IF(I.NE.1) Q(I)=2.*(-1.)**I/CHEB(I)
    1 R(I)=CN(Z)
      Q(1)=-(1.+2.*FNPM1**2)/3.
      Q(NP)=Q(NP)/2.
      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=FLT(4*(J+I-K)**2-1)
      F2=FLT(4*(J-I)**2-1)
      AA(I,J,K)=-(1./F1+1./F2)/2.
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)=A(1,J,2)/2.
      A(N,J,1)=A(N,J,1)/2.
      DO 50 K=1,2
      BA(J,1,K)=BA(J,1,K)/2.
50    A(J,1,K)=A(J,1,K)/2.
      CALL MULT7(AA,A,AA,N,1)
      CALL MULT7(AA,A,AA,N,2)
      CALL MULT8(AA,AA,C,NP,N,2)
      CALL MULT8(BA,BA,B,NP,N,1)
      RETURN
      END
```

```
      SUBROUTINE CCPTR(PHE,NNN,PHEMAX,PV,T,WMU,RO3,ALPHAW,ST,NST,WMU1)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      DIMENSION PHE(301),ST(30)
      DATA TAU,DELTA1,DELTA2 /1.618034,.005,.00001/
      RC=0.
      P=1.E20
      RC1=RC
      P1=P
      RO2=10.*TAU
      DO 15 I=1,20
      CALL RCOTR(PHE,NNN,PHEMAX,PV,T,WMU,RO2,ALPHAW,ST,NST,WMU1)
      P2=-WMUL(WMU,RO2,ALPHAW,WMU1)
      IF(P2-P) 10,20,20
   10 RC1=RC
      P1=P
      RC=RO2
      P=P2
   15 RO2=RC1+(TAU+1.)*(RC-RO1)
   20 DO 90 I=1,20
      RO3=RC1+(RO2-RO1)/TAU
      CALL RCOTR(PHE,NNN,PHEMAX,PV,T,WMU,RO3,ALPHAW,ST,NST,WMU1)
      P3=-WMUL(WMU,RO3,ALPHAW,WMU1)
      IF(P-P3) 60,60,70
   60 RC2=RC1
      P2=P1
      RC1=RO3
      P1=P3
      GO TO 80
   70 RC1=RO
      P1=P
      RC=RC3
      P=P3
   80 IF((AB(P1-P).LT.DELTA1).AND.(AB(P2-P).LT.DELTA1)) GO TO 95
   90 CONTINUE
   95 IF((AB(P1-P).LT.DELTA2).AND.(AB(P2-P).LT.DELTA2)) GO TO 99
      A=(RO-RO2)*P1+(RO1-RO)*P2+(RO2-RO1)*P
      B=(RC-RO2)*(RC+RO2)*P1+(RO1-RO)*(RO1+RO)*P2+
     C(RC2-RC1)*(RC2+RC1)*P
```

```
      RO3=B/(2.*A)
98    CONTINUE
      CALL ROOTR(PHE,NNN,PHEMAX,PV,T,WMU,RO3,ALPHAW,ST,NST,WMU1)
      P3=-WMUL(WMU,RO3,ALPHAW,WMU1)
      IF(P3.LT.P) RETURN
99    IF(RO3.EQ.RO) RETURN
      RO3=RO
      GO TO 98
      END


      SUBROUTINE CTMODE(WMU,C,ALPHAW,CONE,NCONE,NCONOP,NMUOP,NCOP)
      IMPLICIT REAL*8(A-H,O-Z)                                        36C IN
      REAL WMU,C,ALPHAW,CONE,BDY,PV,PC,BT,PMF
      COMMON/BDYP/BDY(3,4,2),PV,PC,BT,PMF
      COMMON/COUNT/NCOUNT,NLP
      COMMON/TIMEQ/TT,DUM1(38)
      COMMON/NCONS/PI,PI2,PIF,GRAV,TCON,VELCON,POWCON,PCON
      COMMON/INDEX/NNN,NST,NCONEP,NCONP,NMUOPP,NCOPP,DUM4(84)
      COMMON/PXXPX/PR(301),PHE(301),TYME(301),FTYME(301),SCRCH(301),
     CAMAG(301),ACONE(301),ACLOCK(301),PHE1(301),PHE2(301),
     CST(30),STP(30,2),CONESC(4,2),BOT,CT,TTDAY,T,WMUP,CP,ALPHAP,WMULP,
     CCONEP(4),TB,PCP,PMFP,ETAP,DUM2(3200)
      DIMENSION CONE(4)
      PVP=PV
      WMUP=WMU
      CP=C
      ALPHAP=ALPHAW
      NCONP=NCONCP
      NMUOPP=NMUOP
      NCOPP=NCOP
      DO 5 I=1,4
5     CONEP(I)=CONE(I)
      NNN=20*NLP+1
      FLTNM1=FLT(NNN-1)
      DT=TT/FLTNM1
      DO 7 I=1,NNN
      X=FLT(I-1)*DT
7     PR(I)=PMAP(X)
      CALL RESCAL(NNN,TAVSTR)
```

```
      T=TT*TAVSTR
      IF(NCCNCP.EQ.1) NCONEP=NCONE
      IF(NCCNOP.NE.1) NCCNEP=0
10    IF(NCONEP.EQ.0) GO TO 20
      DO 19 I=1,NCONEP
      X=CCNEP(I)/PIF
      CONESC(I,1)=CN(X)
19    CONESC(I,2)=AB(SN(X))
20    PHEMAX=0.
      DO 40 I=1,NNN
      X=TYME(I)*TT
      PHE(I)=AMAP(X,CONESC,NCONEP,ZETA,ACONE(I),ACLOCK(I),PHE1(I),
     CPHE2(I))/PMAP(X)
40    PHEMAX=AMX(PHEMAX,PHE(I))
      IF(PHEMAX.EQ.0.) GO TO 60
      IF((NMUCP.EQ.0).AND.(NCOP.EQ.0))
     CCALL ROOTR(PHE,NNN,PHEMAX,PVP,T,WMUP,CP,ALPHAP,ST,NST,WMU1P)
      IF((NMUOP.EQ.0).AND.(NCOP.EQ.1))
     CCALL CCPTR(PHE,NNN,PHEMAX,PVP,T,WMUP,CP,ALPHAP,ST,NST,WMU1P)
      IF((NMUOP.EQ.1).AND.(NCOP.EQ.0))
     CCALL MOPTR(PHE,NNN,PHEMAX,PVP,T,WMUP,CP,ALPHAP,ST,NST,WMU1P)
      IF((NMUOP.EQ.1).AND.(NCOP.EQ.1))
     CCALL MCOPTR(PHE,NNN,PHEMAX,PVP,T,BOT,WMUP,CP,ALPHAP,ST,NST,WMU1P)
      IF(WMU1P) 60,60,80
60    CONTINUE
      BT=TT
      PMF=-1.
      NST=-1
      PC=100.*PV
      WRITE(6,1000)
      RETURN
80    TBF=ST(1)
      NST=NST+1
      ST(NST)=1.
      DO 199 I=1,NST
      K=I-2*(I/2)
      IF(I.GT.1.AND.K.EQ.1) TBF=TBF+(ST(I)-ST(I-1))
      STP(I,1)=ST(I)
199   STP(I,2)=TBF
      ETAP=ETA(CP)
```

```
      IF(PHEMAX.LT.PHE(I))  PHEMAX=PHE(I)
40    CONTINUE
```

```
      IF((NMUOP.EQ.0).OR.(NCUP.EQ.0))
     CBOT=2.*WMLP*ETAP*T*POWCON/(ALPHAP*CP**2)
      IF((NCONOP.NE.2).OR.(NCCNEP.EQ.NCONE)) GO TO 250
      NCCNEP=NCCNE
      CALL CCNOP(CCNEP,NCCNEP,STP,NST,BOT,PHE,PHE1,PHE2,ACONE,NNN)
      GC TO 10
250   CCNTINUE
      DC 300 I=1,NST
      IF(ST(I).GE.1.) GC TO 300
      TS=ST(I)*FLTNM1+1.
      NTS=TS
      TS=TS-FLT(NTS)
      ST(I)=TYME(NTS)+TS*(TYME(NTS+1)-TYME(NTS))
3C0   CONTINUE
      TB=ST(1)
      DC 3C1 I=1,NST
      K=I-2*(I/2)
301   IF(I.GT.1.AND.K.EQ.1) TB=TB+(ST(I)-ST(I-1))
      TTDAY=TCCN*TT
      BT=TB*TT
      TB=TTDAY*TB
      PMFP=WMUL(WMUP,CP,ALPHAP,WMU1P)
      CT=CP*VELCON/T
      PCP=(T*TBF*(BOT*CT)**2)/WMU1P
      WPU=WMUP
      C=CP
      PC=PCP
      PCP=PCCN*PCP
      PMF=PMFP
      DO 305 I=1,4
305   CONE(I)=CCNEP(I)
      RETURN
1CC0  FORMAT(//4X46HMISSICN IMPOSSIBLE EVEN WITH CONTINUOUS THRUST)
      END


      SUBRCUTINE CTCUT
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      REAL BB,DC,TANKS
      COMMCN/INERTS/TANKS
```

```
      COMMON/THRUST/BB,DD
      COMMON/INDEX/NNN,NST,NCONEP,NCONP,NMUOPP,NCOPP,DUM4(84)
      COMMON/PXXPX/PR(301),PHE(301),TYME(301),FTYME(301),SCRCH(301),
     CAMAG(301),ACONE(301),ACLOCK(301),PHE1(301),PHE2(301),
     CST(30),STP(30,2),CONESC(4,2),BOT,CT,TTDAY,T,WMUP,CP,ALPHAP,WMU1P,
     CCONEP(4),TB,PCP,PMFP,ETAP,DUM2(3200)
      DIMENSION A(4,2)
      IF(NST.EQ.(-1)) RETURN
      WRITE(6,100)
      WRITE(6,150) BB,DD,TANKS
      WRITE(6,200) ETAP,ALPHAP
      IF((NMUOPP.EQ.0).AND.(NCOPP.EQ.0)) WRITE(6,201) WMUP,CP
      IF((NMUOPP.EQ.0).AND.(NCOPP.EQ.1)) WRITE(6,202) WMUP,CP
      IF((NMUOPP.EQ.1).AND.(NCOPP.EQ.0)) WRITE(6,203) WMUP,CP
      IF((NMUOPP.EQ.1).AND.(NCOPP.EQ.1)) WRITE(6,204) WMUP,CP
      IF(NCONP.EQ.0) WRITE(6,205)
      IF(NCONP.EQ.1) WRITE(6,206)(CONEP(I),I=1,NCONEP)
      IF(NCONP.EQ.2) WRITE(6,207)(CONEP(I),I=1,NCONEP)
      WRITE(6,208) PCP,TB,WMU1P,PMFP
      NNN2=NNN/2
      NNN2P1=NNN2+1
      L=1
      FLTNM1=FLT(NNN-1)
      DO 20 I=1,NNN
      S=FLT(I-1)/FLTNM1
      IF((S.GE.STP(L,1)).AND.(L.LT.NST)) L=L+1
      K=L-2*(L/2)
      IF(K.EQ.0) AMAG(I)=0.
20    IF(K.EQ.1) AMAG(I)=BOT*CT/(1.-(STP(L,2)+S-STP(L,1))*BOT)
      WRITE(6,1000)
      DO 80 I=1,NNN2P1
      J2=2
      IF(I.EQ.1) J2=1
      DO 75 J=1,J2
      L=I+(J-1)*NNN2
      A(1,J)=TTDAY*TYME(L)
      A(3,J)=ACONE(L)
      A(4,J)=ACLOCK(L)
      FS=TYME(L)*FLTNM1+1.
      K=FS
```

```
      FS=FS-FLT(K)
      A(2,J)=AMAG(L)*(PR(K)+FS*(PR(K+1)-PR(K)))
   75 CONTINUE
   80 WRITE(6,1001)((A(K,J),K=1,4),J=1,J2)
  100 FORMAT(// 40X,33HPREDICTED CONSTANT ISP TRAJECTORY)
  150 FORMAT(/  2X,18HCOMMON BLOCK INPUT,15X2HBB,F7.2,15X2HDD,F7.2,
     C15X5HTANKS,F7.2)
  200 FORMAT(/  2X,16HEFFICIENCY (ETA),F10.3,
     C30X33HPOWERPLANT SPECIFIC MASS (ALPHAW),F10.3,2X5HKG/KW)
  201 FORMAT(/  2X,35HSPECIFIED POWERPLANT FRACTION (MUW),F10.3,11X
     C30HSPECIFIED EXHAUST VELOCITY (C),F10.3,2X6HKM/SEC)
  202 FORMAT(/  2X,35HSPECIFIED POWERPLANT FRACTION (MUW),F10.3,11X
     C30HOPTIMIZED EXHAUST VELOCITY (C),F10.3,2X6HKM/SEC)
  203 FORMAT(/  2X,35HOPTIMIZED POWERPLANT FRACTION (MUW),F10.3,11X
     C30HSPECIFIED EXHAUST VELOCITY (C),F10.3,2X6HKM/SEC)
  204 FORMAT(/  2X,35HOPTIMIZED POWERPLANT FRACTION (MUW),F10.3,11X
     C30HOPTIMIZED EXHAUST VELOCITY (C),F10.3,2X6HKM/SEC)
  205 FORMAT(/  2X,15HCONE ANGLE FREE)
  206 FORMAT(/  2X,25HPERMISSIBLE CONE ANGLES  ,4F9.2)
  207 FORMAT(/  2X,25HOPTIMIZED CONE ANGLES    ,4F9.2)
  208 FORMAT(/  2X,19HCONSTANT ISP J (JC),F10.3,2X7HM2/SEC3,
     C18X15HTHRUST DURATION,F10.3,2X4HDAYS,
     C// 2X,25HFINAL MASS FRACTION (MUL),F10.3,
     C21X27HPAYLOAD MASS FRACTION (MUL),F10.3)
 1000 FORMAT(///2(6X4HTIME,3X9HMAGNITUDE,4X4HCONE,5X5HCLOCK,10X)//)
 1001 FORMAT(4F10.2,10X,4F10.2)
      RETURN
      END


      SUBROUTINE CDERV1(RO,P,PRO,NORDER)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      COMMON/COEFQ/CO(9),COEF(9),NCOEF
      COMMON/DER12/NDER12
      P=0.
      PRO=0.
      DO 10 I=1,NCOEF
      K=NCOEF+1-I
      P=RO*P+COEF(K)
      IF(I.LT.NCOEF) PRO=RO*PRO+P
```

```
10    CONTINUE
      NDER12=1
      CALL CDERV2(RC,P1,PRO1,NORDER)
      NDER12=0
      P=P+P1
      PRO=PRO+PRO1
      RETURN
      END


      SUBROUTINE CDERV2(RC,P,PRO,NORDER)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      COMMON/PXXPX/PX(31,6),PXX(31,31,6),X(30,6),DX(30,6),DUM1(2,6)
      COMMON/STATE/XO(30,6),ALO(3,2),PO
      COMMON/CHANGE/DX1(30,6),DX2(30,6),DAL1(3,2),DAL2(3,2)
      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),TACUM(7),T1(6),T2(6),T3(6),TP(6),TTP
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NO,NH,NAP
      COMMON/INDEX/N1(6),N2(6),NCP(6),NBLN(6),KV(3,2),KA(3,2),NA(2),
     CNMNB(2),LNNB(2),MMM(6),MIM(6),LM(6,2),IM(6,2),KM(6,2)
      COMMON/OPTION/NP1,NP2,NB1,NB2,NV1,NV2,NPO
      COMMON/DER12/NDER12
      COMMON/SKRCH1/XV(30),BAXV(30),WV(30),O(30),Q(30),G(90),
     CVX(6,2),VXA(6,3,2),VXAA(54,2),AL(3,2),DAL(3,2)
      RCS=.5*RO**2
      DO 1C NB=1,2
      DO 10 K=1,3
      AL(K,NB)=ALO(K,NB)+RO*DAL1(K,NB)+ROS*DAL2(K,NB)
      DAL(K,NB)=DAL1(K,NB)+RO*DAL2(K,NB)
      DO 8 M=1,6
8     VXA(M,K,NB)=0.
1C    CONTINUE
      DO 20 LN=1,NL
      DO 20 I=1,NPO
      X(I,LN)=XO(I,LN)+RO*DX1(I,LN)+ROS*DX2(I,LN)
      DX(I,LN)=DX1(I,LN)+RO*DX2(I,LN)
20    CONTINUE
      CALL ALIGN(X,AL,VX,VXA,VXAA)
      IF(NORDER.LT.1) GO TO 50
      DO 40 NB=1,2
```

```
      NAA=NA(NB)
      IF(NAA.EQ.0) GO TO 40
      LN=LNNB(NB)
      NP=NPNB(NB)
      DO 35 M=1,NP
      MM=MPM(M)
      K=KP(P,NB)
      Z=0.
      DO 30 M1=1,NAA
      K1=KA(M1,NB)
      Z=Z+VXA(MP,K1,NB)*DAL(K1,NB)
30    CONTINUE
      DX(K,LN)=Z
35    CONTINUE
40    CONTINUE
50    CONTINUE
      P=0.
      PRO=0.
      DO 90 LN=1,NL
      IF((NDER12.GT.0).AND.((LN.NE.1).OR.(NA(1).EQ.0)).AND.
     C((LN.NE.NL).OR.(NA(2).EQ.0))) GO TO 90
      CALL MULT3(V,X(1,LN),XV,ND,NP)
      CALL MULT4(BA,XV,BAXV,ND,NP,N,1)
      CALL WYDER
     C(TACLP(LN),T(LN),TP(LN),XV,BAXV,BAXV,WV,O,Q,G,G,G,Z,NORDER)
      P=P+Z*T3(LN)
      IF(NCRDER.LT.1) GO TO 90
      CALL MULT3(V,DX(1,LN),XV,ND,NP)
      CALL MULT4(BA,XV,BAXV,ND,NP,N,1)
      CALL MULT2(XV,O,Z,ND,NP)
      CALL MULT2(BAXV,Q,W,ND,NP)
      PRO=PRO+2.*T3(LN)*(W+Z)
90    CONTINUE
      RETURN
      END

      SUBROUTINE DERIV(T3,PX,PXX,PDIAG,PXSAV,W,O,Q,G,GD,H)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      COMMON/OPTION/NP1,NP2,NB1,NB2,NV1,NV2,NPO
```

66

```
      COMMON/PARAM/NL,NC,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NO,NH,NAP
      COMMON/SKRCH2/P(10),R(10,10),S(10,10)
      COMMON/CONS1/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
      COMMON/CONS2/B(10,10),C(10,10),D(10,10),E(10,10),F(10,10)
      DIMENSION PX(31),PXX(31,31),PDIAG(31),PXSAV(31)
      DIMENSION W(10),O(10,3),Q(10,3),G(10,3,3),GD(10,3,3),H(10,3,3)
      CALL MULT4(BA,Q,C,ND,NP,N,2)
      DO 49 M1=1,ND
      DO 39 I1=1,NP
   39 P(I1)=O(I1,M1)+Q(I1,M1)
      K1=-ND+M1
      DO 49 I1=1,NP
      K1=K1+ND
      PX(K1)=T3*(P(I1)+V(I1,1)*P(2)+V(I1,2)*P(NPM1))
   49 PXSAV(K1)=PX(K1)
      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)
      DO 100 M1=1,ND
      DO 99 M2=1,M1
      DO 98 NFLAG=1,2
      DO 89 I1=1,NP
      J2=NP
      IF(M1.EQ.M2) J2=I1
      DO 89 I2=1,J2
      IF(NFLAG.EQ.1) GO TO 84
      Z=S(I1,I2)+G(I1,M2,M1)*B(I1,I2)+B(I2,I1)*G(I2,M1,M2)
      IF(I1.EQ.I2) Z=Z+GD(I1,M1,M2)
      GO TO 88
   84 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(M1.EQ.M2)Z=Z+D(I1,I2)
   88 S(I1,I2)=Z
      IF(M1.EQ.M2) S(I2,I1)=Z
   89 CONTINUE
```

```
      DC 93 I1=1,NP
      J2=NP
      IF((M1.EQ.M2).AND.(I1.NE.2).AND.(I1.NE.NPM1)) J2=I1
      DC 93 I2=1,J2
   93 R(I1,I2)=S(I1,I2)+V(I2,1)*S(I1,2) +V(I2,2)*S(I1,NPM1)
      K1=-ND+M1
      DC 97 I1=1,NP
      K1=K1+ND
      J2=NP
      IF(M1.EQ.M2) J2=I1
      K2=-NC+M2
      DO 96 I2=1,J2
      K2=K2+ND
      Z= 3*(R(I1,I2)+V(I1,1)*R(2,I2)+V(I1,2)*R(NPM1,I2))
      IF(NFLAG.EQ.1) GO TO 95
      KL=MINO(K1,K2)
      KH=MAXO(K1,K2)
      PXX(KH,KL)=Z
      GC TC 96
   95 PXX(K1,K2)=Z
      PXX(K2,K1)=Z
   96 CONTINUE
      IF((NFLAG.EQ.1).AND.(M1.EQ.M2)) PDIAG(K1)=PXX(K1,K1)
   97 CCNTINUE
      IF(NC.EQ.2) GC TO 99
   98 CONTINUE
   99 CCNTINUE
  1CC CONTINUE
      RETURN
      END


      SUBRCUTINE EPHEM(D,N,V)
      IMPLICIT REAL*8(A-H,C-Z)                                        360 IN
      REAL E,ANGL,AYCU,ELEM
      COMMCN/ELEMNT/ELEM(7)
      COMMCN/ESTCFF/ANGL,AYCU
      COMMON/NCCNS/PI,PI2,PIF,GRAV,TCON,VELCON,POWCON,PCON
      CIMENSION V(12),E(7,17),F(7)
      CATA CELTA /1.E-14/
```

```
      DATA E /
     M.387C990,.2056270,.1222427,.8352647,1.340990,3.885480,-3065.0,
     V.7233320,.0067930,.0592405,1.332030,2.286526,3.042010,-3065.0,
     E1.CCCCOO,.0167260,0.000000,0.000000,1.784643,1.748089,-3065.0,
     M1.523691,.0933680,.0322870,.8595577,5.852485,4.516341,-3065.0,
     J5.202803,.0484350,.0227828,1.746105,.2387301,4.534909,-3065.0,
     S9.538843,.0556820,.0434571,1.977588,1.610319,4.898639,-3065.0,
     U19.18195,.0472090,.0134924,1.287988,2.967249,2.466237,-3065.0,
     N30.05778,.0085750,.0309578,2.292312,.7727262,3.786333,-3065.0,
     P39.43871,.2502360,.2996713,1.917866,3.912334,3.170326,-3065.0,
     D3.447700,.623C000,.3423000,2.425700,5.512300,5.512300, 5230.0,
     I1.CCCCOO,0.0CC000,0.CC0C00,0.000000,0.000000,0.000000, 0000.0,
     C2.767500,.0759000,.1851270,1.405230,2.659300,.9140460, 0952.5,
     S.1CCCCOO,0.00C000,0.000000,0.000000,0.000000,0.000000, 0000.0,
     X0.CCCCOO,C.CCCCOO,0.CC0000,0.000000,0.000000,0.000000, 0000.0,
     E1.458100,.223CC00,.1890000,5.3059C0,2.130500,1.971400,-9800.0,
     E2.218000,.8470000,.2085700,5.832700,2.795500,2.795500, 4580.0,
     H17.92900,.9670000,2.831797,1.012640,2.962871,2.962871, 6439.0/
      DC 1 I=1,7
1     F(I)=E(I,N)
      IF((ANGL.EQ.0.).AND.(AYOU.EQ.1.)) GO TO 5
3     IF((N.EQ.11).AND.(ANGL.NE.0.)) F(3)=ANGL/PIE
      IF((N.EQ.13).AND.(AYOU.NE.1.)) F(1)=AYOU
5     CONTINUE
      IF((N.NE.14).CR.(ELEM(1).EQ.0.)) GO TO 6
      DC 4 I=1,7
4     F(I)=ELEM(I)
6     CONTINUE
      AM=AMD(GRAV*(D-F(7))/(TCON*F(1)**1.5)+F(6)-F(5),PI2)
      AE=AM
      DC 1CO I=1,25
      B=AE-(AE-F(2)*SN(AE)-AM)/(1.-F(2)*CN(AE))
      IF(AB(B-AE).LT.DELTA) GO TO 200
1CC   AE=B
2CC   AT=2.*ATN2(SQT(1.+F(2))*SN(AE/2.),SQT(1.-F(2))*CN(AE/2.))
      P=F(1)*(1.-F(2)**2)
      R=P/(1.+F(2)*CN(AT))
      RT=GRAV*F(2)*SN(AT)/(R*SQT(P))
      A=GRAV*SQT(P)/R
      L=F(5)-F(4)+AT
```

IF((N.NE.11).AND.(ANGL.NE.0.)) EPOK11 = D
IF((N.EQ.11).AND.(ANGL.NE.0.)) F(7) = EPOK11

```
      V(1)=R*(CN(U)*CN(F(4))-SN(U)*CN(F(3))*SN(F(4)))
      V(2)=R*(CN(U)*SN(F(4))+SN(U)*CN(F(3))*CN(F(4)))
      V(3)=R*SN(U)*SN(F(3))
      V(4)=RT*V(1)-A*(SN(U)*CN(F(4))+CN(U)*CN(F(3))*SN(F(4)))
      V(5)=RT*V(2)-A*(SN(U)*SN(F(4))-CN(U)*CN(F(3))*CN(F(4)))
      V(6)=RT*V(3)+A*(CN(U)*SN(F(3)))
      GRAVR=GRAV**2/R**3
      DO 300 M=1,3
      V(M+6)=-GRAVR*V(M)
  300 V(M+9)=-GRAVR*(V(M+3)-3.*V(M)*RT)
      RETURN
      END


      FUNCTION ETA(C)
      IMPLICIT REAL*8(A-H,O-Z)                                        360 IN
      REAL BB,DD
      COMMON/THRUST/BB,DD
      ETA=BB/(1.+(DD/C)**2)
      RETURN
      END


      SUBROUTINE INDCAL(NBD1,NBD2,NVL1,NVL2,NBFIX)
      IMPLICIT REAL*8(A-H,O-Z)                                        360 IN
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NO,NH,NAP
      COMMON/OPTION/NP1,NP2,NB1,NB2,NV1,NV2,NPO
      COMMON/INDEX/N1(6),N2(6),NCP(6),NBLN(6),KV(3,2),KA(3,2),NA(2),
     CNMNB(2),LNNB(2),MMM(6),MIM(6),LM(6,2),IM(6,2),KM(6,2)
      NB1=NBD1
      NB2=NBD2
      IF((NBD1.EQ.1).AND.(NBD2.NE.1)) NB2=1
      IF((NBD2.EQ.1).AND.(NBD1.NE.1)) NB1=1
      NV1=NVL1
      NV2=NVL2
      IF(NBFIX.NE.1) GO TO 15
      NB1=2
      NB2=2
      IF(NVL1.EQ.1) NV1=2
      IF(NVL2.EQ.1) NV2=2
```

```
15    NAP=C
      IF((NB1.EC.1).OR.(NB2.EQ.1)) NAP=1
      DC 20 LN=1,NL
      NBLN(LN)=1+(LN/NL)-(LN/(NL**2))
      N1(LN)=1
      N2(LN)=NPC
2C    NCP(LN)=ND2
      DO 22 M=1,ND2
      MPM(M)=M+(M/(NC+1))*(1-(ND/3))
22    MIM(M)=M-(M/(NC+1))*NC
      DO 3C NB=1,2
      LMNB(NB)=(NB-1)*NL+(2-NB)
      NMNB(NB)=ND2
      IF((NB.EQ.1.AND.NV1.EC.0).OR.(NB.EQ.2.AND.NV2.EQ.0)) NMNB(NB)=ND
      NBS=3-2*NB
      NVP=(NB-1)*NV2+(2-NB)*NV1
      NBP=(NB-1)*NB2+(2-NB)*NB1
      NPD2P=(NB-1)*NPD2+(2-NB)*(ND2+1)
      NVK=((ND-2)*NVP**2+(4-NC)*NVP+2*ND)/2-(1-NBP)*(1-NBP/2)
      IF(NB.EQ.1) N1(1)=NPD2P-(ND2-NVK)
      IF(NB.EQ.2) N2(NL)=NPD2P+(ND2-NVK)
      NA(NB)=NVP*(2-NVP)*(ND-1)+(1-NBP/2)
      DC 25 I=1,3
      KA(I,NB)=I+NBP/2
25    KV(I,NB)=NPD2P-NBS*(NA(NB)+1-I+ND*(1-NVP)*(1-NVP/2))
      DC 30 M=1,ND2
      LM(M,NB)=M+(NB-1)*(NPC1-ND2*(M/(ND+1)))
      IM(M,NB)=1+((LM(M,NB)-MIM(M))/ND)
30    KM(M,NB)=IM(M,NB)+NP*(MIM(M)-1)
      RETURN
      END


      SUBRCUTINE KOPTR
     C(PHE,NNN,PHEMAX,PV,T,BOT,WMU,C,ALPHAW,ST,NST,WMUL,P3)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      DIMENSICN PHE(301),ST(30)
      CCMMCN/NCONS/PI,PI2,PIF,GRAV,TCCN,VELCON,PCWCON,PCON
      CATA TAU,CELTA1,DELTA2 /1.618034,.001,.00005/
      RC=0.
```

```
      P=1.E20
      RC2=PHEMAX/(1.+TAU)
      RC1=RC
      P1=P
      DO 15 I=1,20
      WK2=PHEMAX-RC2
      CALL KROOT(PHE,NNN,BOT,WK2,PVTEST,WMU1,ST,NST,TEST)
      C=PV/(PVTEST*VELCON)
      WMU=ALPHAW*BOT*C**2/(2.*POWCON*T*ETA(C))
      P2=-WPUL(WMU,C,ALPHAW,WMU1)
      IF(TEST.LE.0.) P2=1.E25
      IF(P2-P) 10,20,20
10    RC1=RC
      P1=P
      RC=RC2
      P=P2
15    RC2=RC1+(TAU+1.)*(RC-RC1)
20    DO 90 I=1,20
      RC3=RC1+(RC2-RC1)/TAU
      WK3=PHEMAX-RC3
      CALL KROOT(PHE,NNN,BOT,WK3,PVTEST,WMU1,ST,NST,TEST)
      C=PV/(PVTEST*VELCON)
      WMU=ALPHAW*BOT*C**2/(2.*POWCON*T*ETA(C))
      P3=-WPUL(WMU,C,ALPHAW,WMU1)
      IF(TEST.LE.0.) P3=1.E25
      IF(P-P3) 60,50,70
50    IF(RC2.LT.RC1) GO TO 70
60    RC2=RC1
      P2=P1
      RC1=RC3
      P1=P3
      GO TO 80
70    RC1=RC
      P1=P
      RC=RC3
      P=P3
80    IF((AB(P1-P).LT.DELTA1).AND.(AB(P2-P).LT.DELTA1)) GO TO 95
90    CONTINUE
95    IF((AB(P1-P).LT.DELTA2).AND.(AB(P2-P).LT.DELTA2)) GO TO 99
      A=(RC-RC2)*P1+(RC1-RC)*P2+(RC2-RC1)*P
```

```
      B=(RC-RO2)*(RC+RO2)*P1+(RO1-RO)*(RO1+RC)*P2+
     C(RC2-RO1)*(RC2+RO1)*P
      RC3=B/(2.*A)
98    CONTINUE
      WK3=PHEMAX-RO3
      CALL KRCOT(PHE,NNN,BOT,WK3,PVTEST,WMU1,ST,NST,TEST)
      C=PV/(PVTEST*VELCON)
      WMU=ALPHAW*BOT*C**2/(2.*POWCON*T*ETA(C))
      P3=-WMLL(WMU,C,ALPHAW,WMU1)
      IF(TEST.LT.0.) P3=1.E25
      IF(P3.LT.P) RETURN
99    IF(RC3.EC.RC) RETURN
      RC3=RC
      GC TC 98
      END


      SUBROUTINE KRCOT(PHE,N,BZ,WKS,PV,WM1,ST,NST,TEST)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      DIMENSICN PHE(301),ST(30)
      DT=1./FLT(N-1)
      BDT=BZ*DT
      WMO=1.
      SC=C.
      WKO=PHE(1)-WKS
      NST=1
      ST(1)=0.
      B=0.
      IF(WKO.LT.0.) GO TO 7
      ST(1)=1.
      NST=C
      B=BZ
      WKO=BDT*WKO
7     DC 1CO I=2,N
      IF(NST.GT.29) GO TO 120
      DPHE=PHE(I)-PHE(I-1)
      IF(B.EQ.0.) GC TC 50
      WM=WMC-BDT
      IF((WM.LE.0.).AND.(DPHE.GE.0.)) GO TO 120
      IF(WM.LE.C.) GC TC 20
```

```
      DL=ALN(WM/WMO)
      WK=WKO-DPHE*DL
      IF(WK) 20,10,10
10    S=SO-(DPHE+DL*(PHE(I-1)+DPHE*WMO/BDT))
      GO TO 90
20    NST=NST+1
      DL=WKO/DPHE
      WM=1.E-20
      IF(DL.GT.(-46.0517)) WM=WMO*EX(DL)
      X=(WMO-WM)/BDT
      ST(NST)=(FLT(I-2)+X)*DT
      S=SO-(DPHE*X+DL*(PHE(I-1)+DPHE*WMO/BDT))
      WK=DPHE*(1.-X)
      B=0.
      GO TO 90
50    WK=WKO+DPHE
      IF(WK) 95,95,60
60    NST=NST+1
      X=1.+WKO/DPHE
      ST(NST)=(FLT(I-1)-X)*DT
      WM=WMO-BDT*X
      IF(WM.LE.0.) GO TO 120
      DL=ALN(WM/WMO)
      S=SO-(DPHE*X+DL*(PHE(I)+DPHE*WM/BDT))
      WK=-DPHE*DL
      B=BZ
90    WMO=WM
      SO=S
95    WKO=WK
100   CONTINUE
120   WM1=WM
      PV=S
      IF(B.EQ.0.) TEST=(PHE(N)-WK)/WM
      IF(B.GT.0.) TEST=PHE(N)/WM-WK/BDT
      IF(WM.LE.0.) TEST=-1.
      RETURN
      END


      SUBROUTINE LSERCH(RCO,PO,PRCO,RO,P,PRO,IDERIV)
```

```
      IMPLICIT REAL*8(A-H,C-Z)                                          360 IN
      DATA TAU,DELTA /1.61803395,.01/
      RB=1.
      RC1=RCO
      P1=PC
      PRC1=PRCO
      PRCSAV=PRC1
      RC2=RC
      P2=P
      PRC2=PRO
      DC 10 I=1,5
      IF(PRC2) 5,5,12
    5 RC1=RC2
      P1=P2
      PRC1=PRO2
      RC2=RC1+(RB-RC1)/2.
      CALL DDERIV(RC2,P2,PRC2,1)
   10 CONTINUE
      IF(P2.GT.PO) GC TC 45
      RC=RC2
      P=P2
      PRC=PRO2
      GC TC 100
   12 CONTINUE
      DC 20 I=1,20
      RC=(RC2+RC1)/2.
      A=(PRC2/PRC1)**2
      IF((A.GT.DELTA).AND.(A.LT.(1./DELTA)))
     CRC=(RC1*PRC2-RC2*PRC1)/(PRO2-PRO1)
      CALL DDERIV(RC,P,PRC,1)
      IF(PRO) 16,18,18
   16 RC1=RC
      P1=P
      PRO1=PRO
      GC TC 19
   18 RC2=RC
      P2=P
      PRO2=PRO
   19 A=(PRC/PRCSAV)**2
      IF(A.LT.DELTA) GC TO 21
```

```
20    CONTINUE
21    IF(P0-P) 40,40,100
40    RC2=RC
      P2=P
45    RC1=RC0
      P1=P0
      RC3=RC1+(RO2-RC1)*(TAU-1.)
      CALL CDERIV(RC3,P3,PRO3,0)
      DC 90 I=1,20
      RC=RC1+(RC2-RC1)*(2.-TAU)
      CALL ODERIV(RC,P,PRO,0)
      IF((AB(1.-P/P1).LT.DELTA).AND.(AB(1.-P/P2).LT.DELTA).AND.
     C(P.LE.P0)) GC TC 100
      IF(P3.GT.P1) GO TO 70
      IF(P3-P) 60,60,70
60    RC1=RC2
      P1=P2
      RC2=RO
      P2=P
      GO TC 90
70    RO2=RO3
      P2=P3
      RC3=RO
      P3=P
90    CONTINUE
100   CONTINUE
      RETURN
      END


      SUBROUTINE MCOPTR
     C(PHE,NNN,PHEMAX,PV,T,RO3,WMU,C,ALPHAW,ST,NST,WMU1)
      IMPLICIT REAL*8(A-H,O-Z)                                        360 IN
      DIMENSION PHE(301),ST(30)
      DATA TAU,DELTA1,DELTA2 /1.618034,.005,.00001/
      RC=0.
      P=1.E20
      RC2=.1*TAL
      RC1=RC
      P1=P
```

```
      DC 15 I=1,20
      CALL KOPTR(PHE,NNN,PHEMAX,PV,T,RO2,WMU,C,ALPHAW,ST,NST,WMU1,P2)
      IF(P2-P) 10,2C,20
10    RO1=RO
      P1=P
      RC=RC2
      P=P2
15    RC2=RC1+(TAU+1.)*(RO-RO1)
20    DC 90 I=1,20
      RC3=RC1+(RO2-RC1)/TAU
      CALL KOPTR(PHE,NNN,PHEMAX,PV,T,RO3,WMU,C,ALPHAW,ST,NST,WMU1,P3)
      IF(P-P3)60,60,70
60    RO2=RO1
      P2=P1
      RC1=RC3
      P1=P3
      GC TC 80
70    RC1=RC
      P1=P
      RC=RC3
      P=P3
80    IF((AB(P1-P).LT.DELTA1).AND.(AB(P2-P).LT.DELTA1)) GO TO 95
9C    CCNTINUE
95    IF((AB(P1-P).LT.DELTA2).AND.(AB(P2-P).LT.DELTA2)) GC TO 99
      A=(RC-RO2)*P1+(RO1-RO)*P2+(RO2-RO1)*P
      B=(RC-RO2)*(RC+RC2)*P1+(RO1-RO)*(RO1+RO)*P2+
     C(RO2-RO1)*(RO2+RO1)*P
      RC3=B/(2.*A)
98    CCNTINUE
      CALL KOPTR(PHE,NNN,PHEMAX,PV,T,RO3,WMU,C,ALPHAW,ST,NST,WMU1,P3)
      IF(P3.LT.P) RETURN
99    IF(RC3.EQ.RO) RETURN
      RO3=RO
      GC TC 98
      END


      SUBROUTINE MODIFY(PX,DX,DAL,NORDER)
      IMPLICIT REAL*8(A-H,C-Z)                                        360 IN
      COMMCN/TIMEC/TT,T(6),TACUM(7),T1(6),T2(6),T3(6),TP(6),TTP
```

```
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NO,NH,NAP
      COMMON/INDEX/N1(6),N2(6),NCP(6),NBLN(6),KV(3,2),KA(3,2),NA(2),
     CNMNB(2),LNNB(2),MMM(6),MIM(6),LM(6,2),IM(6,2),KM(6,2)
      COMMON/BDYQ/VX(6,2),VXA(6,3,2),VXAA(6,9,2)
      COMMON/SKRCH2/XVA(10,3,2),VXA1(6,3,2),VX1(6,2),DUM2(102)
      COMMON/OPTION/NP1,NP2,NB1,NB2,NV1,NV2,NPO
      DIMENSION PX(31,6),DX(10,3,6),DAL(3,2)
      DO 5 M=1,NC
      DO 4 LN=1,NL
      DO 4 I=1,NP
      K=(I-1)*NC+M
    4 DX(I,M,LN)=-PX(K,LN)
      IF(NV1.NE.0) DX(2,M,1)=0.
      IF(NV2.NE.0) DX(NPM1,M,NL)=0.
      DX(1,M,1)=0.
    5 DX(NP,M,NL)=0.
      IF(NL.EQ.1) GO TO 80
      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)
   80 KA1=KA(1,1)
      KA2=KA(1,2)
      DO 100 NBP=1,2
      NB=3-NBP
      NAA=NA(NB)
      IF(NAA.EQ.0) GO TO 100
      NP=NPNB(NB)
      LN=LNNB(NB)
      NBL=NBLN(LN)
      DO 45 M=1,NAA
      K=KA(M,NB)
      L=KV(M,NB)
      DAL(K,NB)=-PX(L,LN)
   45 CONTINUE
      IF((NAP.EQ.1).AND.(NB.EQ.1)) DAL(KA1,1)=DAL(KA2,2)
      DO 98 M=1,NP
      MM=MMM(M)
      MI=MIM(M)
```

```
      I=IM(M,NB)
      Z1=0.
      Z2=0.
      DO 94 M1=1,NAA
      K1=KA(M1,NB)
      Z1=Z1+VXA(MM,K1,NB)*DAL(K1,NB)
      IF(NORDER.EQ.2) GO TO 94
      Z=0.
      DO 93 M2=1,NAA
      K2=KA(M2,NB)
      K3=K1+3*(K2-1)
   93 Z=Z+VXAA(MM,K3,NB)*DAL(K2,NB)
      VXA1(MM,K1,NB)=Z
      Z2=Z2+DAL(K1,NB)*VXA1(MM,K1,NB)
   94 CONTINUE
      IF(NORDER.EQ.1) VX1(MM,NB)=Z2
      XVA(I,MI,NBL)=Z2
      IF(NORDER.EQ.2) Z1=Z1+VX1(MM,NB)
   98 DX(I,MI,LN)=Z1
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MODLEG(T,NL,S,LN)
      IMPLICIT REAL*8(A-H,O-Z)                                        360 IN
      DIMENSION T(6)
      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 MOPTR(PHE,NNN,PHEMAX,PV,T,R03,C,ALPHAW,ST,NST,WMU1)
      IMPLICIT REAL*8(A-H,O-Z)                                        360 IN
```

```
      DIMENSION PHE(301),ST(30)
      DATA TAU,DELTA1,DELTA2 /1.618034,.005,.00001/
      RC=0.
      P=1.E20
      RO1=RO
      P1=P
      RC2=TAU/10.
      DO 15 I=1,20
      CALL ROOTR(PHE,NNN,PHEMAX,PV,T,RO2,C,ALPHAW,ST,NST,WMU1)
      P2=-WMUL(RO2,C,ALPHAW,WMU1)
      IF(P2-P) 10,20,20
10    RO1=RO
      P1=P
      RC=RC2
      P=P2
15    RC2=RO1+(TAU+1.)*(RO-RO1)
20    DO 90 I=1,20
      RC3=RC1+(RC2-RC1)/TAU
      CALL ROOTR(PHE,NNN,PHEMAX,PV,T,RO3,C,ALPHAW,ST,NST,WMU1)
      P3=-WMUL(RO3,C,ALPHAW,WMU1)
      IF(P-P3) 60,60,70
60    RO2=RO1
      P2=P1
      RC1=RC3
      P1=P3
      GO TO 80
70    RC1=RO
      P1=P
      RO=RC3
      P=P3
80    IF((AB(P1-P).LT.DELTA1).AND.(AB(P2-P).LT.DELTA1)) GO TO 95
90    CONTINUE
95    IF((AB(P1-P).LT.DELTA2).AND.(AB(P2-P).LT.DELTA2)) GO TO 99
      A=(RC-RO2)*P1+(RC1-RC)*P2+(RO2-RO1)*P
      B=(RC-RO2)*(RC+RO2)*P1+(RO1-RC)*(RO1+RO)*P2+
     C(RO2-RO1)*(RC2+RO1)*P
      RC3=B/(2.*A)
98    CONTINUE
      CALL ROOTR(PHE,NNN,PHEMAX,PV,T,RO3,C,ALPHAW,ST,NST,WMU1)
      P3=-WMUL(RO3,C,ALPHAW,WMU1)
```

```
      IF(P3.LT.P) RETURN
   99 IF(RC3.EQ.RC) RETURN
      RC3=RC
      GC TC 98
      END


      SUBROUTINE MULT2(X,Y,Z,ND,NP)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      DIMENSION X(10,3),Y(10,3)
      Z=0.
      DC 1 M=1,ND
      DC 1 I=1,NP
    1 Z=Z+X(I,M)*Y(I,M)
      RETURN
      END


      SUBROUTINE MULT3(V,X,Y,ND,NP)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      DIMENSION V(10,2),X(10,3),Y(10,3)
      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)
      IMPLICIT REAL*8(A-H,C-Z)                                    360 IN
      DIMENSION A(5,5,2),X(10,3),Y(10,3),U1(5),U2(5)
      DC 99 M=1,ND
      DC 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)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      DIMENSION A(5,5,2),B(5,5,2),C(5,5,2),D(5,5,2)
      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 C(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)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      DIMENSION 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)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      DIMENSION 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 PATCH(PX,PXX,PXA,PDG,NORDER,NOPT)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NO,NH,NAP
      COMMON/TIMEQ/TT,T(6),TACUM(7),T1(6),T2(6),T3(6),TP(6),TTP
      COMMON/INDEX/N1(6),N2(6),NCP(6),NBLN(6),KV(3,2),KA(3,2),NA(2),
     CNMNB(2),LNNB(2),MMM(6),MIM(6),LM(6,2),IM(6,2),KM(6,2)
      COMMON/BDYQ/VX(6,2),VXA(6,3,2),VXAA(6,9,2)
      COMMON/OPTION/NP1,NP2,NB1,NB2,NV1,NV2,NPD
      COMMON/SKRCH1/SAV1(30,3),SAV2(3),DUM8(315)
      COMMON/SKRCH2/XVA(30,2),VXA1(6,3,2),VX1(6,2),DUM2(102)
      DIMENSION PX(31,6),PXX(31,31,6),PXA(31,6),PDG(31,6)
      DO 49 NB=1,2
      NAA=NA(NB)
      IF(NAA.EQ.0) GO TO 49
      NP=NPNB(NB)
      LN=LNNB(NB)
      DO 23 M1=1,NAA
      K1=KA(M1,NB)
      Z=0.
      DO 22 M=1,NM
      MM=MMM(M)
      L=LM(M,NB)
      Z=Z+PX(L,LN)*VXA(MM,K1,NB)
      IF(NOPT.EQ.1) Z=Z+PXA(L,LN)*VXA1(MM,K1,NB)
      IF((NOPT.EQ.1).AND.(NH.GT.2).AND.(K1.NE.1))
     CZ=Z+PDG(L,LN)*VXA1(MM,K1,NB)
   22 CONTINUE
   23 SAV2(M1)=Z
      IF(NORDER.LT.2) GO TO 40
      DO 29 M1=1,NAA
      K1=KA(M1,NB)
      DO 29 I=1,NPO
      Z=0.
      DO 27 M=1,NM
      MM=MMM(M)
      K=MAXO(I,LM(M,NB))
      L=MINO(I,LM(M,NB))
   27 Z=Z+PXX(K,L,LN)*VXA(MM,K1,NB)
```

```
29    SAV1(L,M1)=Z
      DO 39 M1=1,NAA
      K1=KA(M1,NB)
      DO 39 M2=1,M1
      K2=KA(M2,NB)
      K3=K1+3*(K2-1)
      Z=0.
      DO 37 M=1,NM
      MM=MMM(M)
      L=LM(M,NB)
      IF((NC.EQ.3).OR.((NH.EQ.3).AND.(K3.NE.1)))
     CZ=Z+PX(L,LN)*VXAA(MM,K3,NB)
37    Z=Z+SAV1(L,M1)*VXA(MM,K2,NB)
      L1=MAXO(KV(M1,NB),KV(M2,NB))
      L2=MINO(KV(M1,NB),KV(M2,NB))
39    PXX(L1,L2,LN)=Z
40    CONTINUE
      DO 46 M1=1,NAA
      L=KV(M1,NB)
      PX(L,LN)=SAV2(M1)
      IF(NORDER.LT.2) GO TO 46
      NMPD=NPD-NM
      DO 45 I=1,NMPD
      K=I+(2-NB)*NM
      L1=MAXO(L,K)
      L2=MINO(L,K)
45    PXX(L1,L2,LN)=SAV1(K,M1)
46    CONTINUE
49    CONTINUE
      IF(NL.EQ.1) GO TO 100
      DO 150 LN=2,NL
      Q=T(LN-1)
      R=T(LN)
      IF(NORDER.LT.2) GO TO 120
      DO 110 J=1,NPD2
      ND2J=ND2+J
      DO 110 I=1,ND
      NDI=ND+I
      NPD1I=NPD1+I
      NPD2I=NPD2+I
```

```
      PXX(ND2J,NDI,LN)=PXX(ND2J,NDI,LN)*R
      Z=PXX(NPD1I,J,LN-1)
      PXX(NPD1I,J,LN-1)=-PXX(NPD2I,J,LN-1)*Q
  110 PXX(NPD2I,J,LN-1)=Z
  120 CONTINUE
      DO 140 I=1,ND
      NPD1I=NPD1+I
      NPD2I=NPD2+I
      NDI=ND+I
      Z=PX(NPD1I,LN-1)+PX(I,LN)
      PX(NPD1I,LN-1)=-PX(NPD2I,LN-1)*Q+PX(NDI,LN)*R
      PX(NPD2I,LN-1)=Z
      IF(NORDER.LT.2) GO TO 140
      DO 130 J=1,I
      NDJ=ND+J
      NPD1J=NPD1+J
      NPD2J=NPD2+J
      Z=PXX(NPD1I,NPD1J,LN-1)+PXX(I,J,LN)
      PXX(NPD1I,NPD1J,LN-1)=PXX(NPD2I,NPD2J,LN-1)*Q**2
     C+PXX(NDI,NDJ,LN)*R**2
      PXX(NPD2I,NPD2J,LN-1)=Z
      Z=PXX(NDI,J,LN)*R-PXX(NPD1J,NPD2I,LN-1)*Q
      IF(J.EQ.I) GO TO 130
      PXX(NPD1J,NPD2I,LN-1)=PXX(NDJ,I,LN)*R-PXX(NPD1I,NPD2J,LN-1)*Q
  130 PXX(NPD1I,NPD2J,LN-1)=Z
  140 CONTINUE
  150 CONTINUE
  160 CONTINUE
      IF(NAP.EQ.0) RETURN
      L1=KV(1,1)
      L2=KV(1,2)
      PX(L2,NL)=PX(L2,NL)+PX(L1,1)
      IF(NORDER.LT.2) RETURN
      PXX(L2,L2,NL)=PXX(L2,L2,NL)+PXX(L1,L1,1)
      IF(NL.EQ.1) PXX(L2,L2,NL)=PXX(L2,L2,NL)+2.*PXX(L2,L1,1)
      DO 90 LN=1,NL
      NCP2=N2(LN)
      NCP1=N1(LN)
      IP1=NCP2+1
      IM1=NCP1-1
```

```
      DO 70 J=NCP1,NCP2
      IF(NL.EQ.1) PXX(IP1,J,LN)=PXX(IP1,J,LN)+PXX(J,IM1,LN)
      IF((LN.GT.1).AND.(LN.LT.NL)) PXX(IP1,J,LN)=0.
      IF((NL.GT.1).AND.(LN.EQ.1)) PXX(IP1,J,LN)=PXX(J,IM1,LN)
70    CONTINUE
      IF(LN.EQ.1) GO TO 90
      NCPL=NCP(LN)
      IPM1=N2(LN-1)+1
      DO 80 J=1,NCPL
      JLN=NCP1-1+J
      JLNM1=N2(LN-1)-NCP(LN)+J
      PXX(IPM1,JLNM1,LN-1)=PXX(IPM1,JLNM1,LN-1)+PXX(IP1,JLN,LN)
80    CONTINUE
90    CONTINUE
      RETURN
      END


      SUBROUTINE PDERIV(NORDER)
      IMPLICIT REAL*8(A-H,O-Z)                                        360 IN
      COMMON/STATE/X(30,6),AL(3,2),P
      COMMON/BDYQ/VX(6,2),VXA(18,2),VXAA(54,2)
      COMMON/PXXPX/PX(31,6),PXX(31,31,6),PXSAV(31,6),PDIAG(31,6)
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NO,NH,NAP
      COMMON/TIMEQ/TT,T(6),TACUM(7),T1(6),T2(6),T3(6),TP(6),TTP
      COMMON/OUT/XV(30,6),AV(30,6),WV(30,6)
      COMMON/CONS1/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
      COMMON/SKRCH1/BAXV(30),O(30),Q(30),G(90),GD(90),H(90),DUM1(48)
      CALL ALIGN(X,AL,VX,VXA,VXAA)
      P=0.
      DO 5 LN=1,NL
      CALL MULT3(V,X(1,LN),XV(1,LN),ND,NP)
      CALL MULT4(BA,XV(1,LN),BAXV,ND,NP,N,1)
      CALL WYDER
     C(TACUM(LN),T(LN),TP(LN),XV(1,LN),BAXV,AV(1,LN),WV(1,LN),
     CC,C,G,GD,H,PL,NORDER)

      IF(NORDER.EQ.0) GO TO 5
      CALL DERIV
     C(T3(LN),PX(1,LN),PXX(1,1,LN),PDIAG(1,LN),PXSAV(1,LN),WV(1,LN),
```

120

```
     CC,Q,G,GD,H)
5     P=P+PL*T3(LN)
      RETURN
      END


      FUNCTION PMAP(H)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NO,NH,NAP
      COMMON/OUT/XV(30,6),AV(30,6),WV(30,6)
      COMMON/TIMEQ/TT,T(6),TACUM(7),T1(6),T2(6),T3(6),TP(6),TTP
      S=H
      CALL MODLEG(T,NL,S,LN)
      CALL POLEVL(WV(1,LN),G,S,1,NP,N,0)
      PMAP=1./G**2
      RETURN
      END

      SUBROUTINE POLEVL(Y,G,S,ND,NP,N,NO)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      DIMENSION Y(10,3),H(10),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)
      IF(NC.EQ.0) GO TO 3
      H(1)=0.
      W=H(2)
      H(2)=2.
      DO 2 I=3,NP
      X=H(I)
      H(I)=Z*H(I-1)-H(I-2)+4.*W
2     W=X
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(P)=.5*Z
      RETURN
      END


      SUBROUTINE POWER(R,T,NPO,NC,PO,P1,P2)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      REAL TILT,PLOSS
      COMMON/ARRAY/TILT,PLOSS
      COMMON/NCONS/PI,PI2,PIF,GRAV,TCON,VELCON,POWCON,PCON
      DATA A1,A2,A3,A4,A5 /.627,5.3054,-10.0376,7.1073,-2.0021/
      DATA RMIN,PMAX /.66,1.396/
      FACTOR=ABS(SIN(TILT/PIF))*EXP(-PLOSS*T)
      IF((R.LT.RMIN).OR.(NPO.EQ.0)) GO TO 30
      SR=1./SQT(R)
      PC=(A1+SR*(A2+SR*(A3+SR*(A4+SR*(A5)))))/R**2
      IF(NC.LT.1) GO TO 40
      P1=-(2.*A1+SR*(2.5*A2+SR*(3.*A3+SR*(3.5*A4+SR*(4.*A5)))))/R**3
      P1=FACTOR*P1
      IF(NC.LT.2) GO TO 40
      P2=(6.*A1+SR*(8.75*A2+SR*(12.*A3+SR*(15.75*A4+SR*(20.*A5)))))/R**4
      P2=FACTOR*P2
      GO TO 40
30    PO=PMAX
      IF(NPC.EQ.0) PO=1.
      P1=0.
      P2=0.
40    CONTINUE
      PO=FACTOR*PO
      RETURN
      END


      SUBROUTINE PXS(XO,DX1,XA,AV,WV,PX,PXA,PDG,TP,T3,NA)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      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,NO,NH,NAP
      COMMON/SKRCH1/R1(10),R2(10),CO1(10),C11(10),GO(10,3),G1(10,3),
     CG2(10,3),GA(10,3),CO(10,3),O1(10,3),O2(10,3),AV1(10,3),X1(10,3),
     CPL(10,3),CO(3),C1(3),CA(3),DUM1(59)
```

```
      DIMENSION XO(10,3),DXI(10,3),XA(10,3),AV(10,3),WV(10,3),
     CPX(31),PXA(31),PDG(31)
      FNH=FLT(NH-2)
      CALL MULT3(V,CX1,X1,ND,NP)
      CALL MULT4(BA,X1,AV1,ND,NP,N,1)
      IF(NA.GT.0) CALL MULT3(V,XA,XA,ND,NP)
      IF(NA.GT.0) CALL MULT4(BA,XA,GA,ND,NP,N,1)
      DO 39 I=1,NP
      Z=0.
      DO 8 M=1,ND
    8 Z=Z+XO(I,M)**2
      R2(I)=1./Z
      R1(I)=SQT(R2(I))
      R2(I)=TP*R2(I)
      Z01=0.
      Z11=0.
      COA=0.
      DO 19 M=1,ND
      CO(M)=R1(I)*XO(I,M)
      C1(M)=R1(I)*X1(I,M)
      CA(M)=0.
      IF(NA.GT.0) CA(M)=R1(I)*XA(I,M)
      COA=COA+CO(M)*CA(M)
      Z01=Z01+CO(M)*C1(M)
   19 Z11=Z11+C1(M)**2
      CO1(I)=Z01
      C11(I)=Z11
      W0=WV(I,1)
      W1=WV(I,2)*CO1(I)
      W2=WV(I,2)*C11(I)+WV(I,3)*CO1(I)**2
      DO 39 M=1,ND
      GO(I,M)=W0*AV(I,M)
      AV1(I,M)=AV1(I,M)+R2(I)*(C1(M)-3.*CO1(I)*CO(M))
      G1(I,M)=W1*GO(I,M)+W0*AV1(I,M)
      G2(I,M)=W2*GO(I,M)+2.*W1*W0*AV1(I,M)+W0*
     C(3.*R2(I)*((5.*CO1(I)**2-C11(I))*CO(M)-2.*CO1(I)*C1(M)))
      IF(NA.EQ.0) GA(I,M)=0.
      IF(NA.GT.0) GA(I,M)=W0*(GA(I,M)+R2(I)*(CA(M)-3.*COA*CO(M)))+
     CCCA*WV(I,2)*GO(I,M)
   39 CONTINUE
```

```
      DO 100 IA=1,3
      DO 49 I=1,NP
      DO 49 M=1,ND
      IF(IA.EQ.1) OO(I,M)=G1(I,M)+GO(I,M)
      IF(IA.EQ.1) C1(I,M)=GA(I,M)+G2(I,M)+G1(I,M)
      IF(IA.EQ.2) OC(I,M)=0.
      IF(IA.EC.2) O1(I,M)=G1(I,M)+GO(I,M)
      IF(IA.EQ.3) CO(I,M)=GO(I,M)
      IF(IA.EQ.3) O1(I,M)=G1(I,M)
49    CONTINUE
      CALL MULT4(AA,CO,OO,ND,NP,N,1)
      CALL MULT4(AA,C1,O1,ND,NP,N,1)
      DO 59 I=1,NP
      WC=WV(I,1)
      W1=WV(I,2)*CO1(I)
      DO 59 M=1,ND
      CO(I,M)=WC*OO(I,M)
      C1(I,M)=WO*C1(I,M)
      C2(I,M)=W1*CO(I,M)+O1(I,M)
59    CONTINUE
      CALL MULT4(BA,C2,G2,NC,NP,N,2)
      DO 80 I=1,NP
      OOCC=0.
      C2CC=0.
      CCC1=0.
      OCAV=0.
      COAV1=0.
      C1AV=0.
      DO 70 M=1,ND
      CO(M)=R1(I)*XO(I,M)
      C1(M)=R1(I)*X1(I,M)
      OO(I,M)=R1(I)*CO(I,M)
      C1(I,M)=R1(I)*O1(I,M)
      C2(I,M)=R1(I)*O2(I,M)
      OOCO=COCO+CO(I,M)*CO(M)
      O2CO=O2CO+O2(I,M)*CO(M)
      COC1=OOC1+CO(I,M)*C1(M)
      COAV=COAV+CO(I,M)*AV(I,M)
      OCAV1=OOAV1+OC(I,M)*AV1(I,M)
      O1AV=C1AV+O1(I,M)*AV(I,M)
```

```
70    CONTINUE
      CC0=-3.*C01(I)
      CC0=3.*R2(I)*(5.*C01(I)*D0C0-U2C0-D0C1)+
     CWV(I,3)*C01(I)*C0AV+WV(I,2)*(D0AV1+D1AV)
      CC1=-3.*R2(I)*C0C0+WV(I,2)*D0AV
      DO 79 M=1,ND
79    PC(I,M)=G2(I,M)+R2(I)*(02(I,M)+C00*D0(I,M))+CC0*C0(M)+CC1*C1(M)
80    CONTINUE
      DO 99 M=1,ND
      K=-ND+M
      DC 99 I=1,NP
      K=K+ND
      Z=T3*(PD(I,M)+V(I,1)*PD(2,M)+V(I,2)*PD(NPM1,M))
      IF(IA.EQ.1) PX(K)=Z
      IF(IA.EQ.2) PXA(K)=Z
      IF(IA.EQ.3) PDG(K)=Z
99    CONTINUE
      IF(NA.EQ.0) GC TO 200
      IF((IA.GT.1).AND.(NH.LT.3)) GO TO 200
100   CONTINUE
200   RETURN
      END


      SUBROUTINE RECIST(X,XV,NL,NLP,ND,NP,N)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      DIMENSION X(30,6),XV(10,3,6),G(3)
      COMMON/TIMEQ/TT,T(6),TACUM(7),T1(6),T2(6),T3(6),TP(6),TTP
      COMMON/CONS1/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
      COMMON/NCONS/PI,PI2,PIF,GRAV,TCON,VELCON,POWCON,PCON
      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,0)
      DO 13 M=1,ND
13    XV(I,M,LNP)=G(M)
20    W=W+TP(LNP)
      TACUM(1)=0.
```

```
      DC 3C LNP=1,NLP
      CALL MULT3(U,XV(1,1,LNP),X(1,LNP),ND,NP)
      T(LNP)=TP(LNP)
      TACUM(LNP+1)=TACUM(LNP)+T(LNP)
      T1(LNP)=1./T(LNP)
      T2(LNP)=T1(LNP)**2
      T3(LNP)=T2(LNP)*T1(LNP)
3C    TP(LNP)=(GRAV*T(LNP))**2
      NL=NLP
      RETURN
      END


      SUBROUTINE RESCAL(N,SCALE)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      COMMON/PXXPX/P(301),PHE(301),TAU(301),TAUS(301),T(301),DUMI(4819)
      SUM=0.
      T(1)=0.
      TAUS(1)=0.
      TAU(1)=0.
      FLTNM1=FLT(N-1)
      DO 1 I=2,N,2
      T(I)=FLT(I-1)/FLTNM1
      T(I+1)=FLT(I)/FLTNM1
      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)
1     TAUS(I+1)=SUM
      J=2
      DO 4 I=2,N
      S=FLT(I-1)*SUM/FLTNM1
      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/FLT(3*N-3)
      DO 5 I=1,N
5     TAUS(I)=TAUS(I)/SUM
```

```
      RETURN
      END


      SUBROUTINE RCCTR(PHE,NNN,PHEMAX,PV,T,WMU,C,ALPHAW,ST,NST,WMU1)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      DIMENSION PHE(301),ST(30)
      COMMON/NCONS/PI,PI2,PIF,GRAV,TCON,VELCON,POWCON,PCON
      DATA TAU,DELTA1 /1.618034,.001/
      WJV=PV/(VELCON*C)
      BCT=2.*WMU*ETA(C)*T*PCWCON/(ALPHAW*C**2)
      WKH=PHEMAX
      WK=WKH
      WJVMAX=0.
      PH=C.
      WKL=WKH/TAU
      DO 30 I=1,20
      CALL KROOT(PHE,NNN,BOT,WKL,WJVTST,WMU1,ST,NST,TEST)
      WJVMAX=AMX(WJVMAX,WJVTST)
      IF(TEST.LT.0.) WJVTST=1.E25
      PL=WJVTST
      IF((WJVTST.GE.WJV).OR.(I.EQ.20)) GO TO 40
      WK=WKH
      WKH=WKL
      PH=PL
      WKL=WK+(TAU+1.)*(WKH-WK)
30    CONTINUE
40    DO 5 I=1,25
      WK=.5*(WKH+WKL)
      CALL KRCOT(PHE,NNN,BOT,WK,WJVTST,WMU1,ST,NST,TEST)
      WJVMAX=AMX(WJVMAX,WJVTST)
      IF(TEST.LT.0.) WJVTST=1.E25
      IF(WJV-WJVTST) 2,3,3
2     WKL=WK
      PL=WJVTST
      GO TO 4
3     WKH=WK
      IF((WJVMAX.LT.WJV).AND.(PH.EQ.WJVTST)) GO TO 6
      PH=WJVTST
4     IF(ABS(1.-WJVTST/WJV).LT.DELTA1) GO TO 50
```

```
 5    CONTINUE
6     WMU1=1.-WJV/WJVMAX
      RETURN
50    WK=((PH-WJV)*WKL-(PL-WJV)*WKH)/(PH-PL)
      CALL KROOT(PHE,NNN,BOT,WK,WJVTST,WMU1,ST,NST,TEST)
      RETURN
      END


      SUBROUTINE SEARCH
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      LOGICAL CVRG1,CVRG2,CVRG3,CVRG4
      EXTERNAL CDERV1,CDERV2
      COMMON/CONST/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)C0000120
      COMMON/STATE/X(30,6),AL(3,2),P
      COMMON/OUT/XV(30,6),AV(30,6),WV(30,6)
      COMMON/PXXPX/PX(31,6),PXX(31,31,6),PXSAV(31,6),PDIAG(31,6)
      COMMON/COEFC/CC(9),COEF(9),NCOEF
      COMMON/INDEX/N1(6),N2(6),NCP(6),NBLN(6),KV(3,2),KA(3,2),NA(2),
     1NANB(2),LANB(2),MPM(6),MIM(6),LM(6,2),IM(6,2),KM(6,2)
      COMMON/CHANGE/DX1(30,6),DX2(30,6),DAL1(3,2),DAL2(3,2)
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NO,NH,NAP
      COMMON/OPTION/NP1,NP2,NB1,NB2,NV1,NV2,NPO
      COMMON/TIMEC/TT,T(6),TACUM(7),T1(6),T2(6),T3(6),TP(6),TTP
      COMMON/SKRCH2/XVA(30,2),VXA1(6,3,2),VX1(6,2),DUM2(102)
      COMMON/CVRG/CVRG1,CVRG2,CVRG3,CVRG4
      COMMON/ITER/IT,NBFIX,NHFIX
      COMMON/DER12/NDER12
      NDER12=0
      CVRG3=.TRUE.
      NCOEF=9
      RCO=0.
      CALL PDERIV(NC)
      PC=P
      NH=3
10    CONTINUE
      IF((NC.EQ.2).AND.((NV1.NE.1).AND.(NV2.NE.1))) NH=2
      IF((NC.EQ.2).AND.(NHFIX.EQ.1)) NH=2
      CALL PATCH(PX,PXX,PXSAV,PDIAG,2,0)
      CALL SQRCCT(PXX,NL,N1,N2,NCP,NAP,NSUCC)
```

```
      IF(NSUCC.EQ.1) GO TO 30
      IF((NC.EQ.2).AND.(NH.EQ.2)) STOP
      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)=PXSAV(I,LN)
      PXX(I,I,LN)=PDIAG(I,LN)
   20 CONTINUE
      IF(NC.EQ.2) NH=2
      NC=2
      GO TO 10
   30 CONTINUE
      DO 35 NB=1,2
      DO 35 K=1,3
      DAL1(K,NB)=0.
   35 DAL2(K,NB)=0.
      DO 40 I=1,NPD
      XVA(I,1)=0.
      XVA(I,2)=0.
      DO 40 LN=1,NL
   40 DX2(I,LN)=0.
      CALL BAKSUB(PXX,PX,PX,NL,N1,N2,NCP,NAP)
      CALL MODIFY(PX,DX1,DAL1,1)
      RC=1.
      CALL CDERV2(RC,P,PRO,0)
      IF(P.LT.(1.001*P0)) GO TO 260
      IF(NC.EQ.3) GO TO 220
      DO 180 LN=1,NL
      NAA=0
      IF((LN.EQ.1.AND.NA(1).GT.0).OR.(LN.EQ.NL.AND.NA(2).GT.0)) NAA=1
      NBL=NBLN(LN)
      CALL PXS(XV(1,LN),DX1(1,LN),XVA(1,NBL),AV(1,LN),WV(1,LN),
     CPX(1,LN),PXSAV(1,LN),PDIAG(1,LN),TP(LN),T3(LN),NAA)
  180 CONTINUE
      CALL PATCH(PX,PXX,PXSAV,PDIAG,1,1)
      CALL BAKSUB(PXX,PX,PX,NL,N1,N2,NCP,NAP)
      CALL MODIFY(PX,DX2,DAL2,2)
  220 CONTINUE
      IF(NC.EQ.3) CVRG3=.FALSE.
```

```
      DO 230 I=1,NCCEF
230   CCEF(I)=0.
      DO 255 LN=1,NL
      IF((LN.EQ.1.AND.NA(1).GT.0).OR.(LN.EQ.NL.AND.NA(2).GT.0))GO TO 255
      CALL COEFF
     C(XV(1,LN),DX1(1,LN),DX2(1,LN),AV(1,LN),WV(1,LN),CO,TP(LN))
      DO 250 I=1,NCCEF
250   CCEF(I)=T3(LN)*CO(I)+CCEF(I)
255   CONTINUE
      CALL CDERV1(RCO,PO,PRCO,1)
      RC=.1
      CALL CDERV1(RC,P,PRC,1)
      CALL LSERCH(RCO,PO,PRCO,RO,P,PRC,DDERV1)
      CALL CDERV2(RC,P,PRC,1)
      IF(P.LT.(1.001*PO)) GO TO 260
      CALL LSERCH(RCO,PO,PRCO,RC,P,PRC,CDERV2)
260   CONTINUE
      RCS=.5*RO**2
      DO 280 LN=1,NL
      DO 280 I=1,NPC
280   X(I,LN)=X(I,LN)+RO*DX1(I,LN)+RCS*DX2(I,LN)
      DO 290 NB=1,2
      DO 290 K=1,3
290   AL(K,NB)=AL(K,NB)+RO*DAL1(K,NB)+RCS*DAL2(K,NB)
      IF((NHFIX.EQ.1).AND.(NBFIX.EQ.0)) NHFIX=0
      IF((RC.LT..1).AND.(NBFIX.EQ.0).AND.(NH.EQ.3)) NHFIX=1
      RETURN
      END


      SUBROUTINE SCRCUT(A,NL,N1,N2,NCP,NAP,NSUCC)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      DIMENSION A(31,31,6),N1(6),N2(6),NCP(6)
      NSUCC=0
      DO 45 LN=1,NL
      NCP3=N1(LN)
      NCP1=NCP3
      NCP2=N2(LN)
      NCP4=NCP2+NAP
      IF(LN.EQ.1) GO TO 9
```

```
      NCPL=NCP(LN)
      NCP1=NCP1+NCPL
      DO 8 J=1,NCPL
      JM1=J-1
      JLN=NCP3-1+J
      JLNM1=N2(LN-1)-NCPL+J
      IF(NAP.EQ.0) GO TO 4
      IP=NCP2+1
      IPM1=N2(LN-1)+1
      A(IP,JLN,LN)=A(IPM1,JLNM1,LN-1)
4     DO 5 I=J,NCPL
      ILN=NCP3-1+I
      ILNM1=N2(LN-1)-NCPL+I
5     A(ILN,JLN,LN)=A(ILNM1,JLNM1,LN-1)
      DO 7 I=NCP1,NCP2
      Z=A(I,JLN,LN)
      IF(J.EQ.1) GO TO 7
      DO 6 K=1,JM1
      KLN=NCP3-1+K
6     Z=Z-A(I,KLN,LN)*A(JLN,KLN,LN)
7     A(I,JLN,LN)=Z*A(JLN,JLN,LN)
8     CONTINUE
9     DO 40 J=NCP1,NCP2
      JP1=J+1
      JM1=J-1
      Z=A(J,J,LN)
      IF(J.EQ.NCP3) GO TO 20
      DO 10 K=NCP3,JM1
10    Z=Z-A(J,K,LN)**2
20    CONTINUE
      IF(Z.LE.0.) RETURN
      A(J,J,LN)=1./SQT(Z)
      IF(J.EQ.NCP4) GO TO 45
      DO 35 I=JP1,NCP4
      Z=A(I,J,LN)
      IF(J.EQ.NCP3) GO TO 35
      DO 30 K=NCP3,JM1
30    Z=Z-A(I,K,LN)*A(J,K,LN)
35    A(I,J,LN)=Z*A(J,J,LN)
40    CONTINUE
```

```
45    CONTINUE
      IF(NAP.EQ.0) GO TO 78
      JP=N2(NL)+1
      Z=A(JP,JP,NL)
      DO 50 LN=1,NL
      NCP3=NI(LN)
      IP=N2(LN)+1
      IF(LN.LT.NL) NCP2=N2(LN)-NCP(LN+1)
      IF(LN.EQ.NL) NCP2=N2(LN)
      DO 50 K=NCP3,NCP2
50    Z=Z-A(IP,K,LN)**2
      IF(Z.LE.0.) RETURN
      A(JP,JP,NL)=1./SQT(Z)
78    NSUCC=1
      RETURN
      END


      SUBROUTINE START(RNN,NLP,NT)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      REAL RNN
      COMMON/BVL/D1,D2,H1,H2,V1(12),V2(12)
      COMMON/OUT/XV(10,3,6),AV(30,6),WV(30,6)
      COMMON/STATE/X(10,3,6),AL(3,2),P
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NO,NH,NAP
      COMMON/TIMEQ/TT,T(6),TACUM(7),T1(6),T2(6),T3(6),TP(6),TTP
      COMMON/CONSI/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
      COMMON/NCONS/PI,PI2,PIF,GRAV,TCON,VELCON,POWCON,PCON
      COMMON/MAXNLG/MAXNL
      FND=FLT(ND-2)
      IF(NT.EQ.1) GO TO 4
      Q1=ATN2(V1(2),V1(1))
      Q2=ATN2(V2(2),V2(1))
      Q=Q2-Q1+PI2
      Q=AMOD(Q,PI2)
      IF(RNN.GE.0.) NR=RNN-(Q/PI2)+1.
      IF(RNN.LT.0.) NR=RNN-(Q/PI2)-1.
      Q=Q+PI2*FLT(NR)
      RO1=V1(1)**2+V1(2)**2
      RO2=V2(1)**2+V2(2)**2
```

```
      R1=SQT(RO1+FND*V1(3)**2)
      R2=SQT(RO2+FND*V2(3)**2)
      R=R2-R1
      O1=ATN(V1(3)/SQT(RO1))
      C2=ATN(V2(3)/SQT(RO2))
      C=O2-C1
      DR=AMX(R1,R2)/AMN(R1,R2)
      RF=1.
      IF(DR.GT.2.) RF=4./3.
      IF(DR.GT.6.) RF=2.
      IF(AMN(R1,R2).LT..3) RF=3.
      NLP=1.+RF*AB(Q)/PI
      NLP=MINO(NLP,MAXNL)
      IF(NLP.GT.3) NLP=NLP-1
      IF(NLP.GT.6) NLP=NLP-1
      NL=NLP
    4 TACUM(1)=C.
      DO 5 LN=1,NL
      IF(NT.EQ.0) T(LN)=TT/FLT(NL)
      IF(NT.EQ.1) T(LN)=T(LN)*TT/TTP
      TACUM(LN+1)=TACUM(LN)+T(LN)
      T1(LN)=1./T(LN)
      T2(LN)=T1(LN)**2
      T3(LN)=T2(LN)*T1(LN)
    5 TP(LN)=(GRAV*T(LN))**2
      IF(NT.EQ.1) RETURN
      R1T=(V1(1)*V1(4)+V1(2)*V1(5)+FND*V1(3)*V1(6))/R1
      R2T=(V2(1)*V2(4)+V2(2)*V2(5)+FND*V2(3)*V2(6))/R2
      C1T=(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
      C2T=(V2(1)*V2(5)-V2(2)*V2(4))/RO2
      IF(R.EQ.0.) R=.0000001
      IF(Q.EQ.0.) Q=.0000001
      IF(C.EQ.0.) C=.0000001
      R1S=TT*R1T/R
      R2S=TT*R2T/R
      C1S=TT*Q1T/Q
      C2S=TT*Q2T/Q
      C1S=TT*O1T/C
```

```
      C2S=TT*Q2T/Q
      R1S=0.
      R2S=C.
      DO 10 LN=1,NL
      DC 8 I=1,NP
      S=(TACUM(LN)+CHEB(I)*T(LN))/TT
      D=D1+STARTF(Q1S,Q2S,S)*Q
      IF(NC.EQ.2) C=0.
      E=Q1+STARTF(Q1S,Q2S,S)*Q
      F=R1+STARTF(R1S,R2S,S)*R
      XV(I,1,LN)=F*CN(E)*CN(D)
      XV(I,2,LN)=F*SN(E)*CN(D)
    8 XV(I,3,LN)=F*SN(D)
   10 CONTINUE
      DC 79 LN=1,NL
   79 CALL MULT3(U,XV(1,1,LN),X(1,1,LN),ND,NP)
      SR=R/AB(R)
      AL(1,1)=D1/TCCN
      AL(1,2)=D2/TCCN
      AL(2,1)=ATN2(SR*V1(5),SR*V1(4))
      AL(2,2)=ATN2(SR*V2(5),SR*V2(4))
      AL(3,1)=ATN(SR*V1(6)/SQT(V1(4)**2+V1(5)**2))
      AL(3,2)=ATN(SR*V2(6)/SQT(V2(4)**2+V2(5)**2))
      IF(ND.EQ.2) AL(3,1)=0.
      IF(ND.EQ.2) AL(3,2)=0.
      RETURN
      END


      FUNCTION STARTF(DZ,D1,S)
      IMPLICIT REAL*8(A-H,O-Z)                                  360 IN
      IF(DZ+D1.GT.2.) GC TO 1
      STARTF= S*(DZ+S*(3.-2.*DZ-D1+S*(DZ+D1-2.)))
      RETURN
    1 B=(2./(DZ+D1))/(1.+SQT(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 TSHIFT(X,XV,NL,NLP,ND,NP,N)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      COMMON/TIMEQ/TT,T(6),TACUM(7),T1(6),T2(6),T3(6),TP(6),TTP
      COMMON/CONS1/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
      COMMON/PXXPX/P(301),PHE(301),TAU(301),DUM1(5421)
      COMMON/MAXNLG/MAXNL
      DIMENSION X(30,6),XV(10,3,6),G(3)
      DATA NPPL,EXP1,EXP2 /20,-.75,-.50/
      IF(NL.EQ.1.AND.NLP.EQ.1) RETURN
      NLP=MINO(NLP,MAXNL)
      DO 50 LN=1,NL
      CALL MULT3(V,X(1,LN),XV(1,1,LN),ND,NP)
   50 CALL MULT4(A,XV(1,1,LN),X(1,LN),ND,NP,N,1)
      NNN=NPPL*NLP+1
      FLTNM1=FLT(NNN-1)
      DO 3 I=1,NNN
      TS=TT*FLT(I-1)/FLTNM1
      CALL MODLEG(T,NL,TS,LN)
      CALL POLEVL(X(1,LN),G,TS,ND,NP,N,0)
      Z=0.
      DO 2 M=1,ND
    2 Z=Z+G(M)**2
      IF(Z.LE.1) EXP=EXP1
      IF(Z.GT.1) EXP=EXP2
    3 P(I)=Z**EXP
      CALL RESCAL(NNN,SCALE)
      DO 5 LN=1,NLP
      NPPLN1=NPPL*LN+1
    5 TP(LN)=TT*TAU(NPPLN1)
      TP(NLP)=TT
      IF(NLP.LT.2) GO TO 900
      DO 6 I=2,NLP
      NLP2I=NLP+2-I
      NLP1I=NLP+1-I
    6 TP(NLP2I)=TP(NLP2I)-TP(NLP1I)
```

```
9C0   CALL REDIST(X,XV,NL,NLP,NC,NP,N)
95C   RETURN
      END


      SUBRCUTINE VCAL(NP,NB,NV,H,T,AL,V,VX,VXA,VXAA)
      IMPLICIT REAL*8(A-H,O-Z)                                        360 IN
      CCMMON/NCCNS/PI,PI2,PIF,GRAV,TCON,VELCON,POWCON,PCON
      DIMENSION AL(3),VX(6),VXA(6,3),VXAA(6,3,3),V(12)
      IF(NB.NE.2) CALL EPHEM(TCON*AL(1),NP,V)
      HT=H*T
      AL(2)=AMD(AL(2),PI2)
      CAL2=CN(AL(2))
      SAL2=SN(AL(2))
      CAL3=CN(AL(3))
      SAL3=SN(AL(3))
      VX(4)=HT*CAL2*CAL3
      VX(5)=HT*SAL2*CAL3
      VX(6)=HT*SAL3
      IF(NV.NE.1) GC TO 15
      VXA(4,2)=-VX(5)
      VXA(5,2)=VX(4)
      VXA(4,3)=-HT*CAL2*SAL3
      VXA(5,3)=-HT*SAL2*SAL3
      VXA(6,3)=HT*CAL3
      VXAA(4,2,3)=-VXA(5,3)
      VXAA(5,2,3)=VXA(4,3)
      VXAA(4,3,2)=VXAA(4,2,3)
      VXAA(5,3,2)=VXAA(5,2,3)
15    DC 20 M=1,3
      VX(M)=V(M)
      VXAA(M+3,3,3)=-VX(M+3)
      IF(M.LT.3) VXAA(M+3,2,2)=-VX(M+3)
      VX(M+3)=VX(M+3)+T*V(M+3)
      VXA(M,1)=V(M+3)
      VXA(M+3,1)=T*V(M+6)
      VXAA(M,1,1)=V(M+6)
      VXAA(M+3,1,1)=T*V(M+9)
20    CONTINUE
      RETURN
```

```
      END


      SUBROUTINE VTMODE
     C(NC,RN,NPL1,NPL2,NBD1,NBD2,NVL1,NVL2,DA1,DA2,HV1,HV2,NPOW,NT)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      REAL PRAT1,PRAT2,RRR1,RRR2,VEL1,VEL2,DD1,DD2
      LOGICAL CVRG1,CVRG2,CVRG3,CVRG4
      REAL RN,DA1,DA2,HV1,HV2,RNN,HH1,HH2,BDY,PV,PC,BT,PMF
      COMMON/BDYP/BDY(3,4,2),PV,PC,BT,PMF
      COMMON/STATE/X(10,3,6),AL(3,2),P
      COMMON/BVL/D1,D2,H1,H2,V1(12),V2(12)
      COMMON/INDEX/N1(6),N2(6),NCP(6),NBLN(6),KV(3,2),KA(3,2),NA(2),
     CNMNB(2),LNNB(2),MMM(6),MIM(6),LM(6,2),IM(6,2),KM(6,2)
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NO,NH,NAP
      COMMON/OPTION/NP1,NP2,NB1,NB2,NV1,NV2,NPO
      COMMON/OUT/XV(30,6),AV(10,3,6),WV(30,6)
      COMMON/CONS1/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)C0000120
      COMMON/TIMEQ/TT,T(6),TACUM(7),T1(6),T2(6),T3(6),TP(6),TTP
      COMMON/NCONS/PI,PI2,PIF,GRAV,TCON,VELCON,POWCON,PCON
      COMMON/INTOUT/RNN,NCC,NTT,ITT,HH1,HH2
      COMMON/PANDR/VEL1,PRAT1,PRAT2,RRR1,RRR2,VEL2,DD1,DD2
      COMMON/CVRG/CVRG1,CVRG2,CVRG3,CVRG4
      COMMON/COUNT/NCOUNT,NLP
      COMMON/ITER/IT,NBFIX,NHFIX
      COMMON/MAXNLG/MAXNL
      DIMENSION G1(3),GNL(3)
      DATA DELTA1,DELTA2,DELTAN,DELTAP /.001,.0001,.01,.05/
      DATA NCON,NIT /0,200/
      NCOUNT=NCOUNT+1
      CVRG4=.FALSE.
      NF=0
      IF(NT.EQ.1) NF=1
      ITT=0
      NPC=NPOW
      NCC=NC
      RNN=RN
      NTT=NT
      D1=DA1
      D2=DA2
```

```
      NP1=NPL1
      NP2=NPL2
      HH1=HV1
      HH2=HV2
      H1=VELCON*HV1*FLT(2-NVL1)
      H2=VELCON*HV2*FLT(2-NVL2)
      NC=2
      IF(NT.EQ.1) NC=3
      TTP=TT
      TT=(C2-D1)/TCON
      CALL EPHEM(D1,NP1,V1)
      CALL EPHEM(D2,NP2,V2)
      PS=0.
      NP=10
      NPM1=NP-1
      N=NP/2
      IF(NCON.EQ.0) CALL CONST(NP,N)
      NCON=1
      ND=NC
      IF(NC.EQ.3.AND.NT.EQ.0)      ND=2
      IF(NC.EQ.1) NC=3
      CALL START(RAA,NLP,NT)
   15 CONTINUE
      NPD=NP*NC
      ND2=2*ND
      NPD1=NPD-NC
      NPC2=NPD-ND2
      NBFIX=1
      NHFIX=0
      DELTA=DELTA1
      IF((NBD1.LT.2).OR.(NBC2.LT.2)) DELTA=DELTA2
      DO 99 IT=1,NIT
      ITT=ITT+1
      IF(NL.LT.3) NC=3
      IF((NIT.EQ.1).AND.(IT.GT.1).AND.(TEST.LT.1.)) NBFIX=0
      IF((NIT.EQ.1).AND.(TEST.GT.1.)) NBFIX=1
      IF((NIT.EQ.0).AND.(NL.LE.3)) NBFIX=0
      CALL INCCAL(NBC1,NBC2,NVL1,NVL2,NBFIX)
      CALL SEARCH
      IF(NF.EQ.0) CVRG3=.TRUE.
```

```
      TEST=ABS((PS-P)/P)
      PS=P
      CVRG4=(TEST.LT.DELTAN)
      CVRG1=(IT.GT.1).AND.(TEST.LT.DELTA).AND.(NBFIX.EQ.0)
      IF(NF.EQ.C) CVRG1=(IT.GT.1).AND.(TEST.LT.DELTAP).AND.(NBFIX.EQ.0)
      CVRG2=(NO.EQ.3).OR.(NF.EQ.0)
      IF(IT.EC.NIT) GC TC 940
      IF(CVRG1.AND.CVRG2.AND.CVRG3) GO TO 100
      IF(NC.LT.3) NTT=0
      NBFIX=1-MOD(IT,2)
      IF(NC.EQ.3) NBFIX=0
      IF((IT.GT.1).AND.(CVRG4.OR.(NL.LE.3))) NO=3
   99 CONTINUE
  100 CONTINUE
      CALL PDERIV(0)
      CALL        AVTEST(AV,T2,NLP,NL,ND,NP)
      IF(((NF.EC.1).AND.(NL.EQ.NLP)).OR.(NF.EQ.2)) GO TO 925
      IF((NL.EQ.MAXNL).AND.(NF.EQ.1)) NF=2
      IF(NF.EQ.C) NF=1
      CALL TSHIFT(X,XV,NL,NLP,ND,NP,N)
      GC TC 15
  925 IF(ND.EQ.NC.CR.ND.EQ.3) GC TO 950
      ND=NC
      OC 930 LN=1,NL
      DO 930 I=1,NP
  930 X(I,3,LN)=0.
      GC TC 15
  94C CONTINUE
      WRITE(6,1C09)
  950 CONTINUE
      Z1=C.
      Z2=0.
      R1=C.
      R2=0.
      DO 975 M=1,ND
      BDY(M,1,1)=X(1,M,1)
      BDY(M,2,1)=X(2,M,1)*T1(1)
      BDY(M,1,2)=X(NP,M,NL)
      BDY(M,2,2)=-X(NPM1,M,NL)*T1(NL)
      BDY(M,3,1)=T2(1)*AV(1,M,1)
```

```
      BDY(M,3,2)=T2(NL)*AV(NP,M,NL)
      R1=R1+X(1,M,1)**2
      R2=R2+X(NP,M,NL)**2
      G1(M)=X(2,M,1)/T(1)-V1(M+3)
      GNL(M)=-X(NPM1,M,NL)/T(NL)-V2(M+3)
      Z1=Z1+G1(M)**2
  975 Z2=Z2+GNL(M)**2
      R1=SQT(Z1)/VELCON
      R2=SQT(Z2)/VELCON
      PRAT1=WV(1,1)
      PRAT2=WV(NP,NL)
      RRR1=SQT(R1)
      RRR2=SQT(R2)
      VEL1=H1
      VEL2=H2
      CC1=TCON*AL(1,1)
      CC2=TCON*AL(1,2)
      IF(NV1.EQ.2) AL(2,1)=ATN2(AV(1,2,1),AV(1,1,1))
      IF(NV2.EQ.2) AL(2,2)=ATN2(AV(NP,2,NL),AV(NP,1,NL))
      IF((NV1.EQ.2).AND.(ND.EQ.3))
     CAL(3,1)=ATN(AV(1,3,1)/SQT(AV(1,2,1)**2+AV(1,1,1)**2))
      IF((NV2.EQ.2).AND.(ND.EQ.3))
     CAL(3,2)=ATN(AV(NP,3,NL)/SQT(AV(NP,2,NL)**2+AV(NP,1,NL)**2))
      IF(NV1.EQ.0) AL(2,1)=ATN2(G1(2),G1(1))
      IF(NV2.EQ.0) AL(2,2)=ATN2(GNL(2),GNL(1))
      IF((NV1.EQ.0).AND.(ND.EQ.3))
     CAL(3,1)=ATN(G1(3)/SQT(G1(1)**2+G1(2)**2))
      IF((NV2.EQ.0).AND.(ND.EQ.3))
     CAL(3,2)=ATN(GNL(3)/SQT(GNL(1)**2+GNL(2)**2))
      DO 985 LN=1,NL
      CALL MULT3(V,X(1,1,LN),XV(1,LN),ND,NP)
      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)
  985 CONTINUE
      R1=0.
      R2=1.
```

```
Z1=SQT(Z1)/VELCON
Z2=SQT(Z2)/VELCON
R1=SQT(R1)
R2=SQT(R2)
H1=Z1
H2=Z2
VEL1=Z1
VEL2=Z2
RRR1=R1
RRR2=R2
PRAT1=1./WV(1,1)**2
PRAT2=1./WV(NP,NL)**2
```

```
      CALL POLEVL(AV(1,1,1),G1,R1,ND,NP,N,1)
      CALL POLEVL(AV(1,1,NL),GNL,R2,ND,NP,N,1)
      DO 990 M=1,ND
      BDY(M,4,1)=G1(M)*T1(1)
990   BDY(M,4,2)=GNL(M)*T1(NL)
      PV=P
      NTT=NT
      RETURN
1000  FORMAT(//4X39HFOLLOWING TRAJECTORY FAILED TO CONVERGE)
      END


      SUBROUTINE VTOUT
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      REAL TILT,PLOSS,ANGL,AYOU,ELEM,RNN,HH1,HH2
      COMMON/COUNT/NCOUNT,NLP
      COMMON/BVL/D1,D2,H1,H2,V1(12),V2(12)
      COMMON/TIMEQ/TT,T(6),TACUM(7),T1(6),T2(6),T3(6),TP(6),TTP
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NO,NH,NAP
      COMMON/OPTION/NP1,NP2,NB1,NB2,NV1,NV2,NPO
      COMMON/INTOUT/RNN,NCC,NTT,ITT,HH1,HH2
      COMMON/ARRAY/TILT,PLOSS
      COMMON/ESTUFF/ANGL,AYOU
      COMMON/ELEMNT/ELEM(7)
      COMMON/STATE/X(10,3,6),AL(3,2),P
      COMMON/OUT/XV(30,6),AV(30,6),WV(30,6)
      COMMON/NCONS/PI,PI2,PIF,GRAV,TCON,VELCON,POWCON,PCON
      COMMON/SKRCH2/B(15),G(3),GT(3),CONESC(4,2),DUM1(181)
      AL11=TCON*AL(1,1)
      AL12=TCON*AL(1,2)
      AL21=PIF*AL(2,1)
      AL22=PIF*AL(2,2)
      AL31=PIF*AL(3,1)
      AL32=PIF*AL(3,2)
      PS=PCON*P
      WRITE(6,8000)
      WRITE(6,90) NCOUNT
      WRITE(6,92) NCC,RNN,NP1,NP2,NB1,NB2,NV1,NV2,D1,D2,HH1,HH2,NPO,NTT
      WRITE(6,94) TILT,PLOSS,ANGL,AYOU
      WRITE(6,95) ELEM
```

```
      WRITE(6,96) NL,ITT
      WRITE(6,98) AL11,H1,AL21,AL31,AL12,H2,AL22,AL32,PS
      WRITE(6,109)
      G(3)=0.
      GT(3)=0.
      DO 20 I=1,26
      S=FLT((I-1)*NL+25)/25.
      LN=S
      IF(LN.GT.NL) LN=NL
      S=S-FLT(LN)
      Y=TACUM(LN)+S*T(LN)
      B(1)=TCON*Y
      CALL POLEVL(WV(1,LN),Z,S,1,NP,N,0)
      B(12)=1./Z**2
      CALL POLEVL(XV(1,LN),G,S,ND,NP,N,0)
      CALL POLEVL(XV(1,LN),GT,S,ND,NP,N,1)
      W=0.
      DO 15 M=1,3
      B(M+1)=G(M)
      B(M+12)=GT(M)/T(LN)
15    W=W+G(M)**2
      B(5)=SQT(W)
      B(11)=AMAP(Y,CONESC,0,B(10),B(8),B(9),PHE1,PHE2)
      Z=PIF*ATN2(G(2),G(1))
      IF(I.EQ.1) GO TO 17
      DO 16 K=1,5
      IF((Z-ZSAV).GT.180.) Z=Z-360.
16    IF((Z-ZSAV).LT.-180.) Z=Z+360.
17    B(6)=Z
      ZSAV=Z
      B(7)=PIF*AIN(G(3)/SQT(G(1)**2+G(2)**2))
20    WRITE(6,100)(B(J),J=1,15)
      RETURN
90    FORMAT(////50X4HCASE,I5)
92    FORMAT(/1X15HCALL LIST INPUT,15X,
     163HNC RN  NP1 NP2 NB1 NB2 NV1 NV2   D1      D2      H1     H2  NPO NT
     2/29X,I3,F5.1,I3,5I4,F8.1,F7.1,2F6.1,I3,I4)
94    FORMAT(/1X18HCOMMON BLOCK INPUT,12X,
     126HTILT   PLOSS   ANGL   AYOU,/30X,F5.1,F8.2,F7.1,F7.1)
95    FORMAT(/1X18HCOMMON BLOCK INPUT,12X,2X2HE1,8X2HE2,8X2HE3,8X2HE4,
```

```
     C8X2HE5,8X2HE6,8X2HE7,/30X,F10.5,5F10.7,F7.1)
96    FORMAT(/1X19HINTERNAL PARAMETERS,11X,6HNL NIT,/31X,I2,I4)
98    FORMAT(/1X6HOUTPUT,24X,
     124HC1      H1      AL1     BE1,8X,30HD2      H2      AL2     BE2     JV,
     2/28X,F7.1,F6.1,F8.1,F7.1,4X,F7.1,F6.1,F8.1,F7.1,F9.3)
100   FORMAT(F9.2,4F8.3,6F8.2,F8.3,8X,3F8.3)
109   FORMAT(//3X4HTIME6X1HX7X1HY7X1HZ7X1HR6X5HTHETA4X3HPHI5X4HCONE
     C3X5HCLOCK3X4HXZEE4X6HMAG. A3X4HP/P012X2HXD6X2HYD6X2HZD//)
8000  FORMAT(1H1,//)
      END


      FUNCTION WMUL(WMU,C,ALPHAW,WMU1)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      REAL TANKS
      COMMON/INERTS/TANKS
      WMUL=WMU1*(1.+TANKS)-WMU-TANKS
      RETURN
      END


      SUBROUTINE WYDER(TO,TL,TP,XV,BAXV,AV,WV,O,Q,G,GD,H,P,NORDER)
      IMPLICIT REAL*8(A-H,O-Z)                                          360 IN
      COMMON/OPTION/NP1,NP2,NB1,NB2,NV1,NV2,NPO
      COMMON/PARAM/NL,ND,NP,N,NPD,NPM1,ND2,NPD2,NPD1,NO,NH,NAP
      COMMON/CONST/A(5,5,2),AA(5,5,2),BA(5,5,2),U(10,2),V(10,2),CHEB(10)
      DIMENSION XV(10,3),BAXV(10,3),AV(10,3),WV(10,3),RV(10),O(10,3),
     CC(10,3),G(10,3,3),GD(10,3,3),H(10,3,3)
      DIMENSION RXV(3),RC(3),RAV(3),RHV(3),QQ(3)
      DO 39 I=1,NP
      TI=TO+CHEB(I)*TL
      Z=0.
      DO  8 M=1,ND
8     Z=Z+XV(I,M)**2
      RV(I)=SQT(Z)
      R1=1./RV(I)
      R3=TP*R1**3
      CALL POWER(RV(I),TI,NPD,NORDER,PO,P1,P2)
      WV(I,1)=1./SQT(PO)
      IF(NORDER.GT.0) WV(I,2)=-P1*RV(I)/(2.*PO)
```

```
      IF(NORDER.GT.1) WV(I,3)=-P2*RV(I)**2/(2.*P0)
     C+WV(I,2)*(3.*WV(I,2)-1.)
      DO 39 M=1,ND
      AV(I,M)=BAXV(I,M)+R3*XV(I,M)
39    C(I,M)=WV(I,1)*AV(I,M)
      CALL MULT4(AA,C,C,ND,NP,N,1)
      CALL MULT2(Q,C,P,ND,NP)
      IF(NORDER.LT.1) GO TO 100
      DO 80 I=1,NP
      R1=1./RV(I)
      R3=TP*R1**3
      RCXV=0.
      RCAV=0.
      RCHV=0.
      DO 49 M=1,ND
      C(I,M)=WV(I,1)*C(I,M)
      RXV(M)=R1*XV(I,M)
      RC(M)=R1*C(I,M)
      RAV(M)=R1*AV(I,M)
      RHV(M)=WV(I,2)*RAV(M)-3.*R3*RXV(M)
      RCXV=RQXV+RC(M)*RXV(M)
      RCAV=RCAV+RC(M)*RAV(M)
49    RCHV=RQHV+Q(I,M)*RHV(M)
      IF(NORDER.LT.3) GO TO 60
      CRC=R3*(WV(I,2)-3.)
      CRXV=3.*R3*RQXV*(2.5-WV(I,2))+.5*RQAV*WV(I,3)
      GCC=RQAV*WV(I,2)-3.*R3*RQXV
60    DO 79 M1=1,ND
      C(I,M1)=R3*C(I,M1)+RQHV*RXV(M1)
      IF(NORDER.LT.2) GO TO 79
      DO 75 M2=1,ND
      G(I,M1,M2)=RC(M1)*WV(I,2)*RXV(M2)
75    H(I,M1,M2)=RXV(M2)*RHV(M1)
      H(I,M1,M1)=H(I,M1,M1)+R3
      IF(NORDER.LT.3) GO TO 79
      CC(M1)=CRC*RC(M1)+CRXV*RXV(M1)
      DO 77 M2=1,M1
77    GC(I,M1,M2)=RXV(M1)*QC(M2)+RXV(M2)*QC(M1)
      GC(I,M1,M1)=GC(I,M1,M1)+GCC
79    CONTINUE
```

```
80    CONTINUE
100   CONTINUE
      RETURN
      END


      FUNCTION AB(X)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      AB=DABS(X)                                                  360 IN
      AB=ABS(X)                                                   360 OUT
      RETURN
      END
      FUNCTION ALN(X)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      ALN=DLOG(X)                                                 360 IN
      ALN=ALOG(X)                                                 360 OUT
      RETURN
      END
      FUNCTION AMD(X,Y)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      AMD=DMOD(X,Y)                                               360 IN
      AMD=AMOD(X,Y)                                               360 OUT
      RETURN
      END
      FUNCTION AMN(X,Y)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      AMN=DMIN1(X,Y)                                              360 IN
      AMN=AMIN1(X,Y)                                              360 OUT
      RETURN
      END
      FUNCTION AMX(X,Y)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      AMX=DMAX1(X,Y)                                              360 IN
      AMX=AMAX1(X,Y)                                              360 OUT
      RETURN
      END
      FUNCTION ATN(X)
      IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
      ATN=DATAN(X)                                                360 IN
      ATN=ATAN(X)                                                 360 OUT
```

```
RETURN
END
FUNCTION ATN2(X,Y)
IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
ATN2=DATAN2(X,Y)                                            360 IN
ATN2=ATAN2(X,Y)                                             360 OUT
RETURN
END
FUNCTION CN(X)
IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
CN=DCOS(X)                                                  360 IN
CN=CCS(X)                                                   360 OUT
RETURN
END
FUNCTION EX(X)
IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
EX=DEXP(X)                                                  360 IN
EX=EXP(X)                                                   360 OUT
RETURN
END
FUNCTION FLT(N)
IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
FLT=DFLOAT(N)                                               360 IN
FLT=FLOAT(N)                                                360 OUT
RETURN
END
FUNCTION SN(X)
IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
SN=DSIN(X)                                                  360 IN
SN=SIN(X)                                                   360 OUT
RETURN
END
FUNCTION SQT(X)
IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
SQT=DSQRT(DABS(X))                                          360 IN
SQT=SQRT(ABS(X))                                            360 OUT
RETURN
END
BLOCK DATA
IMPLICIT REAL*8(A-H,O-Z)                                    360 IN
```

```
      REAL TILT,PLOSS,ANGL,AYOU,E,BB,DD,TANKS
      COMMON/INERTS/TANKS
      COMMON/THRUST/BB,CC
      COMMON/ELEMNT/E(7)
      COMMON/COUNT/NCOUNT,NLP
      COMMON/MAXNLG/MAXNL
      COMMON/NCONS/PI,PI2,PIF,GRAV,TCON,VELCON,POWCON,PCON
      COMMON/ARRAY/TILT,PLOSS
      COMMON/BDYQ/VX(6,2),VXA(18,2),VXAA(54,2)
      COMMON/ESTUFF/ANGL,AYOU
      DATA NCOUNT,NLP /0,1/
      DATA MAXNL /6/
      DATA PI,PI2,PIF,GRAV,TCON,VELCON,POWCON,PCON
     C/3.1415926535898,6.2831853071796,57.295779513082,6.2831853071796,
     C365.25,.2104,31557.5,.7121/
      DATA VX,VXA,VXAA /156*0./
      DATA TILT,PLOSS /90.,0./
      DATA ANGL,AYOU /0.,1./
      DATA E /7*0./
      DATA TANKS /0./
      DATA BB,CC /1.,0./
      END
```

```
OVERLAY A
INSERT CCNCP,CCPTR,CTMODE,CTOUT,ETA,KOPTR,KROOT,MCOPTR,MOPTR
INSERT PMAP,RCCTR,VTCUT,WPUL
OVERLAY A
INSERT ALIGN,AVTEST,BAKSUB,COEFF,CONST,DDERV1,DDERV2,DERIV,EPHEM
INSERT INDCAL,LSERCH,MODIFY,MULT2,MULT3,MULT4,MULT7,MULT8,MULT9
INSERT PATCH,PDERIV,PXS,REDIST,SEARCH,SQROOT,START,STARTF,TSHIFT
INSERT VCAL,WYDER
ENTRY MAIN
```