X-553-67-491

NASA TM X- 63085

COWELL ORBIT GENERATOR ERROR AND TIME ANALYSIS PROGRAM

G. A. CIGARSKI

C. E. VELEZ

AUGUST 1967

GODDARD SPACE FLIGHT CENTER

GREENBELT, MARYLAND

X-553-67-491

COWELL ORBIT GENERATOR ERROR AND TIME ANALYSIS PROGRAM

G. A. Cigarski
Wolf Research and Development Corp.

Under the Technical Direction
of
C. E. Velez

August 1967

Program Systems Branch
Mission & Trajectory Analysis Division
GODDARD SPACE FLIGHT CENTER
Greenbelt, Maryland

COWELL ORBIT GENERATOR ERROR AND TIME ANALYSIS PROGRAM

G. A. Cigarski
Wolf Research and Development Corp.

Under the Technical Direction
of
C. E. Velez

ABSTRACT

A program designed to numerically integrate the equations of motion of an artificial earth satellite is described. The method used is the Cowell predictor-corrector process with a local error control which is based on a variation of order and/or stepsize during the integration, as described in Velez, C. E., "Local Error Control and its Effects on the Optimization of Orbital Integration," NASA X-553-67-366. The effects of such controls on the integration are displayed in the form of a resume which includes computer timing data, a stepsize distribution analysis, and a propagated numerical error estimate. In addition, an option to numerically integrate in multirevolution steps is described.

CONTENTS

CONTENTS (Continued)

I. DESCRIPTION

In the computation of orbits of artificial earth satellites by numerical integration, advances in the theory of perturbations have led to the use of more complex force models and have resulted in the need for more efficient integration techniques. On the other hand, the availability of highly accurate tracking systems has increased the need for correspondingly accurate numerical processes.

The program to be described is the result of an investigation into numerical integration methods as applied to the calculation of orbits. In particular, the program is designed to examine how controlling the local error affects the efficiency and accuracy of the process.

The numerical integration method considered is the multi-step technique using the well-known "summed" form of the Adams-Cowell formulas.* During this process the local error is controlled (restricted) by variation of the parameters p and h, the order and stepsize associated with these formulas. Variation of the stepsize and/or order is governed by specifying a set of error tolerances or upper and lower bounds on the order. (See Section IV.)

The effects of such control on the integration are displayed in the form of a resume which includes timing estimates, a stepsize distribution analysis and a propagated numerical error estimate.** As an application, these results could be used in the calibration of other numerical integration techniques which require the specification of p or h.

An additional feature has been incorporated into the program which allows one to use a multirevolution process along with the integration, when many revolutions of orbit are required. This process is described in Section V.

---

*Equivalent Methods: (1) Adams-Moulton; Störmer-Cowell. (2) Gauss-Jackson (central differences).
**For the case of 2-body (elliptic) motion.

II. FORMULAS FOR INTEGRATION MODEL

The integration model employed is a multi-step process using the following form of the Adams-Cowell formulas[1]:

$$\text{(Predictor)} \quad \overline{X}_{n+1} = h^2\left[{}^{II}\overline{S}_n + \frac{1}{12}\ddot{\overline{X}}_n + \frac{1}{12}\nabla\ddot{\overline{X}}_n + \frac{19}{240}\nabla^2\ddot{\overline{X}}_n + \ldots + \sigma_k\nabla^{k-2}\ddot{\overline{X}}_n\right] \quad (1)$$

$$\text{(Corrector)} \quad \overline{X}_{n+1} = h^2\left[{}^{II}\overline{S}_n + \frac{1}{12}\ddot{\overline{X}}_{n+1} - \frac{1}{240}\nabla^2\ddot{\overline{X}}_{n+1} + \ldots + \sigma_k^*\nabla^{k-2}\ddot{\overline{X}}_{n+1}\right] \quad (1)'$$

$$\text{(Predictor)} \quad \dot{\overline{X}}_{n+1} = h\left[{}^{I}\overline{S}_n + \frac{1}{2}\ddot{\overline{X}}_n + \frac{5}{12}\nabla\ddot{\overline{X}}_n + \ldots + \alpha_k\nabla^{k-1}\ddot{\overline{X}}_n\right] \quad (2)$$

$$\text{(Corrector)} \quad \dot{\overline{X}}_{n+1} = h\left[{}^{I}\overline{S}_n + \frac{1}{2}\ddot{\overline{X}}_{n+1} - \frac{1}{12}\nabla\ddot{\overline{X}}_{n+1} + \ldots + \alpha_k^*\nabla^{k-1}\ddot{\overline{X}}_{n+1}\right] \quad (2)'$$

where

$\overline{X} = (X, Y, Z)$, $\dot{\overline{X}} = (\dot{X}, \dot{Y}, \dot{Z})$, h is the step size and $p = k + 1$ is the order; the ${}^{II}\overline{S}_n$, ${}^{I}\overline{S}_n$ are the 2nd and 1st "sums" respectively.

The coefficients — σ_i, σ_i^*, α_i, α_i^* — are obtained by using the recurrence relationships:

$$\sigma_0 = \sigma_0^* = \alpha_0 = \alpha_0^* = 1$$

$$\alpha_m = 1 - \sum_{j=1}^{m} \frac{\alpha_{m-j}}{j+1} \quad (3)$$

[1] Henrici, P. (1962): "Discrete Variable Methods in Ordinary Differential Equations", John Wiley & Sons, Inc., New York, pp 192-195 and pp 291-293.

$$\alpha_m^* = -\sum_{j=1}^{m} \frac{\alpha_{m-j}^*}{j+1}$$

$$\sigma_m = 1 - \sum_{j=1}^{m} \frac{2h_{j+1}}{j+2} \sigma_{m-j}$$

$$\sigma_m^* = -\sum_{j=1}^{m} \frac{2h_{j+1}}{j+2} \sigma_{m-j}^*$$

$$h_m = \sum_{j=1}^{m} \frac{1}{j}$$

The backward differences are computed using the relationship

$$\nabla^r \ddot{X}_n = \sum_{i=0}^{r} (-1)^i \binom{r}{i} \ddot{X}_{n-i} \,. \qquad (4)$$

The first and second "sums" are defined by

$$\nabla^{-1} \ddot{X}_n = {}^{I}\overline{S}_n \,, \quad \nabla^{-2} \ddot{X}_n = \nabla^{-1}\left({}^{I}\overline{S}_n\right) = {}^{II}\overline{S}_n$$

and are computed by inverting the corrector formulas for $\overline{X}$ and $\dot{\overline{X}}$:

$$
\begin{aligned}
{}^{II}\overline{S}_{n-1} &= \frac{\overline{X}_n}{h^2} - \left[\frac{1}{12} \ddot{X}_n - \frac{1}{240} \nabla^2 \ddot{X}_n + \ldots + \sigma_k^* \nabla^{k-2} \ddot{X}_n\right] \\
{}^{I}\overline{S}_{n-1} &= \frac{\dot{\overline{X}}_m}{h} - \left[\frac{1}{2} \ddot{X}_n - \frac{1}{12} \nabla \ddot{X}_n + \ldots + \alpha_k^* \nabla^{k-1} \ddot{X}_n\right] ,
\end{aligned}
\qquad (5)
$$

then updated using

$$ {}^{I}\overline{S}_n = {}^{I}\overline{S}_{n-1} + \ddot{\overline{X}}_n $$

$$ {}^{II}\overline{S}_n = {}^{II}\overline{S}_{n-1} + {}^{I}\overline{S}_n \,. \tag{6} $$

Remark: In order to keep ${}^{II}\overline{S}$ the same order of magnitude as $\overline{X}$, the equations to be integrated are taken as $h^2\ddot{\overline{X}}$. The above formulas were modified to allow this adjustment.

III. COMPUTATION PROCEDURE

3.1 INTEGRATION STARTING PROCESS

Formulas (1) and (2) required (k) starting values of the solution in order to obtain the (k - 1) backward differences required for the given order (p).

The starting values (See Figure 1) are computed by employing high order Runge-Kutta type formulas (Appendix A) which are particularly suited because of the degree of accuracy obtainable. The computation procedure is then:

(1) Given the initial position* vectors $\overline{X}_o = (X_o, Y_o, Z_o)$, the order (p), and the step size (h), the values $\overline{X}_i = \overline{X}(t_o + ih)$, $i = 1, 2, \dots k - 1$ are computed. (The velocity vectors $\dot{\overline{X}}_i = (\dot{X}_i, \dot{Y}_i, \dot{Z}_i)$ are also produced by this process.) The acceleration vectors $\ddot{\overline{X}}_i = (\ddot{X}_i, \ddot{Y}_i, \ddot{Z}_i)$, $i = 1, 2, \dots k - 1$ are computed using the equations of motion given in Appendix B.

(2) Using (4) with $r = i$, the backward differences $\nabla^i \ddot{\overline{X}}_n$, $i = 1, 2, \dots k -1$ are produced, and using (5) and (6) with $n = k - 1$, ${}^{I}\overline{S}_n$ and ${}^{II}\overline{S}_n$ are obtained.

Once the starting table is complete, the integration proceeds as in Section 3.2.

3.2 COWELL INTEGRATION PROCEDURE

Having generated a table of starting values, (information above the line in Figure 1), the Cowell integration proceeds as follows:

(a) Compute $\overline{X}^p_{n+1}$ $(n = k - 1)$, using the predictor formula in (1).

(b) Compute the backward differences $\nabla^i \ddot{\overline{X}}_{n+1}$, $i = 1, 2, \dots k - 1$ as follows:

---

*For simplicity in describing the procedure, only $\overline{X} = (X, Y, Z)$ and $\ddot{\overline{X}} = (\ddot{X}, \ddot{Y}, \ddot{Z})$ will be used. Also note that no further references to $\dot{\overline{X}} = (\dot{X}, \dot{Y}, \dot{Z})$ will be made since these can be obtained by performing the operations described for integrating $\overline{X}$.

2ND SUM	1ST SUM	FUNCTION	1ST DIFF.	2ND DIFF.	3RD DIFF.	-------	-------	-------
		$\ddot{\bar{X}}_0$						
		$\vdots$						
		$\ddot{\bar{X}}_{n-4}$						$\nabla^{k-1}\ddot{\bar{X}}_n$
		$\ddot{\bar{X}}_{n-3}$	$\nabla\ddot{\bar{X}}_{n-2}$					
		$\ddot{\bar{X}}_{n-2}$		$\nabla^2\ddot{\bar{X}}_{n-1}$				$\nabla^{k-1}\ddot{\bar{X}}_{n+1}$
			$\nabla\ddot{\bar{X}}_{n-1}$		$\nabla^3\ddot{\bar{X}}_n$			
		$\ddot{\bar{X}}_{n-1}$		$\nabla^2\ddot{\bar{X}}_n$				
	${}^{I}\bar{S}_{n-1}$		$\nabla\ddot{\bar{X}}_n$		$\nabla^3\ddot{\bar{X}}_{n+1}$			
${}^{II}\bar{S}_{n-1}$		$\ddot{\bar{X}}_n$		$\nabla^2\ddot{\bar{X}}_{n+1}$				
	${}^{I}\bar{S}_n$		$\nabla\ddot{\bar{X}}_{n+1}$					
${}^{II}\bar{S}_n$		$\ddot{\bar{X}}_{n+1}$						
	${}^{I}\bar{S}_{n+1}$	$\vdots$						
${}^{II}\bar{S}_{n+1}$								

Values above the line are initial starting table values. Given an order $p = k + 1$, the values $\bar{X}_0$ to $\bar{X}_n$ $(n = k - 1)$ are the k required points.

Figure 1

$$\nabla \ddot{\overline{X}}_{n+1} = \ddot{\overline{X}}_{n+1} - \ddot{\overline{X}}_n$$

$$\nabla^2 \ddot{\overline{X}}_{n+1} = \nabla \ddot{\overline{X}}_{n+1} - \nabla \ddot{\overline{X}}_n$$

$$\vdots$$

$$\nabla^{k-1} \ddot{\overline{X}}_{n+1} = \nabla^{k-2} \ddot{\overline{X}}_{n+1} - \nabla^{k-2} \ddot{\overline{X}}_n .$$

(c) Obtain a corrected value for $\overline{X}^c_{n+1}$ using (1)'.

(d) Test $|\overline{X}^p_{n+1} - \overline{X}^c_{n+1}| < \delta$, where ($\delta$) represents a specified allowable tolerance. Since $\overline{X} = (X, Y, Z)$, if any one of the three coordinates fails the test, $\ddot{\overline{X}}_{n+1}$ is evaluated and steps (b), (c), and (d) are repeated with $\overline{X}^{ci}_{n+1}$ replacing $\overline{X}^p_{n+1}$ and $\overline{X}^{ci+1}_{n+1}$ replacing $\overline{X}^c_{n+1}$ in (d).

(e) If the difference of two successive values of the solution in step (d) $< \delta$, the iterative process is complete and the retained $\ddot{\overline{X}}_{n+1}$ is for the former value of $\overline{X}_{n+1}$. (Note: If the first corrected value of $\overline{X}_{n+1}$ is acceptable, the step is complete with only one evaluation of the derivative.)

Since two corrections are usually sufficient [2], the integration is terminated if the number of iterations exceeds three. It might then be advisable to change the order (p) or the step size (h) or if the prescribed predictor-corrector tolerance requirements are not too stringent, the tolerance (δ) might be increased to acquire a more rapid convergence of the predictor-corrector cycle.

(f) This completes the predictor-corrector cycle for the evaluation of a point. The index (n) is updated by one, the sums are updated using (6), and the integration process is continued by returning to step (a).

As the integration proceeds the requested vectors are supplied as output data. If a requested vector is not one generated by the integration process, it is produced by interpolation.

[2] Hull, T. and Creemer, A. (1963): "Efficiency of Predictor-Corrector Procedures", J. ACM, Vol. 10, pp. 291-301.

IV. LOCAL ERROR CONTROL

An improvement in the efficiency of the integration process is obtained by controlling the local truncation error[3]. Associated with the integration formula (1) is an approximation to the local error given by

$$U_n \cong |\sigma_k h^2 \nabla^{p-3} \ddot{X}_n| , \qquad (7)$$

the magnitude of last difference vector retained in the formulas.

It is evident from (7) that controls on the local error are directly dependent on the variability of the stepsize (h) and the order (p). The program has, therefore, been designed to alter these parameters during the integration as U_n varies. In particular, the order and/or stepsize can be varied so that the relation $T_1 \geq |U_n| \geq T_2$ is satisfied for each n, i.e., at each step of the integration, U_n is tested. If $|U_n| < T_2$, the order (p) can be decreased or the stepsize (h) increased. If $|U_n| > T_1$, the order (p) can be increased or the step (h) decreased.

4.1 PROCEDURE FOR CHANGING STEPSIZE

A stepsize change during the integration requires k- values of the solution at the new stepsize to provide the required table (Figure 1). Since, in general, many values of the solution at the old stepsize are available, an interpolation scheme over these values is used to obtain the corresponding values in increments of the new stepsize. We remark that if an increase in stepsize is required, the last point computed is accepted, and if the step is to be decreased, this point is rejected since the associated local error is larger than the allowable upper bound T_1.

In the program, two techniques designed for changing the stepsize are available:

(a) Halving-Doubling:

When using this option, the stepsize is modified by either halving, or doubling the current stepsize. In the case of doubling, the required back points

[3]Velez, C. E. (1967): "Local Error Control and Its Effects on the Optimization of Orbital Integration", NASA X-553-67-366.

are obtained by selecting every other point of the last 2k points. In the case of halving, the last k/2 points are used along with midpoints obtained by interpolation.

(b) Stepsize Computation:

When using this option, the stepsize is computed as a function of the current order and local error estimate, i.e., given an "allowable" local error σ_1, $[\sigma_1 \in (T_2, T_1)]$, the new stepsize h_{opt} is given by

$$h_{opt} = h\left[\frac{\sigma_1}{U_n}\right]^{1/p+2} \tag{8}$$

If $U_n < \sigma_1$, the stepsize is increased, and if $U_n > \sigma_1$, the stepsize is decreased, where the new stepsize is approximately "optimal" with respect to σ_1, i.e., the "largest" stepsize which will allow the local error σ_1 for the given order p.

Once the new stepsize has been computed, the required k values are obtained by interpolation over the backpoints available at the old stepsize.

In both cases, once the required backpoints at the new stepsize are obtained, the computation proceeds starting with step 2 of Section 3.1.

4.2 PROCEDURE FOR CHANGING ORDER

The process of changing the order is less involved than that of changing the step. Because of the type of formulas being used, changing the order amounts to decreasing or increasing the number of terms retained. If the order is to be decreased, one less difference is retained in the computation of subsequent points. If an increase in order is required, the last point is rejected, the order is increased, and the point is recomputed by returning to step 2 of Section 3.1.

Since the local error is allowed to vary through a range, the order being used is one that will satisfy the tolerances but will not necessarily be the smallest or "optimum" order that can be used. An option has been included to test the differences at each step against a tolerance σ_2 until the optimal order corresponding to this tolerance is established. Although $T_2 \leq \sigma_2 \leq T_1$, σ_2 should be close to T_1 to obtain a "smallest" order whose local error will satisfy the upper bound.

4.3 MECHANICS FOR VARYING ORDER AND STEP

Depending on the nature of the orbit being integrated, the step and order can be varied alone, in combination*, or can remain fixed throughout the integration. These operations are referenced as (a) vary order - fixed step, (b) vary step - fixed order, (c) vary order - vary step and (d) fixed order - fixed step modes.

The mechanics of the operations are as follows:

(a) The controls for vary order - fixed step are applied such that for $U_n < T_2$, the order (p) is decreased by one; for $U_n > T_1$, the order is increased by one.

(b) The controls for vary step - fixed order are applied such that for $U_n < T_2$, the stepsize is increased; for $U_n > T_1$, the stepsize is decreased.

(c) The controls for vary order - vary step are applied such that for $U_n < T_2$ and $p \leq L_1$, the stepsize is increased; for $U_n > T_1$ and $p \geq L_2$, the stepsize is decreased, where L_1 and L_2 are integers specifying the lower and upper bounds on the order, i.e., the stepsize is modified if p fails the inequality $L_2 \geq p \geq L_1$. The order (p) is then arbitrarily adjusted to $L_1 + (L_2 - L_1)/2$. After one point is computed at the new stepsize, (p) is optimized.

If, however, $U_n < T_2$ and $p > L_1$, the order is decreased; if $U_n > T_1$ with $p < L_2$, the order is increased.

The modes (a), (b), and (c) when selected are also used to adjust the initial starting table. Mode (a), by varying the order, insures that the best order for the given step is selected. When modes (b) or (c) are selected, the table is restarted if the stepsize requires changing.

*Both step and order optimization can be used in combination with a restriction $\sigma_1 > \sigma_2$.

V. THE MULTIREVOLUTION PROCESS

5.1 FORMULATION

An option has been included to "step ahead" the calculations in multirevolution increments. The multirevolution procedure is a combination extrapolation and integration technique.

A cycle in the algorithm consists of first extrapolating or "predicting" the orbital elements n - revolutions ahead. Then, starting with these extrapolated values, the equations of motion are integrated over one revolution. Finally, the extrapolated values are improved by using a corrector formula along with the information obtained from the single revolution integration.

Consider the following definitions as they relate to the formulas and procedure to be outlined:

(a) f_j — the value of an orbital element at the descending node of the j^{th} revolution. Since this is a 3-coordinate system, every reference to a function actually represents a computation for each coordinate, both position and velocity.

(b) n — the number of revolutions to be stepped ahead.

(c) ∇_n — the backward difference taken at n times the step of ∇, so that if $\nabla f_j = f_j - f_{j-1}$, then $\nabla_n f_j = f_j - f_{j-n}$.

(d) k — the order of the highest backward difference ∇_n^i to be retained in the multirevolution formulas.

The formulas used in the multirevolution process[4] are:

(Predictor) $$\Delta_n f_j = n \sum_{i=0}^{k} \alpha_i \nabla_n^i \left(\Delta f_j\right) \qquad (9)$$

[4]Velez C. (1967): Notes on the Numerical Integration of Orbits in Multirevolution Steps, NASA X-542-67-341.

(Corrector)
$$\nabla_n f_j = n \sum_{i=0}^{k} \alpha_i^* \nabla_n^i \left(\Delta f_j\right) \tag{10}$$

where

$$\Delta f_j = f_{j+1} - f_j$$

$$\alpha_i = 1 - \left[\sum_{j=1}^{i} b_j \, \alpha_{i-j}\right] \qquad i = 1, 2, 3 \ldots .$$

$$\alpha_i^* = - \sum_{j=1}^{i} b_j \, \alpha_{i-j}^* \qquad i = 1, 2, 3 \ldots .$$

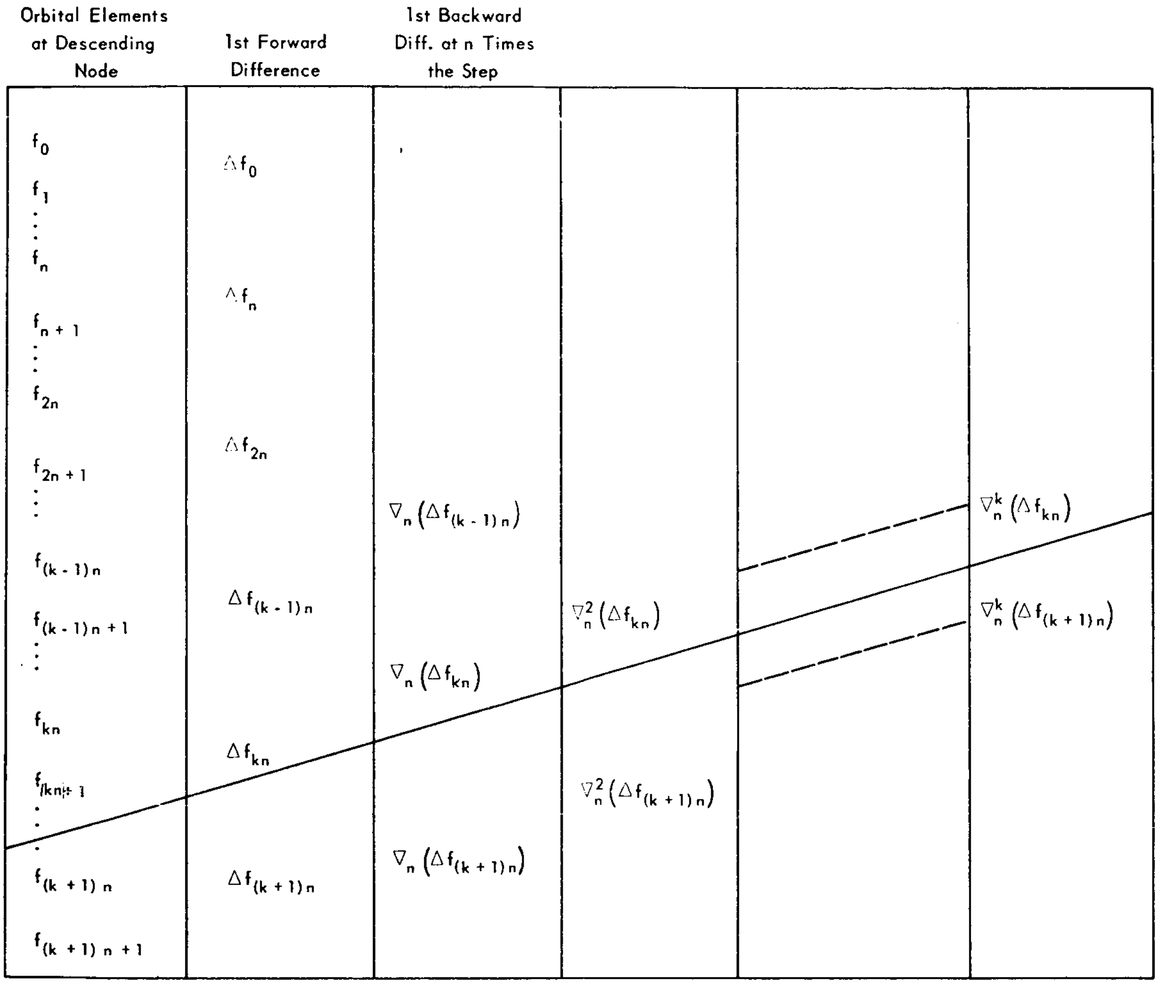
$$b_k = \frac{(-1)^{k+1} \prod_{j=1}^{k} \left(\frac{1}{n} + j\right)}{(k + 1)\,!} \qquad k = 1, 2, 3, \ldots .$$

where

$$\alpha_0, \; \alpha_0^*, \; b_0 = 1$$

5.2 COMPUTATION PROCEDURE FOR MULTIREVOLUTION ALGORITHM

Just as in the Cowell integration, the multirevolution formulas require a set of starting values. This starting table (see Figure 2) is obtained by integrating using the Cowell procedure until a sufficient number ($kn + 2$) of values of orbital elements at the descending node of a revolution are obtained. (Values of the orbital elements at the descending node of any revolution are obtained by inverse interpolation, i.e., a direct interpolation is made to find the time of nodal crossing, followed by an inverse interpolation to obtain the values of the orbital elements.) Then the forward differences $\Delta f_i = f_{i+1} - f_i$, $i = 0, n, 2n, \ldots \ldots . kn$ are computed, and from these values the differences $\nabla_n^i (\Delta f_{kn})$, $i = 1, 2, \ldots . k$; where



Values above the line are required starting information.
Those below the line are then produced using the extrapolation procedure.

Figure 2

$$\nabla_n (\Delta f_{kn}) = \Delta f_{kn} - \Delta f_{(k-1)n} \tag{11}$$

$$\nabla_n^2 (\Delta f_{kn}) = \Delta f_{kn} - 2\Delta f_{(k-1)n} + \Delta f_{(k-2)n}$$

$$\vdots$$

Having obtained the necessary starting information, the extrapolation cycle can begin:

(1) Extrapolate n-revolutions ahead using (9) with $j = kn$ to obtain $\Delta_n f_{kn}$.

(2) Compute a predicted (extrapolated) value $f_{(k+1)n}$, — the element at the descending node of the $(k + 1)\,n^{th}$ revolution by

$$f_{(k+1)n} = \Delta f_{kn} + f_{(k-1)(n+1)}.$$

(3) Using the extrapolated values, integrate one revolution to the next node. Then compute the forward difference $\Delta f_{(k+1)n} = f_{(k+1)} - f_{(k+1)n}$.

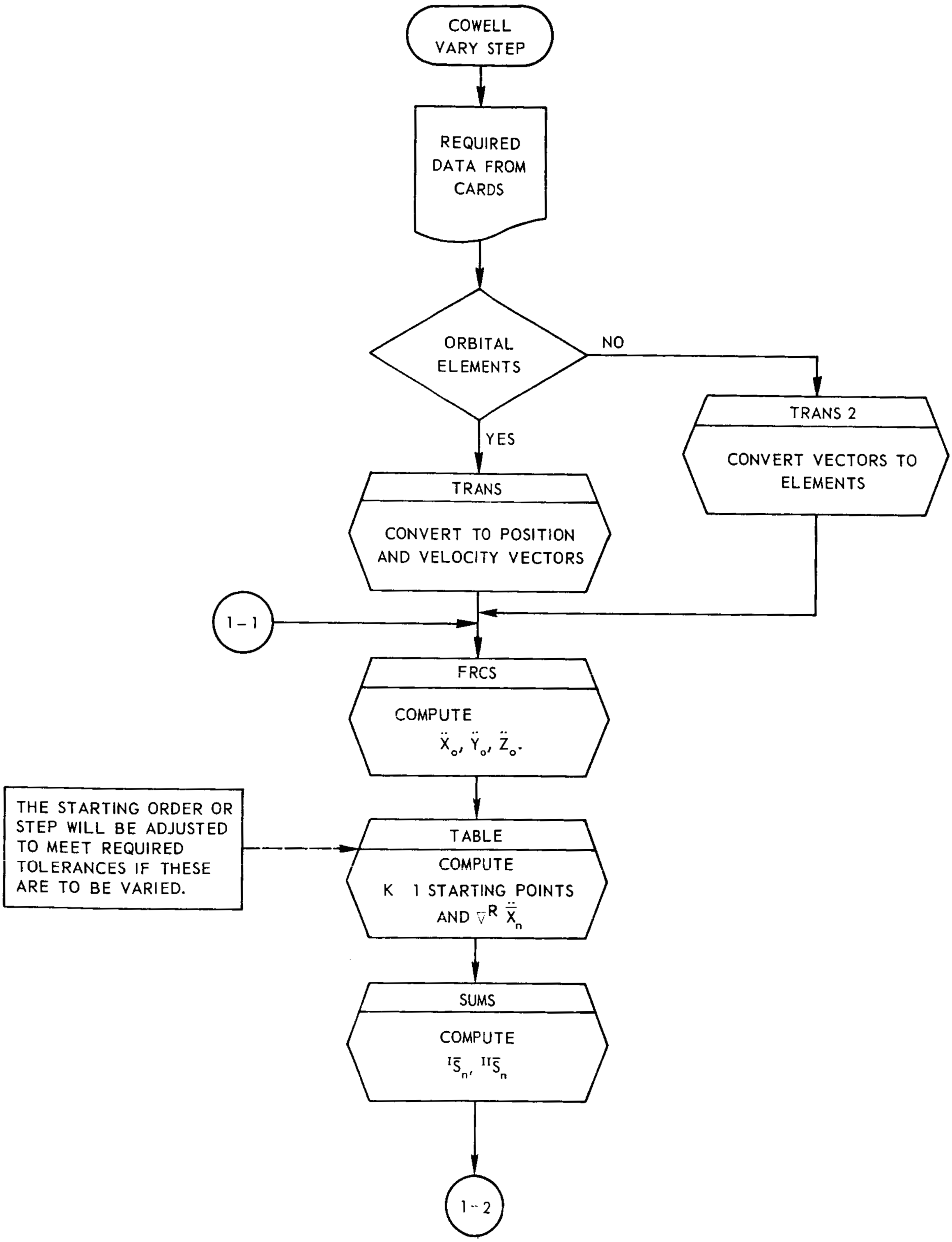
(4) Compute the backward differences $\nabla_n^i (f_{(k+1)n})$ $i = 1, 2 \ldots k$ using (11).

(5) Using (10) to obtain a corrected $\nabla_n(f_{(k+1)n})$, a corrected value of the orbital elements $f_{(k+1)n} = \nabla_n(f_{(k+1)n} + f_{(k-2)n+1})$ is computed and the cycle is complete.

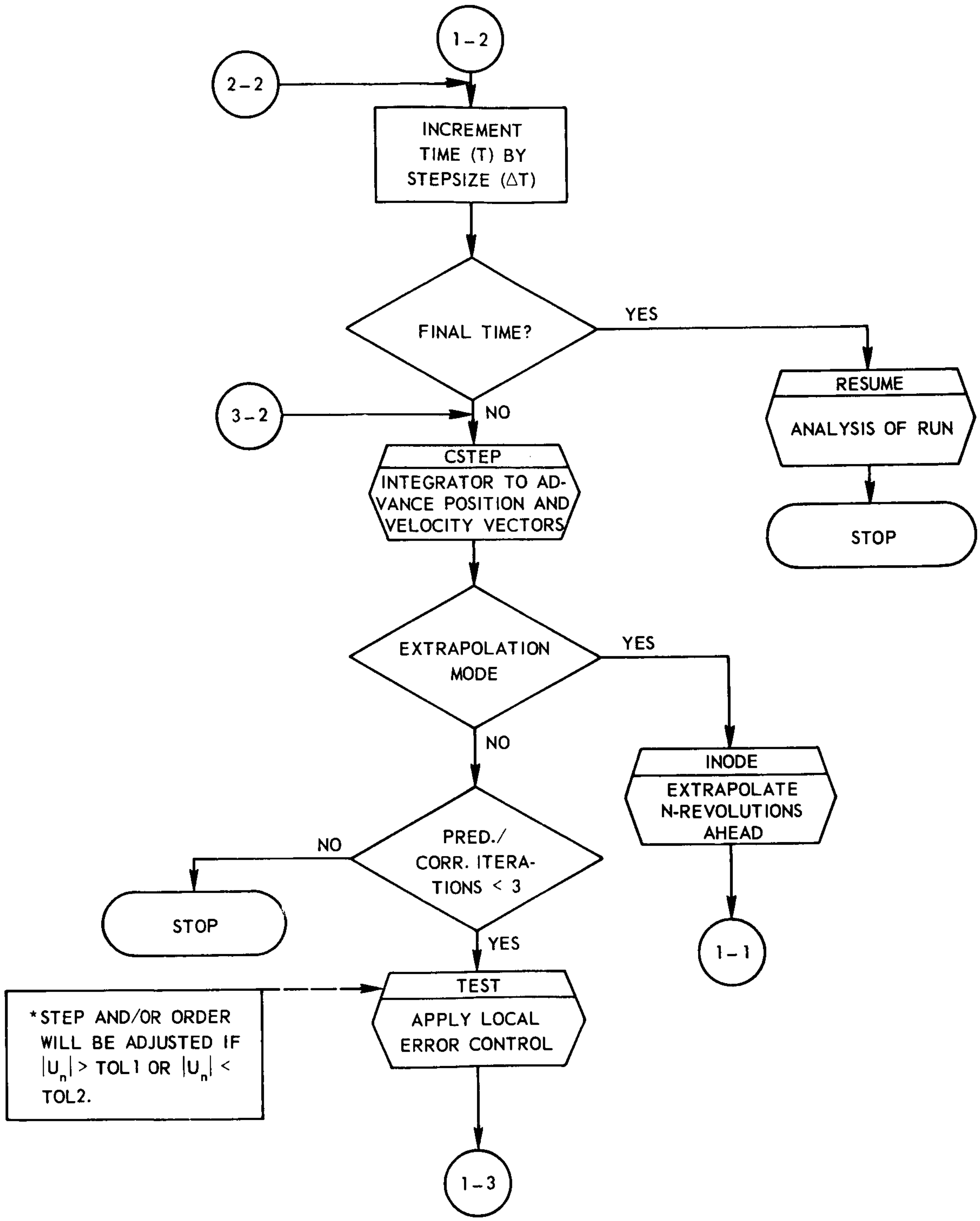
The multirevolution process is continued by repetition of the 5 cycle steps with (j) in step 1 updated for the current revolution, i.e., $j = (k + 1)n, (k + 2)n, \ldots$.

Remark: There is an option in the program to delete step (5), i.e., to extrapolate using only the predictor formula.

VI. FLOW CHARTS

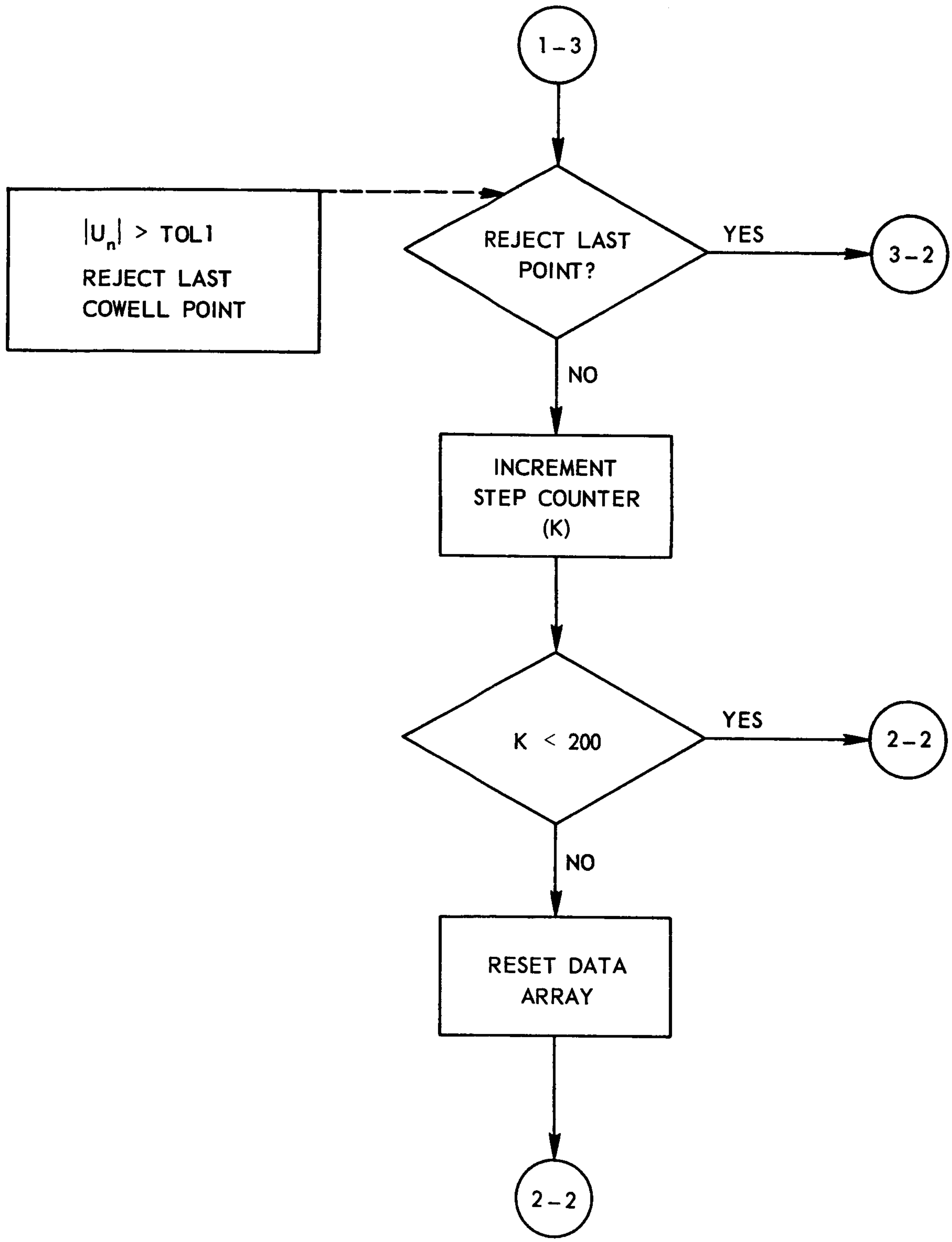


Flow Chart A



*REQUIRED BACKPOINTS FOR A STEP CHANGE ARE COMPUTED USING INTERPOLATION.

Flow Chart B


1 – 3
$|U_n| > TOL1$
REJECT LAST
COWELL POINT
REJECT LAST
POINT?
YES
3 – 2
NO
INCREMENT
STEP COUNTER
(K)
K < 200
YES
2 – 2
NO
RESET DATA
ARRAY
2 – 2

Flow Chart C

VII. OPERATING INSTRUCTIONS

7.1 SEQUENCE OF INPUT DATA CARDS

1. Logical switch settings

2. Cowell Integration beginning and ending times

3. Step sizes for Runge Kutta integration

4. Epoch date

5.-6. Orbital elements at epoch

7. Tolerances for controlling local error

8. Mode of Operation

9. Cowell integration order, step or order limits.

10. Special print out interval

11. Extrapolation mode — order and stepped revolutions.

During a computer application in which several integrations are to be performed, data cards 1, through 9 are required for the first integration while successive integrations require only the logical switch settings (Card 1) and changed information.

7.2 DESCRIPTION OF INPUT CARDS

Card #1 — Format (30L1, I1) A true (T) in the column corresponding to the ISWT index number indicates:

Column	Format	Remarks
1	T	ISWT(1) — New run — Read all new parameters.
2	T	ISWT(2) — Repeat last run — Different mode.
3	T	ISWT(3) — Repeat last run — Different orbital elements.

Column	Format	Remarks
4	T	ISWT(4) — Repeat last run — Different tolerances.
5	T	ISWT(5) — Repeat last run — Different order or step.
6	T	ISWT(6) — Repeat last run — Different integration times.
7	T	ISWT(7) — Use with options other than 2 or 5 to suppress print out except for a specified portion of the orbit.
8	T	ISWT(8) — Use NPT to produce print out based on the number of integration steps.
9	T	ISWT(9) — Used to indicate that the initial conditions are to be input as position and velocity vectors.
10	T	ISWT(10) — Repeat last run — Change Runge-Kutta step-sizes.
11	T	ISWT(11) — Run in extrapolation mode.
12	T	ISWT(12) — Print error vectors for elliptic motion.
13	T	ISWT(13) — Restart with new extrapolation case.
14	F	ISWT(14) — Not used.
15	T	ISWT(15) — Extrapolate by prediction only. (False — by prediction and correction.)
16	T	ISWT(16) — Run with step optimization.
17	T	ISWT(17) — Run with order optimization.
18	T	ISWT(18) — Print out nodal information.
19	F	ISWT(19) — Not used.
20	F	ISWT(20) — Not used.
21	F	Full earth gravity applied. (True-elliptic motion only.)

Column	Format	Remarks
22	T	Lunar gravity applied.
23	T	Solar gravity applied.
24	T	Atmospheric drag applied.
25	T	Solar radiation applied.
31	I	IND — Integer $\neq 0$, program execution will be terminated.

Card #2 — Format (2D15.8,I8)

Column	Format	Remarks
1 - 15	±X.XXXXXXXXD±XX	T_o — Integration starting time (minutes).
16 - 30	±X.XXXXXXXXD±XX	FT — Integration final time (minutes).
31 - 38	IIIIIIII	NPT — Print out interval. For normal output [ISWT(8) = false] NPT is interpreted as minutes of orbit. If [ISWT(8) = true], NPT is interpreted as the number of integrated points.

Card #3 — Format (2D10.3)

Column	Format	Remarks
1 - 10	±X.XXXD±XX	H1 — Runge Kutta step size when integrating forward (c.u.t.).
11 - 20	±X.XXXD±XX	H2 — Runge Kutta step size for integrating backward.

Card #4 — Format (I6,I4,D7.4)

Column	Format	Remarks
1 - 6	YYMMDD	Year, month, day of epoch.
7 - 10	HHMM	Hours and minutes of epoch date.
11 - 17	XX.XXXX	Seconds of epoch date.

Cards #5-6 — Format (3D24.17/3D24.17)

Column	Format	Remarks
1 - 24	±X.XXXXXXXXXXXXXXXXXD±XX	ELEM(1) — Semi-major axis (c.u.l.).
25 - 48	±X.XXXXXXXXXXXXXXXXXD±XX	ELEM(2) — Eccentricity.
49 - 72	±X.XXXXXXXXXXXXXXXXXD±XX	ELEM(3) — Inclination (rad.).
1 - 24	±X.XXXXXXXXXXXXXXXXXD±XX	ELEM(4) — Mean anomaly (rad.).
25 - 48	±X.XXXXXXXXXXXXXXXXXD±XX	ELEM(5) — Argument of perigee (rad.).
49 - 72	±X.XXXXXXXXXXXXXXXXXD±XX	ELEM(6) — Longitude of node (rad.).
	— or —	
1 - 24	±X.XXXXXXXXXXXXXXXXXD±XX	X1(3) — X_0, Y_0, Z_0, coordinates for initial position vector. (c.u.l.)
25 - 48	±X.XXXXXXXXXXXXXXXXXD±XX	
49 - 72	±X.XXXXXXXXXXXXXXXXXD±XX	
1 - 24	±X.XXXXXXXXXXXXXXXXXD±XX	XD1(3) — $\dot{X}_0$, $\dot{Y}_0$, $\dot{Z}_0$ coordinates for initial velocity vector. (c.u.l./c.u.t.)
25 - 48	±X.XXXXXXXXXXXXXXXXXD±XX	
49 - 72	±X.XXXXXXXXXXXXXXXXXD±XX	

Card #7 — Format (5D10.3)

Column	Format	Remarks
1 - 10	±X.XXXD±XX	TOL1 — Upper tolerance limit for local truncation error control. (T_1)

Column	Format	Remarks
11 - 20	±X.XXXD±XX	TOL2 — Lower tolerance limit for local truncation error control. (T_2)
21 - 30	±X.XXXD±XX	CTOL — Accuracy requirement for integration corrector cycle convergence. (δ)
31 - 40	±X.XXXD±XX	TOL3 — Tolerance used for optimum step computation. (σ_1)
41 - 50	±X.XXXD±XX	TOL4 — Tolerance used for optimum order option. (σ_2)

Card #8 — Format (I1)

Column	Format	Remarks
1	I	Mode — 1 - indicates vary step vary order option. 2 - indicates vary step fixed order option. 3 - indicates vary order fixed step option. 4 - indicates fixed order fixed step option.

Card #9 — (With Mode = 1) Format (2I3)

Column	Format	Remarks
1 - 3	III	L1 — Lower order limit for vary order-vary step mode.
4 - 6	III	L2 — Upper order limit for vary order-vary step mode.

Card #9 — (With Mode = 2) Format (D24.17,I2)

Column	Format	Remarks
1 -24	±X.XXXXXXXXXXXXXXXXXD±XX	DUM — Dummy variable.
25 - 26	II	ORDER — Cowell integration order

Card #9 — (With Mode = 3) Format (D24.17,I2)

Column	Format	Remarks
1 - 24	±X.XXXXXXXXXXXXXXXXXD±XX	DEL — Cowell integration stepsize in minutes.
25 - 26	II	DUM — Dummy variable.

Card #9 — (With Mode = 4) Format (D24.17,I2)

Column	Format	Remarks
1 - 24	±X.XXXXXXXXXXXXXXXXXD±XX	DEL — Integration stepsize (mins.).
25 - 26	II	ORDER — Order to be used for integration.

Card #10 — (With ISWT(7) = TRUE) Format (3D10.3)

Column	Format	Remarks
1 - 10	±X.XXXD±XX	BEGTIM — Starting time for print out (mins.).
11 - 20	±X.XXXD±XX	ENDTIM — End time for print out (mins.).
21 - 30	±X.XXXD±XX	BEGT2 — Start print out at this time and continue to end of run (mins.).

Card #11 — (With ISWT(11) = TRUE) Format (2I3)

Column	Format	Remarks
1 - 3	III	NEXT — Number of revolutions to be "stepped".
4 - 6	III	KEXT — Number of differences retained in the extrapolation formulas.

7.3 BLOCK DATA

Input parameters are also supplied through a block data section of the program. These are parameters required by the force model and may be changed by replacing the values in the following data statements:

(1) DATA GM/1.D0/, AE/1.D0/

GM — gravitational constant times mass of the earth.

AE — semi-major axis of reference ellipsoid in c.u.l.

(2) DATA EMASS (1)/.012299896D0/

EMASS (1) — ratio of mass of moon to mass of earth

(3) DATA EMASS (2)/332951.3D0/

EMASS (2) — ratio of mass of sun to mass of earth.

(4) DATA WE, F, CAPR, CD/.5883369D-01, 298.25D0, 6378.166D0, 2.3D0/

WE — angular rotation of earth

F — earth flattening constant

CAPR — 1 c.u.l. in km.

CD — drag coefficient

(5) DATA AREA, SATMAS/.XXXX——D ±XX, .XXXX——D ±XX/

AREA — area of satellite in grams/cm^2.

SATMAS — mass of satellite in grams

(6) DATA RHO1, RHO2, RHO4/2*.XXXD ±XX, 30.D0/

$\left.\begin{array}{l}\text{RHO1} - 1 \\ \text{RHO2} - 1\end{array}\right\}$ differential correction parameters

RHO4 — atmospheric bulge angle in degrees

(7) DATA ((CS (I, J), I = 1, 20), J = 1, 23)/

CS — harmonic coefficients

VIII. SUBROUTINES USED

The Cowell integration program is designed as a system of subroutines under the control of an executive routine. The FORTRAN name and brief description follow:

EXEC — Directs the integration process.

PCCOFF — Computes the coefficients for the Cowell integration formulas.

Calling Sequence: CALL PCCOFF (PX, PXD, CX, CXD, I).

where I = the number of coefficients to be computed.

TRANS — Converts orbital elements to position and velocity coordinates.

Calling Sequence: CALL TRANS.

TRANS2 — Converts position and velocity coordinates to orbital elements.

Calling Sequence: CALL TRANS2.

TABLE — Using Runge Kutta integration, computes the starting table of values required for the Cowell integration model.

Calling Sequence: CALL TABLE.

FRCS — Computes accelerations using the equations of motion (Appendix B) or elliptic motion.

Calling Sequence: CALL FRCS.

CSTEP — Performs Cowell predictor-corrector cycle.

Calling Sequence: CALL CSTEP.

SUMS — Computes ${}^{I}\bar{S}$ and ${}^{II}\bar{S}$ for Cowell integration.

Calling Sequence: CALL SUMS.

CKDIFF — Computes the i^{th} backward differences $\nabla^{i}\ddot{X}_n$ for Cowell integration.

Calling Sequence: CALL CKDIFF (I).

TEST	— Test and control section for local truncation error associated with Cowell integration. Calling Sequence: CALL TEST.
TABLEB	— Computes the new starting points when a change of step size is required. Calling Sequence: CALL TABLEB.
HEMINT	— Hermite interpolation used to produce backpoints when changing step size or to produce points at print out request times. Calling Sequence: CALL HEMINT.
RK	— Performs Runge Kutta type integration using formulas in Appendix A.
OUTPUT	— Formats printed output at print request times.
RESUME	— Collects information throughout the integration to produce an efficiency summary.
TNODE	— Extrapolates a point n-revolutions ahead.
EXTCF	— Computes coefficients for extrapolation formulas.
FKDIFF	— Computes the differences ∇^i_n of the larger forward differences for extrapolation formulas.
EPHEM	— Computes lunar and solar ephemerides. Calling Sequence: CALL EPHEM.
EPHQAN	— Obtains the lunar and solar quantities for a 5-day interval and obtains the coefficients for a least squares fit to a 4th order polynomial.
EGRAV	— Computes the acceleration due to earth's gravity.
SLGRAV	— Computes the acceleration effect due to solar and lunar gravity.
DRAG	— Computes the acceleration effect due to atmospheric drag.

SOLRAD — Computes the acceleration effect due to solar radiation.

BLOCK DATA — See Section 7.3.

IX. CODING INFORMATION

Defined Symbols*

BEGT2 — start printout at this time and continue printing to end of run.

BEGTIM — begin printout at this time and print to a specified final time (ENDTIM).

CAPR — conversion factor meters/c.u.l.**

CDEL — Cowell integration step size h in c.u.t.**

CDP — atmospheric drag constant

/COFFS/ — name of common block containing CSAVE.

COWL — running time for routine CSTEP.

/COWS/ — common block containing variables S1, S2, PX, PXD, CX, CXD, CTOL, SAVE, ITER.

CSAVE — location for saving position predictor and corrector coefficients for last retained term of series.

C, CC — predictor-corrector coefficients for extrapolation process.

CTOL — accuracy criterion for Cowell corrector cycle convergence.

CX, CXD — arrays of coefficients for Cowell corrector formulas.

DEL — Cowell integration step size in minutes.

---

*Symbols relating to the constants used in the force model have been omitted; see BLOCK DATA subroutine.

**Canonical unit of length (c.u.l.) = 6378.166 kms.
Canonical unit of time (c.u.t.) = 806.81242 secs.

DIFF	—	array containing differences $\nabla^i \ddot{X}_n$ to be used in the Cowell equations for correcting.
ELEM	—	array containing orbital elements.
/EMS/	—	common block containing ELEM.
ENDTIM	—	print out end time to be used with (BEGTIM).
FNP	—	used with print request to adjust interpolation location. (FNP = 2 will cause interpolation between 2nd and 3rd end points of data array.)
FORCS	—	running time for routine FRCS.
FT	—	the total length of integration in mins.
FX FXD FXDD	—	arrays for position, velocity and acceleration coordinates interpolated using the Hermite interpolation subroutine.
H1	—	Runge Kutta step size for forward integration.
IENT	—	number of step sizes selected during the integration.
INCOWL	—	number of integration steps.
/INTERP/	—	common block containing FX, FXD, FXDD, T1, K1, M, M1 (Parameter M1 in HERMITE routine is called L.)
ISCT	—	counter for the number of points at a particular step size.
ISWT	—	set of logical switches to control program operation.
ITER	—	number of Cowell corrector cycle iterations.
KEXT	—	the order to be used for extrapolation.
K1	—	starting point of array to be interpolated using Hermite interpolation.

K	— location of current integrated point(s) in the vector arrays.
L1, L2	— limits on order to be used with vary order — vary step mode to control step size changes.
/LIMITS/	— common block containing TOL1, TOL2, TC, ISCT, ISWT, SW, ORDER, L1, L2, MODE.
M1	— location of interpolated value returned from HEMINT routine.
MODE	— integer value to indicate the manner in which the program is operating. 1 = Vary Step — Vary Order 2 = Vary Step — Fixed Order 3 = Vary Order — Fixed Step 4 = Fixed Order — Fixed Step
M	— order of hermite interpolation polynomial.
NCT	— total number of Cowell steps taken.
NL1, NL2	— maximum and minimum order limits.
NEXT	— number of revolutions to be "stepped" in extrapolation process.
/NODE/	— common section containing XNODE, XDNODE, TNOD, C, CC, KK, J1, NDIFF, NEXT, KEXT, INODE.
NPT	— output interval.
N	— number of differences carried in the integration.
/ODEL/	— common block containing NL1, NL2.
/OPT/	— common block containing BEGTIM, ENDTIM, BEGT2.
ORDER	— order of the Cowell integration formulas.
PERIOD	— computed period of the satellite.

/PERTS/ — common block containing ALT, DEN, HM, WE, ESQ, B, F, CDP, RHO1, RHO2, RHO4, CAPR, AREA, SATMAS, CD, MD.

PXD, PX — arrays of coefficients for Cowell predictor formulas.

SAVE — array containing differences $\nabla^i \ddot{X}_n$ to be used in the Cowell equations for predicting.

S1, S2 — arrays containing first and second sums, ${}^{I}S$, ${}^{II}S$.

SW — set of logical switches for internal program control.

T — elapsed time from epoch in c.u.t.

TC — constant for converting minutes to c.u.t.

TEPOCH — integration starting time in minutes.

/TIMING/ — common block containing stored information for a print out resume.

TI — interpolation time for positions and velocities when using Hermite interpolation.

TNOD — array containing times of nodal crossings.

TT — square of Cowell integration step size in c.u.t.

TREQ — output request time in minutes from epoch.

TOL1, TOL2 — upper and lower bounds on local truncation error.

TOL3, TOL4 — σ_1, σ_2 defining "allowable" local error for step and order computation.

TOUT — elapsed time from epoch in minutes.

TSEC — conversion factor sec/cut.

/WORKER/	—	common block containing X, XD, XDD, XXDD, DIFF, CDEL, TT, T, TREQ, TOUT, K, N.
XDD	—	array for storing acceleration coordinates of the form $h^2 \ddot{X}$.
XDNODE	—	array for storing extrapolated velocities.
XD	—	storage array for integrated velocity coordinates.
XNODE	—	storage array under extrapolation mode for position coordinates.
X	—	storage array for integrated position coordinates.
XXDD	—	value for current acceleration coordinate computed without h^2 in the equation.

X. ERROR CODES

(1) INITIAL TABLE FAILED TOLERANCES ----- PROCEDURE RESTART WITH NEW STEPSIZE.

(2) RUN TERMINATED ... ITERATIONS GREATER THAN 3.
FINAL TIME = .XXX_________D±XX

(3) ORDER CYCLE DOES NOT CONVERGE WITHIN SET LIMITS.
FINAL TIME = .XX__________D±XX

(4) STEP CHANGES GREATER THAN 90. RESUME INCOMPLETE.

XI. STORAGE REQUIREMENTS (1108 configuration)

II.1 INSTRUCTION SPACE REQUIRED

Routine	No. of Decimal Locations Approximate
EXEC	1292
CKDIFF	108
COEFF	161
CSTEP	280
DIFF	126
DNVERT	579
DRAG	302
EGRAV	480
EPHEM	782
EPHQAN	77
EXTCF	107
FIT	72
FKDIFF	114
FRCS	234
HEMINT	594
OUTPUT	348
PCCOFF	188
RESUME	308
RK	826
RYMDI	25
SLGRAV	115
SOLRAD	70
SUMS	77
TABLE	187
TABLEB	690
TEST	794
TNODE	490
TRANS	234
TRANS2	303
YMDAY	44
NTAB$	9
SHADOW	120

11.2 DATA SPACE REQUIRED

Labeled Common	No. of Decimal Locations Approximate
WORKER	3978
LIMITS	61
EMS	14
OPTIM	4
COWS	879
NODE	2886
TIMING	556
INTERP	1085
COFFS	4
ODEL	2
RKT	2
RKST	4
OPTIM	4
OPT	6
TIMES	3
BRIEF	20
CONST	8
CONST1	29
CONST2	14
CONST3	11
COFIT	96
CSUN	13
FMODEL	922
PERTS	433
COFIT	96
DDRAG	6

Additional Storage:	Approximate Locations
Parameters and constants	3672

XII. PROGRAM LISTING

```
@ ELT EXEC,1,671009, 48538
@ EOF
C
C
C
C  COWELL ORBIT ERROR AND TIME ANALYSIS PROGRAM
C
C
C
C
      DOUBLE PRECISION X,XD,XDD,DIFF,PX,PXD,CX,CXD,T,TC,DEL,CDEL,TOUT,S1
     1,S2,X1,XD1,XXDD,TEPOCH,TOL1,TOL2,CTOL,FT,TT,DSQRT,H1,H2,ELEM,
     2ENDTIM,BEGTIM,FT1,PERIOD,CSAVE,STEPD,OLDT,SAVE,DUM,TM,SDEL,S
     3ORD,TEMP3,TEMP4,BEGT2,TOL3,TOL4
      DOUBLE PRECISION FX,FXD,FXDD,TI,TREQ,FNPT,FNP,RSAVE,SFNPT,TEP
      DIMENSION X1(3),XD1(3)
      DOUBLE PRECISION XNODE,XDNODE,TNOD,C,CC
      DOUBLE PRECISION THETG0,THDOT1,THDOT2,DAY1,DMIN,ETIME,
     *                 DRAD,DTWOPI,DRSEC,GM,AE,R,RSQ,RQ,THETG,
     *                 CS,CENTER,EPSEC,DSTART,YMDAY,TC1,DLOG
      DOUBLE PRECISION ALT,DEN,HM,WE,ESQ,B,RHO1,RHO2,RHO3,RHO4,CN,CDP,
     *                 F,CAPR,AREA,SATMAS,CD
      DOUBLE PRECISION XYZ,EMASS,PSUN,CSUBR,SIGMA,TSEC
      INTEGER ORDER
      LOGICAL ISWT,SW
      COMMON/EMS/ELEM(6),TEP
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     1          CDEL,TT,T,TREQ,TOUT,K,N
      COMMON/LIMITS/TOL1,TOL2,TC,ISCT,ISWT(30),SW(20),ORDER,L1,L2,MODE
      COMMON/OPTIM/TOL3,TOL4
      COMMON/COWS/S1(3),S2(3),PX(63),PXD(63),CX(63),CXD(63),CTOL,SAVE(60
     1          ,3),ITER
      COMMON/NODE/XNODE(200,3),XDNODE(200,3),TNOD(200),C(20),CC(20),
     1KK,J1,NDIFF,NEXT,KEXT,INODE
      COMMON/TIMING/STEPD(90,3),TEMP3,TEMP4
     *          ,OLDT,TM,XM,YM,ZM,XS,IENT,ITERS,INCOWL,ICH
      COMMON/INTERP/FX(60,3),FXD(60,3),FXDD(60,3),TI,K1,M,M1
      COMMON/COFFS/CSAVE(2)
      COMMON/ODEL/NL1,NL2
      COMMON/RKST/H1,H2
      COMMON/OPT/BEGTIM,ENDTIM,BEGT2
      COMMON/TIMES/TAB,COWL,FORCS
      COMMON/BRIEF/RSAVE(10)
      COMMON/CONST/DRAD,DTWOPI,DRSEC,RAD,RESEC
      COMMON/CONST1/THETG0(10),THDOT1,THDOT2,DMIN,ETIME,IYBEG
      COMMON/CONST2/GM,AE,R,RSQ,RG,THETG,TC1
      COMMON/CONST3/EMASS(2),XYZ(4),LS
      COMMON/CONST4/PSUN,CSUBR,SIGMA,F
      COMMON/PERTS/ALT(41),DEN(41,2),HM(40,2),WE,ESQ,B,CN,CDP,RHO1,RHO
     *         RHO3,RHO4,CAPR,AREA,SATMAS,CD,MD
      COMMON/COFIT/CS(5,9),DAY1,CENTER,DSTART
      COMMON/CSUN/ASUN,PMOON,EMOON,TASUN(10)
   16 FORMAT (1H0 23X, 20HPROGRAM OPERATING IN, I2, 40H REVOLUTIONS EXTR
     *APOLATION MODE OF ORDER,I2)
  193 FORMAT (2I3)
  194 FORMAT(3D24.17)
  195 FORMAT(2D10.3)
  196 FORMAT(D24.17,I2)
  197 FORMAT(5D10.3)
```

```
198 FORMAT(2D15.8,I8)
199 FORMAT(I1)
200 FORMAT(30L1,I1)
201 FORMAT(3D24.17/3D24.17)
202 FORMAT(1H1,48X28HCOWELL NUMERICAL INTEGRATION)
203 FORMAT(1H0,49X10HTOLERANCES/1H020X4HTOL129X,4HTOL229X4HCTOL/17X,
   *D10.3,2(23X,D10.3))
204 FORMAT(1H040X32HEXTRAPOLATION BY PREDICTION ONLY)
205 FORMAT(1H033X42HEXTRAPOLATION BY PREDICTION AND CORRECTION)
211 FORMAT(1H039X31HRUNGE-KUTTA STEPSIZES IN V.U.T./1H040X,3HH1=D10.
   13X,3HH2=D10.3)
212 FORMAT(1H0,40X11HFINAL TIME=D19.12)
213 FORMAT(1H0,45X,7HPERIOD=D19.12)
217 FORMAT (1H0//,49X,12HORDER LIMITS/ 50X, I3,4X,I3)
244 FORMAT(1H ,////,18X,40HSTEP AND ORDER OPTIMIZATION WITH TOL3 OF
   * D15.8, 12H AND TOL4 OF D15.8)
245 FORMAT (1X,///27X,31HORDER OPTIMIZATION WITH TOL4 OF D15.8)
246 FORMAT(1H ,////,28X,30HSTEP OPTIMIZATION WITH TOL3 OF D15.8)
247 FORMAT(1H1//////////// 26X, 57HVARI-ORDER VARI-STEP COWELL NUMER
   *AL INTEGRATION PROGRAM)
248 FORMAT(1H1//////////////////////////26X58HVARI-ORDER FIXED-STEP COW
   1ELL NUMERICAL INTEGRATION PROGRAM)
249 FORMAT(1H1//////////////////////////26X59HFIXED-ORDER FIXED-STEP
   1WELL NUMERICAL INTEGRATION PROGRAM)
250 FORMAT(1H1        ///////////////////26X58HFIXED-ORDER VARI-STEP COW
   1ELL NUMERICAL INTEGRATION PROGRAM)
251 FORMAT(1H043X24HINITIAL ORBITAL ELEMENTS/1H016X9HS.M. AXIS24X12HEC
   1CENTRICITY24X11HINCLINATION)
252 FORMAT(3(10X,D24.16))
253 FORMAT(1H015X12HMEAN ANOMALY,20X15HARG. OF PERIGEE20X13HLONG. OF N
   1ODE)
350 FORMAT (21H0      TIME IN ROUTINE,1X,A6,F12.5,4H SEC)
355 FORMAT (21H      TIME IN ROUTINE 1X,A6,F12.5)
360 FORMAT (1H1////////////////////60X11HFORCE MODEL)
362 FORMAT(///50X,36HA. GEOPOTENTIAL COEFFICIENTS APPLIED)
364 FORMAT(///50X,18HA. ELLIPTIC MOTION)
366 FORMAT(///50X,24HB. LUNAR GRAVITY APPLIED)
368 FORMAT(///50X,24HC. SOLAR GRAVITY APPLIED)
369 FORMAT(///50X,20HD. ATM. DRAG APPLIED)
372 FORMAT(///50X,26HE. SOLAR RADIATION APPLIED)
370 FORMAT(///50X 11HEPOCH DATE ,I6,1X,I4,1X,D15.4)
    TSEC=806.81242D0
    TC1=TC/8.64D4
    TC=60.D0/TSEC
C   DRAG CONSTANTS
    DO 24 J=1,2
    DO 24 I=1,40
 24 HM(I,J)=(ALT(I)-ALT(I+1))/(DLOG(DEN(I+1,J)/DEN(I,J)))
    CDP=(2.3D0*AREA)/(2.D0*SATMAS)
    F=1.D0/CN
    ESQ=F*(2.D0-F)
    B=(1.D0-F)
    RHO4=RHO4*DRAD
C   SOLAR RADIATION CONSTANT
    SIGMA=(CSUBR*PSUN*AREA)/SATMAS
C   CONVERT TO CUL/CUT**2
    SIGMA=(SIGMA*TSEC**2)/(CAPR*1.D+05)
C   MAXIMUM ORDER LIMITS
    NL1=4
    NL2=30
```

```
      NO=NL2+1
      THDOT1=THDOT1*DRAD
      DO 25 I=1,10
      TASUN(I)=TASUN(I)*RAD
   25 THETG0(I)=THETG0(I)*DRAD
      THDOT2=THDOT1+DTWOPI
C     CONVERSION FROM METERS TO VUL.
      ASUN=ASUN/(CAPR*1.D+03)
      PMOON=PMOON/(CAPR*1.D+03)
      DO 1 I=1,NO
      CALL PCCOFF(PX,PXD,CX,CXD,I)
    1 CONTINUE
  551 READ(2,200)ISWT,IEND
      IF(IEND.NE.0) GO TO 103
      CALL CLOCK(XS)
      TAB=0.0
      COWL=0.0
      FORCS=0.0
      INCOWL=0
      ITERS=0
      ICH=0
      IPT=0
      TEMP3=0.D0
      DO 2 I=1,20
    2 SW(I)=.FALSE.
      DO 3 J=1,3
      DO 3 I=1,20
    3 STEPD(I,J)=0.D0
      IENT=1
C     FNP=INTERPOLATION LOCATION
      FNP=2.D0
      IF(ISWT(8)) FNP=0.D0
      IF(ISWT(1)) GO TO 401
      IF(ISWT(2)) GO TO 404
      IF(ISWT(3)) GO TO 402
      IF(ISWT(4)) GO TO 403
      IF(ISWT(5)) GO TO 405
      IF(ISWT(6)) GO TO 401
      IF(ISWT(10)) GO TO 423
      IF(ISWT(13)) GO TO 425
  401 READ(2,198)TEPOCH,FT,NPT
      FNPT=NPT
      SFNPT=FNPT
      FT1=FT*TC
      TEP=TEPOCH*TC
      IF(ISWT(6)) GO TO 407
  423 READ(2,195) H1,H2
      IF(ISWT(10)) GO TO 407
  402 T=TEPOCH*TC
      TOUT=TEPOCH
      READ(2,192)IEPYMD,IEPHM,EPSEC
  192 FORMAT(I6,I4,D7.4)
      DSTART=YMDAY(IEPYMD,IEPHM,EPSEC)
      IYBEG=IEPYMD/10000
      IF(ISWT(9)) GO TO 409
      READ(2,201)ELEM
      CALL TRANS(X1,XD1)
      GO TO 410
  409 READ(2,201)X1,XD1
      CALL TRANS2(X1,XD1)
```

```
410 DO 412 I=1,3
    X(1,I)=X1(I)
412 XD(1,I)=XD1(I)
    PERIOD=DSQRT(ELEM(1)**3)*6.283185307179586400/TC
    IF(ISWT(3)) GO TO 411
403 READ (2,197) TOL1,TOL2,CTOL,TOL3,TOL4
    IF(ISWT(4))GO TO 407
404 READ(2,199)MODE
405 GO TO (413,415,414,416),MODE
413 READ(2,193) L1,L2
    ORDER=L1+(L2-L1)/2
    CDEL= 2.D0**(-5)
    NO=2*NL2
    GO TO 417
414 ORDER=9
    READ(2,196)DEL
    CDEL=DEL*TC
    NO=NL2
    L1=NL1
    L2=NL2
    GO TO 417
415 CDEL=2.0D0**(-5)
    READ(2,196)DUM,ORDER
    L1=ORDER
    L2=L1
    NO=2*ORDER
    GO TO 417
416 READ(2,196)DEL,ORDER
    SW(8)=.FALSE.
    SW(9)=.FALSE.
    CDEL=DEL*TC
    NO=ORDER
417 SDEL=CDEL
    SORD=ORDER
    IF(ISWT(2).OR.ISWT(5)) GO TO 407
    IF(.NOT.ISWT(7))GO TO 450
    READ(2,194)BEGTIM,ENDTIM,BEGT2
    GO TO 450
425 ISWT(11)=.TRUE.
407 DO 408 I=1,3
    X(1,I)=X1(I)
408 XD(1,I)=XD1(I)
    T=TEPOCH*TC
411 CDEL=SDEL
    ORDER=SORD
    FNPT=SFNPT
450 TT=CDEL*CDEL
    N=ORDER-2
    TREQ=TEPOCH+FNPT
    IF(ISWT(7))  TREQ=TREQ+BEGTIM
    ITT=1
    IF(FT.GT.TEPOCH)  GO TO (418,420,419,421),MODE
    ITT=2
    FNPT=-FNPT
    TREQ=TEPOCH+FNPT
    CDEL=-CDEL
    GO TO (418,420,419,421),MODE
418 WRITE(3,247)
    IF(ISWT(16) .AND. ISWT(17))  GO TO 44
    IF(ISWT(17))  GO TO 45
```

```
      IF(ISWT(16))  WRITE(3,246) TOL3
      GO TO 40
   44 WRITE(3,244) TOL3,TOL4
      GO TO 40
   45 WRITE(3,245) TOL4
   40 WRITE(3,217) L1,L2
      WRITE(3,203)TOL1,TOL2,CTOL
      GO TO 422
  419 WRITE(3,248)
      IF(ISWT(17)) WRITE(3,245) TOL4
      WRITE(3,203) TOL1,TOL2,CTOL
      GO TO 422
  420 WRITE(3,250)
      IF(ISWT(16))  WRITE(3,246) TOL3
      WRITE(3,203)TOL1,TOL2,CTOL
      GO TO 422
  421 WRITE(3,249)
      WRITE(3,203)TOL1,TOL2,CTOL
  422 WRITE(3,251)
      WRITE(3,252)(ELEM(I),I=1,3)
      WRITE(3,253)
      WRITE(3,252)(ELEM(I),I=4,6)
      WRITE(3,213)PERIOD
      WRITE (3,212)FT
      WRITE(3,211) H1,H2
      IF(.NOT.ISWT(11)) GO TO 30
      READ(2,193)NEXT,KEXT
      NDIFF=NEXT*KEXT+2
      DO 35 I=1,3
      DO 35 J=1,INODE
      TNOD(J)=0.0D0
      XNODE(J,I)=0.D0
   35 XDNODE(J,I)=0.D0
      INODE=0
      KK=1
      J1=KEXT+1
      CALL EXTCF
      WRITE(3,16)NEXT,KEXT
      IF(ISWT(15)) WRITE(3,204)
      IF(.NOT.ISWT(15)) WRITE(3,205)
   30 CONTINUE
      WRITE(3,360)
      IF(ISWT(21))  GO TO 32
      WRITE(3,362)
      GO TO 34
   32 WRITE(3,364)
   34 IF(.NOT.ISWT(22))  GO TO 36
      WRITE(3,366)
   36 IF(.NOT. ISWT(23))  GO TO 38
      WRITE(3,368)
   38 IF(.NOT. ISWT(24))  GO TO 39
      WRITE(3,369)
   39 IF(.NOT. ISWT(25))  GO TO 72
      WRITE(3,372)
   72 WRITE(3,370) IEPYMD,IEPHMS,EPSEC
   31 K=1
      RSAVE(4)=T
      OLDT=T
      DAY1=DSTART
      CALL EPHGAN
```

```
      CALL FRCS
      CALL CLOCK(XM)
      CALL TABLE
      CSAVE(1)=PX(N+3)
      CSAVE(2)=CX(N+3)
      PX(N+3)=0.D0
      CX(N+3)=0.D0
      RSAVE(1)=DBLE(FLOAT(ORDER))
      RSAVE(2)=CDEL/TC
      STEPD(1,1)=CDEL
      TEMP4=CDEL
      M=ORDER
      TOUT=T/TC
      IF(ISWT(8))  GO TO 17
   15 GO TO (19,18),ITT
   19 IF(TREQ.GT.TOUT)  GO TO 17
      GO TO 14
   18 IF(TREQ.LT.TOUT)  GO TO 17
   14 TREQ=TREQ+FNPT
      GO TO 15
   17 CONTINUE
      IF(SW(9)) GO TO 550
      CALL CLOCK(YM)
      ZM=YM-XM
      TAB =TAB+ZM
      ISCT=N+1
      IF(NPT-N)20,21,22
   20 NCT=N-(N/NPT)*NPT
      GO TO 23
   21 NCT=0
      GO TO 23
   22 NCT=N
   23 L=K+1
C     TO COMPUTE FIRST AND SECOND SUMS
      CALL SUMS
      IF(SW(12))  GO TO 11
      WRITE(3,202)
   11 CALL CLOCK(XM)
      IF(ISWT(16))  SW(20)=.TRUE.
   52 K=L
   60 CONTINUE
      T=T+CDEL
      TOUT = T/TC+.5D-13
      ETIME=T*TC1+DSTART
      GO TO (61,62),ITT
   61 IF(FT1+(FNP+1.D0)*CDEL .LT.T)  GO TO 300
      GO TO 43
   62 IF(FT1+(FNP+1.D0)*CDEL .GT.T)  GO TO 300
   43 CONTINUE
      IF(ETIME.LE.DAY1)  GO TO 1000
      CALL EPHQAN
 1000 CALL CSTEP
      M=K-3
      SW(12)=.FALSE.
      IF((X(M,3).LT.0.0D0.AND.X(M-1,3).GT.0.0D0).AND.(ISWT(11).OR.ISWT(
     18)))CALL TNODE
      IF(SW(12)) GO TO 31
   41 INCOWL=INCOWL + 1
      ITERS= ITERS + ITER
      IF(ITER.LE.3) GO TO 302
```

```
      WRITE (3,325) TOUT
325   FORMAT (5X, 43HRUN TERMINATED...ITERATIONS GREATER THAN 3./
     *        1H0, 40X, 11HFINAL TIME= D24.12)
      GO TO 550
  302 IF(SW(19)) IPT=IPT+1
      IF(IPT .LT. 50)  GO TO 303
      IPT=0
      SW(19)=.FALSE.
      MODE=1
      IF(L1.EQ.L2)MODE=2
  303 IF(MODE.EQ.4)  GO TO 56
      SW(8)=.FALSE.
      SW(9)=.FALSE.
      CALL TEST
      IF(SW(9)) GO TO 550
      IF(SW(8)) GO TO 60
      ISCT=ISCT+1
   56 IF(SW(14).AND.ISWT(11)) GO TO 50
      IF(ISWT(8)) GO TO 65
      M=MAX0(5,ORDER/2)
   54 TI=TREQ*TC
      GO TO (57,58),ITT
   57 IF(TI.GT.T-FNP*CDEL)  GO TO 50
      GO TO 59
   58 IF(TI.LT.T-FNP*CDEL)  GO TO 50
   59 M1=1
   53 SW(10)=.TRUE.
      K1=K
      CALL HEMINT
      CALL OUTPUT
      TREQ=TREQ +FNPT
      GO TO 54
   65 NCT=NCT+1
      IF(NCT-NPT)50,55,55
   55 NCT=0
      CALL OUTPUT
   50 K=K+1
      IF(K.LE.200)GO TO 60
      DO 51 I=1,3
      DO 51 K=1,NO
      J=200+K-NO
      X(K,I)=X(J,I)
      XD(K,I)=XD(J,I)
   51 XDD(K,I)=XDD(J,I)
      L=NO+1
      IF(ISCT.GT.NO)  ISCT=NO
      RSAVE(4)=T-FLOAT(ISCT-1)*CDEL
      GO TO 52
  300 SW(13)=.TRUE.
      CALL CLOCK(YM)
      T=T-CDEL
      TM=T-OLDT
      ZM=YM-XM
      CALL RESUME
      SW(13)=.FALSE.
  550 PX(N+3)=CSAVE(1)
      CX(N+3)=CSAVE(2)
      IF(.NOT.(ISWT(11).OR.ISWT(18)))GO TO 551
  100 FORMAT(1H0,19HNODAL CROSSING DATA/1H012X4HNODE27X1HX27X1HY27X1HZ)
  101 FORMAT(13X,I3,13X,3(5X,D24.16))
```

```
  102 FORMAT(4(5X,D24.16))
      WRITE(3,100)
      DO 104 I=1,INODE
      J=1
      WRITE(3,101)J,(XNODE(I,L),L=1,3)
      TNOD(I)=TNOD(I)/TC
  104 WRITE(3,102)TNOD(I),(XLNODE(I,L),L=1,3)
      GO TO 551
  103 CONTINUE
      STOP
      END
@ ELT CKDIFF,1,670830, 47578
@ EOF @
      SUBROUTINE CKDIFF (LD)
C     LD- INTEGER SIGNIFYING DIFFERENCE TO BE COMPUTED.
      DOUBLE PRECISION X,XD,XDD,XXDD,DIFF,CDEL,TT,T,TREQ,TOUT
      DOUBLE PRECISION BINC,FL,FI
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     1                    CDEL,TT,T,TREQ,TOUT,K,N
      DIMENSION BINC(60)
      IF (K.GT.1) GO TO 2
      DO 1 J=1,3
    1 DIFF(LD,J)=XDD(K,J)-XDD(K-1,J)
      GO TO 4
    2 FL=LD
      BINC(1)=FL
      DO 5 J=1,3
    5 DIFF(LD,J)=XDD(K,J)-BINC(1)*XDD(K-1,J)
      DO 3 I=2,LD
      FI=I
      BINC(I)=(BINC(I-1)*(FL-FI+1.0D0))/FI
      L=K-I
      DO 3 J=1,3
    3 DIFF(LD,J)=DIFF(LD,J)+((-1.0D0)**I*BINC(I)*XDD(L,J))
    4 RETURN
      END
@ ELT COEFF,1,670830, 47578
@ EOF @
      SUBROUTINE COEFF(X,T,N,ORDER,A)
C
C    X          (N)           DEPENDENT VARIABLES
C    T          (N)           INDEPENDENT VARIABLES
C    N                        NUMBER OF INPUT VARIABLES (MAXIMUM OF 20)
C    ORDER                    ORDER OF POLYNOMIAL FIT (MAXIMUM OF 7)
C    A*         (ORDER+1)     COEFFICIENTS OF POLYNOMIAL FIT
C
      DOUBLE PRECISION X(1)
      DOUBLE PRECISION T(1),A(1),SUM,BT,BTB
      COMMON/SCRTCH/BT(8,20),BTB(8,8),SUM,M,G1(49)
      INTEGER ORDER
      M=MIN0(ORDER+1,8)
      DO 100 I=1,N
      BT(1,I)=1.0D0
      DO 100 J=2,M
100   BT(J,I)=T(I)*BT(J-1,I)
      DO 300 I=1,M
      DO 300 J=1,M
      SUM=0.0D0
      DO 200 K=1,N
200   SUM=SUM+BT(I,K)*BT(J,K)
```

```
300    BTB(I,J)=SUM
       CALL DNVERT(M,BT3,B,A)
       DO 500 I=1,M
       A(I)=0.0D0
       DO 500 J=1,N
       SUM=0.0D0
       DO 400 K=1,M
400    SUM=SUM+BTB(I,K)*BT(K,J)
500    A(I)=A(I)+SUM*X(J)
       RETURN
       END
@ ELT CSTEP,1,670830, 47580
@ EOF @
       SUBROUTINE CSTEP
       DOUBLE PRECISION X,XD,XDD,XXDD,DIFF,CDEL,TT,T,TREQ,TOUT
       DOUBLE PRECISION S1,S2,PX,PXD,CX,CXD,CTOL,SAVE
       DOUBLE PRECISION SUM1,SUM2
       DOUBLE PRECISION SC0,SPC0
       COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     1                CDEL,TT,T,TREQ,TOUT,K,N
       COMMON/COWS/S1(3),S2(3),PX(63),PXD(63),CX(63),CXD(63),CTOL,SAVE(60
     1           ,3),ITER
       COMMON/TIMES/ TAB, COWL, FORCS
       DIMENSION SC0(3),SPC0(3)
       INTEGER B
       CALL CLOCK (XE)
       DO 1 I=1,3
       SUM1=0.D0
       SUM2=0.D0
       DO 2 J=1,N
       B=N-J+1
       SUM1=SUM1+PX(B+3)*DIFF(B,I)
     2 SUM2=SUM2+PXD(B+2)*DIFF(B,I)
       SC0(I)=SUM1 + PX(3)* XDD(K-1,I)
       X(K,I)= SC0(I) + S2(I)
     1 XD(K,I)= (SUM2 + PXD(2)*XDD(K-1,I)+S1(I))/CDEL
       CALL FRCS
       DO 15 I=1,3
       SAVE(1,I)=DIFF(1,I)
       DIFF(1,I)=XDD(K,I) - XDD(K-1,I)
       DO 7 J=2,N
       SAVE(J,I)=DIFF(J,I)
     7 DIFF(J,I)=DIFF(J-1,I)-SAVE(J-1,I)
    15 CONTINUE
       ITER=0
   100 DO 5 I=1,3
       SPC0(I)=SC0(I)
       SUM1=0.D0
       SUM2=0.D0
       DO 3 J=1,N
       B=N-J+1
       SUM1=SUM1+CX(B+3)*DIFF(B,I)
     3 SUM2=SUM2+CXD(B+2)*DIFF(B,I)
       SC0(I)=SUM1+CX(3)*XDD(K,I)
       X(K,I)=SC0(I) + S2(I)
     5 XD(K,I)= (SUM2-CXD(2)*XDD(K,I) + S1(I))/CDEL
       ITER=ITER+1
       DO 10 I=1,3
       IF(DABS(SPC0(I)-SC0(I)).LE.CTOL) GO TO 11
    10 CONTINUE
```

```
      CALL FRCS
      DO 13 I=1,3
      DIFF(1,I)=XDD(K,I)-XDD(K-1 ,I)
      DO 13 J=2,N
   13 DIFF(J,I)=DIFF(J-1,I)- SAVE(J-1,I)
      IF (ITER .LE. 3) GO TO 100
      RETURN
   11 DO 12 I=1,3
      S1(I)=S1(I) + XDD(K,I)
   12 S2(I)=S1(I)+S2(I)
      CALL CLOCK (XR)
      COWL= COWL + XR - XE
      RETURN
      END
@ ELT DIFF,1,670830, 47581
@ EOF @                                                                       @
C                                                                       DIFF   7
C PURPOSE          TO CALCULATE THE DIFFERENCE BETWEEN ANY TWO TIME     DIFF   8
C                  POINTS IN THE 20TH CENTURY                           DIFF   9
C                                                                       DIFF  10
C                                                                       DIFF  11
C CALLING SEQUENCE       CALL DIFF(IYMD1,IHMS1,IYMD2,IHMS2,IDAY,ISEC)
C                                                                       DIFF  13
C SYMBOL           TYPE        DESCRIPTION                              DIFF  14
C                                                                       DIFF  15
C IYMD1(1)         I           INPUT - DATE IN THE FORM YYMMDD
C                                                                       DIFF  17
C IHMS1(1)         I           INPUT - TIME ON DATE IYMD1 IN FORM YYMMDD
C                                                                       DIFF  19
C IYMD2(1)         I           INPUT - DATE IN THE FORM YYMMDD
C                                                                       DIFF  21
C IHMS2(1)         I           INPUT - TIME ON DATE IYMD2 IN FORM HHMMSS
C                                                                       DIFF  23
C IDAY(1)          I           OUTPUT - ELAPSED FULL DAYS DIFFERENCE    DIFF  24
C                                       IDAY IS NEGATIVE IF IYMD2,IHMS2
C                                       IS THE EARLIER TIME
C                                                                       DIFF  28
C ISEC(1)          I           OUTPUT - REMAINDER OF DIFFERENCE IN SECONDS.  DIFF  29
C                                       ISEC FOLLOWS THE SIGN OF IDAY
C                                                                       DIFF  32
C                                                                       DIFF  33
C ROUTINES REQUIRED            RYMDI
C                                                                       DIFF  39
      SUBROUTINE DIFF(IYMD1,IHMS1,IYMD2,IHMS2,IDAY,ISEC)
C
      DIMENSION NYEAR(12),LYEAR(12)
C
C----------SET THE ELAPSED DAYS AT THE BEGINNING OF EACH MONTH
C                  SINCE THE START OF A NON-LEAP YEAR
C
      DATA NYEAR /0,31,59,90,120,151,181,212,243,273,304,334/
C
C----------SET THE ELAPSED DAYS AT THE BEGINNING OF EACH MONTH
C                  SINCE THE START OF A LEAP YEAR
C
      DATA LYEAR /0,31,60,91,121,152,182,213,244,274,305,335/
C
      IYEAR1=0
      IYEAR2=0
C
```

```
C----------CHECK FOR A DIFFERENCE OF LESS THAN ONE DAY
C
      IF(IYMD1.EQ.IYMD2) GOTO 4000
C
C----------SEPARATE IYMD1 AND IYMD2 INTO THREE WORDS EACH
C
      CALL RYMDI(IYMD1,IY1,IM1,ID1)
      CALL RYMDI(IYMD2,IY2,IM2,ID2)
C
C----------COMPUTE THE ELAPSED DAYS SINCE JANUARY 0,1900
C
      IYEAR1=36525*(IY1-1)/100
      IYEAR2=36525*(IY2-1)/100
C
C----------CORRECT THE VALUES FOR LEAP YEAR
C
      IF(MOD(IY1,Z).EQ.0) GOTO 1000
      IYEAR1=IYEAR1+NYEAR(IM1)+ID1
  500 IF(MOD(IY2,4).EQ.0) GOTO 2000
      IYEAR2=IYEAR2+NYEAR(IM2)+ID2
      GOTO 3000
 1000 IYEAR1=IYEAR1+LYEAR(IM1)+ID1
      GOTO 500
 2000 IYEAR2=IYEAR2+LYEAR(IM2)+ID2
C
C----------CONVERT ELAPSED DAYS INTO ELAPSED SECONDS
C
 3000 IYEAR1=IYEAR1*86400
      IYEAR2=IYEAR2*86400
C
C----------CALCULATE ELAPSED SECONDS INTO EACH DAY
C
 4000 ISEC1=IHMS1-40*(IHMS1/100)-2400*(IHMS1/10000)
      ISEC2=IHMS2-40*(IHMS2/100)-2400*(IHMS2/10000)
C
C----------SUBTRACT THE TWO ELAPSED SECONDS VALUES
C
      ISEC=IYEAR2+ISEC2-IYEAR1-ISEC1
C
C----------COMPUTE IDAY
C
      IDAY=ISEC/86200
C
C----------COMPUTE ISEC
C
      ISEC=ISEC-IDAY*86400
C
      RETURN
C
      END
@ FLT DNVERT,1,670830, 47582
@ EOF @
      SUBROUTINE DNVERT(N,A,NT,IROW)
      DOUBLE PRECISION A,X,Y,Z
      DIMENSION A(NT,NT),IROW(NT)
      DO 1 I = 1,N
    1 IROW(I) = I
      L  =  1
    2 IF(L.EQ.N)   GO  TO  5
      I  =  L
```

```
      K  =  L + 1
      DO 4 J = K,N
      X = ABS(A(J,L))
      Y = ABS(A(I,L))
      IF(Y.GT.X)GO TO 4
      I  = J
    4 CONTINUE
      IF(I.EQ.L)  GO TO 5
      DO 8 J = 1,N
      Z = A(L,J)
      A(L,J)  = A(I,J)
    8 A(I,J) = Z
      K = IROW(L)
      IROW(L)  = IROW(I)
      IROW(I)  = K
    5 IF (A(L,L)-0.) 9,6,9
    6 PRINT 10
   10 FORMAT( 19H MATRIX IS SINGULAR)
      RETURN
    9 Z= 1.0 / A(L,L)
      IF(L.LE.1) GO TO 13
      K = L-1
      DO  11 I  = 1,K
   11 A(L,I) = A(L,I) * Z
      IF(N.LE.L) GO TO 15
   13 K  = L + 1
      DO 14 I = K ,N
   14 A(L,I)  = A(L,I) * Z
15    IF(L.GE.N) GOTO 23
      DO 22 I = K,N
      IF  (A(I,L)-0.) 18,22,18
   18 IF(L.LE.1) GOTO 20
      JJ =  L - 1
      DO 19 J = 1,JJ
   19 A(I,J) = A(I,J)- A(I,L) * A(L,J)
   20 JJ = L + 1
      DO 21 J = JJ, N
   21 A(I,J)  = A(I,J) - A(I,L) * A(L,J)
      A(I,L)  = - A(I,L) * Z
   22 CONTINUE
   23 A(L,L) = Z
      IF(N.LE.L)  GO  TO  25
      L  = L + 1
      GO TO 2
   25 DO 32 L = 2,N
      K = L - 1
      DO 32 I = 1,K
      IF(A(I,L)-0.) 26,32,26
   26 DO 27 J=1,K
   27 A(I,J) = A(I,J) - A(I,L) * A(L,J)
      IF(N.EQ.L) GOTO 31
      KK = L + 1
      DO 30  J = KK,N
   30 A(I,J) = A(I,J) - A(I,L) * A(L,J)
   31 A(I,L) = - A(I,L) * A(L,L)
   32 CONTINUE
      JJ = N - 1
      DO 37 I = 1 ,JJ
      IF(IROW(I).EQ.I)  GO  TO  37
   33 DO 34 J = I,N
```

```
      IF(IROW(J).EQ.1)  GO  TO  35
   34 CONTINUE
   35 DO 36 K = 1,N
      Z = A(K,I)
      A(K,I) = A(K,J)
   36 A(K,J) = Z
      IROW(J) = IROW(I)
   37 CONTINUE
      RETURN
      END
@ ELT DRAG,1,871807, 48960
@ EOF
      SUBROUTINE DRAG
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     *                CDEL,TT,T,TREQ,TOUT,K,N
      COMMON/CONST2/GM,AE,R,RSQ,RG,THETG,TC1
      COMMON/CONST3/EMASS(2),XYZ(4),LS
      COMMON/PERTS/ALT(41),DEN(41,2),HM(40,2),WE,ESQ,B,CN,CDP,RHO1,
     .               RHO2,RHO3,RHO4,CAPR,AREA,SATMAS,CD,MD
      DOUBLE PRECISION X,XD,XDD,XXDD,DIFF,CDEL,TT,T,TREQ,TOUT
      DOUBLE PRECISION GM,AE,R,RSQ,RG,THETG,TC1,EMASS,XYZ
      DOUBLE PRECISION ALT,DEN,DRAG(3),VR,WE,Q3,Q4,Q5,ESQ,DEL,R7,B,ETA,
     *                 HI,HM,RHO1,RHO2,RHO3,RHO4,COSPSI,RHO,RMAX,RMIN,
     *                 CN,CDP,CAPR,AREA,SATMAS,CD,DEXP
      DOUBLE PRECISION DSQRT,DCOS,DSIN
      DRAG(1)=XD(K,1)+WE*X(K,2)
      DRAG(2)=XD(K,2)-WE*X(K,1)
      DRAG(3)=XD(K,3)
      VR=0.D0
      DO 4 I=1,3
    4 VR=VR+DRAG(I)**2
      VR=DSQRT(VR)
      Q3=X(K,1)**2+X(K,2)**2
      Q4=Q3/RSQ
      Q5=ESQ*Q4
      DEL=(ESQ*X(K,3)*DSQRT(Q3))/(RSQ-ESQ*Q3)
      R7= B*(1.D0+0.5D0*Q5*(1.D0+0.75D0*Q5))
      ETA=DEL*R7/R
      HI=(R-R7)*(1.D0-ETA*DEL/2.0D0)*CAPR
      IF(HI.LT.1.0D+03) GO TO 22
      RHO=0.D0
      GO TO 21
   22 DO 10 I=1,41
      IF (HI.GT.ALT(I)) GO TO 10
      RMAX= DEN(I-1,1)*DEXP((ALT(I-1)-HI)/HM(I-1,1))
      RMIN= DEN(I-1,2)*DEXP((ALT(I-1)-HI)/HM(I-1,2))
      GO TO 15
   10 CONTINUE
   15 COSPSI=X(K,1)/R*(XYZ(1)*DCOS(RHO4))
     *       +X(K,2)/R*(XYZ(2)*DSIN(RHO4))
     *         +X(K,3)/R*XYZ(3)
      COSPSI=(DSQRT((1.D0+COSPSI)/2.D0))**MD
      RHO=(RMIN+(RMAX-RMIN)*COSPSI)*1.0D-10*CAPR
   21 DO 20 I=1,3
   20 DRAG(I)=(1.D0+RHO1)*CDP*(1.D0+RHO2*T)*RHO*VR*DRAG(I)
      DO 25 I=1,3
   25 XXDD(I)=XXDD(I)-DRAG(I)
      RETURN
      END
@ ELT EGRAV,1,670830, 47574
```

```
@ EOF @
      SUBROUTINE EGRAV
C
C****NOTATION****
C
C     PSI-LATITUDE (GEOCENTRIC)
C     LAMBDA-LONGITUDE (+EASTWARD)
C     R-GEOCENTRIC RADIUS TO SATELLITE IN EARTH RADII
C     GM-GRAVITATIONAL CONSTANT TIMES MASS OF EARTH
C     P(M,N)-COEFFICIENTS OF LEGENDRE POLYNOMIAL
C     C(N,M)-COEFFICIENTS OF COSINE FUNCTION
C     S(N,M)-COFFICIENTS OF SINE FUNCTION
C     INDEX1-DEGREE OF SUMMATION PLUS 1
C
      DIMENSION P(22,20),S(20,23),COSLAM(21),SINLAM(21),TPSIM(21),
     .          C(20,23)
C
      DOUBLE PRECISION X,XD,XDD,XXDD,DIFF,CDEL,TT,T,TREQ,TOUT
      DOUBLE PRECISION GM,AE,R,RSQ,RQ,THETG,CS,S,C,RASAT,PR,DR,PLAMDA,
     *                 DLAMDA,PPSI,DPSI,LAMBDA,GMR,PRR,PLXY,PPTP,COSLA
     *                 SINLAM,TPSIM,TANPSI,SINPSI,COSPSI,ZERO,CP3,CL2,
     *                 F1,F2,F3,F4,FN1,RN,RINV,FM,P1,XYSQ,RTXYSQ,DX(3)
     *     ,DATAN2,DSIN,DCOS,DSQRT,P,TC1
C
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     *              CDEL,TT,T,TREQ,TOUT,K,NORD
      COMMON/CONST2/GM,AE,R,RSQ,RQ,THETG,TC1
C     C AND S COEFFICIENTS ARE STORED IN SAME MATRIX
      COMMON/FMODEL/CS(20,23),INDEX1,INDEX3
C
      EQUIVALENCE (TPSIM(2),TANPSI),(C,S,CS),(P,SINPSI),(P(2,1),COSPSI),
     .            (TPSIM,ZERO)
      DATA ZERO,SINLAM(1),COSLAM(1)/2*0.D0,1.D0/
C     SET ZERO ELEMENTS OF LEGENDRE POLYNOMIALS
      DATA P(3,1),P(4,2),P(5,3),P(6,4),P(7,5),P(8,6),P(9,7),
     .     P(10,8),P(11,9),P(12,10),P(13,11),P(14,12),P(15,13),P(16,14),
     .     P(17,15),P(18,16),P(19,17),P(20,18),P(21,19),P(22,20)/
     .     20*0.D0/
C
      RASAT=DATAN2(X(K,2),X(K,1))
      XYSQ=X(K,1)**2+X(K,2)**2
      RTXYSQ=DSQRT(XYSQ)
      LAMBDA=RASAT - THETG
      SINLAM(2)=DSIN(LAMBDA)
      COSLAM(2)=DCOS(LAMBDA)
C
C     SINE,COSINE,AND TANGENT OF LATITUDE
C
      SINPSI=X(K,3)/R
      COSPSI=RTXYSQ/R
      TANPSI= SINPSI/COSPSI
C
      RINV=AE/R
C
C****ALL INDEXES 1 HIGHER THAN IN ANALYSIS****
C
C     CALCULATE POLYNOMIAL TERMS
C               ...NOTE - P TAKES FORM P(M,N)
C
      INDEX2=INDEX1-1
```

```
      CP3=3.D0*COSPSI
      P(1,2)=1.5D0*SINPSI**2-.5D0
      P(2,2)=CP3*SINPSI
      P(3,2)=CP3*COSPSI
      TPSIM(3)=2.D0*TANPSI
C
C     CALCULATE AND SAVE SINES AND COSINES OF LONGITUDE
C
      CL2=2.D0*COSLAM(2)
      SINLAM(3)=CL2*SINLAM(2)
      COSLAM(3)=CL2*COSLAM(2)-1.D0
      DO 120 N=3,INDEX2
      F1=N
      F2=F1-1.D0
      F3=2.D0*F1-1.D0
      F4=F3*COSPSI
      N1=N-1
      N2=N-2
C
C     ZONAL HARMONICS (M = 0)
C
      P(1,N)=(F3*SINPSI*P(1,N1)-F2*P(1,N2))/F1
      IF(INDEX3.LT.2)GO TO 120
      NX=MIN0(N,INDEX3)
      DO 110 M=2,NX
C
C     TESSERAL HARMONICS (M NON ZERO, LESS THAN N)
C
  110 P(M,N)=P(M,N2)+F4*P(M-1,N1)
      IF(NX.LT.N)GO TO 120
      NN1=N+1
C
C     SECTORAL HARMONICS (M EQUAL TO N, NON ZERO)
C
      P(NN1,N)=F4*P(N,N1)
      TPSIM(NN1)=TPSIM(N)+TANPSI
      SINLAM(NN1)=CL2*SINLAM(N)-SINLAM(N1)
      COSLAM(NN1)=CL2*COSLAM(N)-COSLAM(N1)
  120 CONTINUE
C
C     INITIALIZATION FOR SUMMATION FOR PARTIALS
C
  140 PR=0.
      PLAMDA=0.
      PPSI=0.
      FN1=2.
      RN=RINV
C
C     SUMMATION FOR PARTIALS
C
      DO 250 NC=2,INDEX2
      NS=21-NC
      RN=RN*RINV
      FN1=FN1+1.D0
      FM=0.
      DR=0.
      DLAMDA=0.
      DPSI=0.
      N1=MIN0(NC+1,INDEX3)
      DO 200 MC=1,N1
```

```
      MS=24-MC
      P1=P(MC,NC)
      IF(MC.EQ.1)GO TO 150
      FM=FM+1.D0
C
C     PARTIAL WITH RESPECT TO LAMBDA (SUMMATIOM)
C
      DLAMDA=DLAMDA+FM*P1*(S(NS,MS)*COSLAM(MC)-C(NC,MC)*SINLAM(MC))
 150  F1=C(NC,MC)*COSLAM(MC)+S(NS,MS)*SINLAM(MC)
      IF(F1)175,200,175
C
C     PARTIAL WITH RESPECT TO R (SUMMATION)
C
 175  DR=DR+F1*P1
C
C     PARTIAL WITH RESPECT TO PSI (SUMMATION)
C
      DPSI=DPSI+F1*(P(MC+1,NC)-TPSIM(MC)*P1)
 200  CONTINUE
      PR=PR+DR*FN1*RN
      PLAMDA=PLAMDA+DLAMDA*RN
 250  PPSI=PPSI+DPSI*RN
      GMR = GM/R
C
C     COMPLETE PARTIAL WITH RESPECT TO R
C
      PR=-GMR*(1.0D0+PR)/R
C
C     COMPLETE PARTIAL WITH RESPECT TO LAMBDA
C
      PLAMDA=GMR*PLAMDA
C
C     COMPLETE PARTIAL WITH RESPECT TO PSI
C
      PPSI=GMR*PPSI
C
C     CONVERT ACCELERATION IN SPHERICAL COORDINATES TO ACCELERATION IN
C     RECTANGULAR COORDINATES  (MULTIPLY BY MATRIX OF PARTIALS OF
C     SPHERICAL WITH RESPECT TO RECTANGULAR)
C
      PRR=PR/R
      PLXY=PLAMDA/XYSQ
      PPTP=PRR-PPSI*X(K,3)/(RTXYSQ*RSQ)
      DX(1)=X(K,1)*PPTP-PLXY*X(K,2)
      DX(2)=X(K,2)*PPTP+PLXY*X(K,1)
      DX(3)=PRR*X(K,3)+PPSI*RTXYSQ/RSQ
      XXDD(1)=DX(1)
      XXDD(2)=DX(2)
      XXDD(3)=DX(3)
      RETURN
      END
@ ELT EPHEM,1,670927, 50403
@ EOF @
C CALLING SEQUENCE      CALL EPHEM(IY1,D1,AO)
C
C IY1       I     INPUT     YEARS SINCE 1960
C D1        D     INPUT     TIME OF INTEREST IN DAYS SINCE-IY1-
C AO        R     OUTPUT    VECTOR OF NINE ELEMENTS
C                            ELEMENTS 1-3   INERTIAL X,Y,Z COMPONENTS OF
C                                           UNIT VECTOR TO MOON (METERS)
```

```
C                            ELEMENT     4    RANGE TO MMON (METERS)
C                            ELEMENTS 5-7   INERTIAL X,Y,Z, COMPONENTS OF
C                                             UNIT VECTOR TO SUN (METERS)
C                            ELEMENTS    8    RANGE TO SUN (METERS)
C                            ELEMENT     9    EQUATION OF THE EQUINOXES
C
C
      SUBROUTINE EPHEM(IY1,D1,A0)
      DOUBLE PRECISION D,T,TWOPID,DRAD,DRSEC,THDOT1,THDOT2,DUMMY
      DOUBLE PRECISION D1,DMIN,ETIME
      DOUBLE PRECISION SLPT
      DIMENSION A0(9)
      DIMENSION DAYS(10)
      COMMON/CONST/DRAD,TWOPID,DRSEC,RAD,RSEC
      COMMON/CONST1/DUMMY(10),THDOT1,THDOT2,DMIN,ETIME,IYBEG
      COMMON/CSUN/ASUN,PMOON,EMOON,TASUN(10)
C
      REAL LS,LM
C
      EQUIVALENCE (SL,SLMGM),(SP,SLSGS),(CP,CLSGS)
C
C
      DATA DAYS/0.,366.,731.,1096.,1461.,1827.,2192.,2557.,2922.,3288.
C
C     EPHEMERIS DAYS SINCE  JAN 0.5 1900
C
      D=D1+21913.5D0+DAYS(IY1)
C
C     TIME IN JULIAN CENTURIES SINCE JAN 0.5 1900
C
      T=D/36525.0D0
      TSQ=T**2
      DSQ=1.0E-8*D**2
C
C     MEAN OBLIQUITY OF THE ECLIPTIC
C
      EPS=RAD*(23.452294-.0130125*T-.164E-5*TSQ)
C
C     MEAN LONGITUDE OF THE SUN
C
      LS=DMOD(DRSEC*(.10069080407+.12960276813D9*T+1.089*TSQ),TWOPID)
C
      SLS=SIN(LS)
      CLS=COS(LS)
      S2LS=2.*SLS*CLS
      C1=CLS**2
      C2LS=2.*C1-1.
      S3LS=SLS*(3.-4.*SLS**2)
      C3LS=CLS*(4.*C1-3.)
C
C     MEAN LONGITUDE OF PERIGEE OF THE SUN
C
      GS=RSEC*(1012395.0D0+6189.03*T+1.63*TSQ)
C
      SGS=SIN(GS)
      CGS=COS(GS)
C
C     MEAN LONGITUDE OF THE MOON
C
      LM=DMOD(DRAD*(270.434164+13.1763965268D0*D-.85E-4*DSQ),TWOPID)
```

```
C
      SLM=SIN(LM)
      CLM=COS(LM)
      S2LM=2.*SLM*CLM
      C1=CLM**2
      C2LM=2.*C1-1.
      S3LM=SLM*(3.-4.*SLM**2)
      C3LM=CLM*(4.*C1-3.)
C
C     MEAN LONGITUDE OF LUNAR PERIGEE
C
      GM=DMOD(DRAD*(334.329556+.1114040803D0*D-.7739E-3*DSQ),TWOPID)
C
      SGM=SIN(GM)
      CGM=COS(GM)
C
C     LONGITUDE OF MEAN ASCENDING NODE OF THE LUNAR ORBIT
C
      OM=DMOD(DRAD*(259.183275-.529539222E-1*D+.1557E-3*DSQ),TWOPID)
C
      SOM=SIN(OM)
      COM=COS(OM)
      S2OM=2.*SOM*COM
      C2OM=2.*COM**2-1.
      C1=SGS*CLS
      SLSGS=SLS*CGS
      SLSPGS=SLSGS+C1
      SLSGS=SLSGS-C1
      CLSGS=CLS*CGS+SLS*SGS
      SLMGM=SLM*CGM
      C1=SGM*CLM
      SLMPGM=SLMGM+C1
      SLMGM=SLMGM-C1
      CLMPGM=CLM*CGM+SLM*SGM
      S2LMOM=S2LM*COM-SOM*C2LM
      S2LSOM=S2LS*COM-SOM*C2LS
      C2LMOM=C2LM*COM+S2LM*SOM
      S3LMGM=S3LM*CGM-SGM*C3LM
      S3LMGS=S3LM*CGS-SGS*C3LM
      S3LSGS=S3LS*CGS-SGS*C3LS
      C3LSGS=C3LS*CGS-S3LS*SGS
      S2LSGL=S2LS*CLMPGM-SLMPGM*C2LS
C
C     NUTATION IN LONGITUDE
C
      DELPSI=DRSEC*(.2088*S2OM-SOM*(17.2327+.01737*T)-1.273*S2LS
     .       -.2037*S2LM+.1259*SLSGS-.0496*S3LSGS+.0214*SLSPGS
     .       +.0675*SLMGM-.0342*S2LMOM-.0261*S3LMGM+.0114*SLMPGM
     .       +.0124*S2LSOM-.0149*S2LSGL)
C
C     NUTATION IN OBLIQUITY
C
      DELEPS=DRSEC*(9.2106*COM-.0904*C2OM+.552*C2LS+.0884*C2LM
     .              +.0183*C2LMOM+.0216*C3LSGS)
C
C     TRUE OBLIQUITY OF THE ECLIPTIC
C
      AEPS=EPS+DELEPS
      SOB=SIN(AEPS)
      COB=SQRT(1.-SOB**2)
```

```
C
C     EQUATION OF EQUINOXES
C
      A0(9)=COB*DELPSI
C
C     ECCENTRICITY OF EARTH'S ORBIT
C
      ESUN=.01675104-.418E-4*T
C
C     APPARENT LONGITUDE OF THE SUN
C
      D3=DINT(D1)
      D2=D1-D3
      D2=D3*THDOT1+D2*THDOT2
      ALS=LS+2.*ESUN*SIN(D2-TASUN(IY1))
C
      SALS=SIN(ALS)
      CALS=COS(ALS)
C
C     INERTIAL UNIT VECTOR TO SUN
C
      A0(5)=CALS
      A0(6)=SALS*COB
      A0(7)=SALS*SOB
C
C     RADIUS VECTOR TO SUN
C
      PSUN=ASUN*(1.-ESUN**2)
      SLPT =CALS*CGS+SALS*SGS
      A0(8)=PSUN/(1.+ESUN*SLPT)
C
C     TIME IN EPHEMERIS DAYS SINCE JAN 0.5 1960
C
      T1=D-21914.
C
      CL=CLM*CGM+SLM*SGM
      SF=SLM*COM-SOM*CLM
      CF=CLM*COM+SLM*SOM
      SD=SLM*CLS-SLS*CLM
      CD=CLS*CLM+SLS*SLM
      S2L=2.*SL*CL
      C2L=2.*CL**2-1.
      S2D=2.*SD*CD
      C2D=2.*CD**2-1.
      S2F=2.*SF*CF
      C2F=2.*CF**2-1.
      SLPP=SL*CP
      C1=SP*CL
      SLMP=SLPP-C1
      SLPP=SLPP+C1
      SLP2D=SL*C2D
      C1=S2D*CL
      SLM2D=SLP2D-C1
      SLP2D=SLP2D+C1
      CLP2D=CL*C2D
      C1=SL*S2D
      CLM2D=CLP2D+C1
      CLP2D=CLP2D-C1
      SPP2D=SP*C2D
      C1=S2D*CP
```

```
      SPM2D=SPP2D-C1
      SPP2D=SPP2D+C1
      C1=SL*C2F
      C2=S2F*CL
      S4D=2.*S2D*C2D
      C4D=2.*C2D**2-1.
      S=LM-OM+.1096*SL-.0222*SLM2D+.0115*S2D+.0037*S2L
C
C     APPARENT LONGITUDE OF MOON
C
      ALM = 206265.*LM+22640.*SL-4586.*SLM2D
     .     +2370.*S2D+769.*S2L-668.*SP-412.*S2F
     .     -212.*(S2L*C2D-S2D*C2L)-206.*(SLPP*C2D-S2D*(CL*CP-SL*SP))
     .     +192.*SLP2D-165.*SPM2D+148.*SLMP-125.*SD
      ALM =(ALM-109.*SLPP-55.*(S2F*C2D-S2D*C2F)-45.*(C1+C2)
     .     +40.*(C1-C2)-38.*(SL*C4D-S4D*CL)+144.*(.75*SL-SL**3)
     .     -62.*SLM2D*CLM2D+28.*(SLM2D*CP-SP*CLM2D)
     .     -24.*SPP2D+19.*(SL*CD-SD*CL)+18.*(SP*CD+SD*CP))/206265.
C
      SALM=SIN(ALM)
      CALM=COS(ALM)
C
C     APPARENT LATITUDE OF MOON
C
      ALAT=(18520.*SIN(S)*(1.-.00293*SIN(1.403808-.0009242203*T1))
     .    -31.*(SF*CLP2D-SLP2D*CF)-25.*(SF*C2L-S2L*CF)
     .    -23.*(SPM2D*CF+SF*(CP*C2D+SP*S2D))+21.*(SF*CL-SL*CF)
     .    -526.*(SF*C2D-S2D*CF)+44.*(SLM2D*CF+SF*CLM2D))/206265.
C
      SALAT=SIN(ALAT)
      CALAT=SQRT(1.-SALAT**2)
C
C     INERTIAL UNIT VECTOR TO MOON
C
      C1=CALAT*SALM
      A0(1)=CALAT*CALM
      A0(2)=C1*COB-SALAT*SOB
      A0(3)=C1*SOB+SALAT*COB
C
C     RADIUS VECTOR TO MOON
C
      A0(4)=PMOON/(1.+EMOON*CL)
      RETURN
      END
@ ELT EXTCF,1,670830, 47586
@ EOF @
      SUBROUTINE EXTCF
      DOUBLE PRECISION XNODE,XDNODE,TNOD
      DOUBLE PRECISION B,C,ALPHA,FN,FI,SUM,SUM1,CC
      DIMENSION B(20)
      COMMON/NODE/XNODE(200,3),XDNODE(200,3),TNOD(200),C(20),CC(20),
     1KK,J1,NDIFF,NEXT,KEXT,INODE
      N=NEXT
      K=KEXT
      FN=N
      ALPHA=-1.D0/FN
      B(1)=(1.D0-ALPHA)/2.D0
      C(1)= 1.D0-B(1)
      CC(1)=-B(1)
      DO 1 I=2,K
```

```
      FI=I
      B(I)=(FI-ALPHA)*B(I-1)/(FI+1.D0)
      SUM=0.D0
      SUM1=0.D0
      L=I-1
      DO 2 J=1,L
      M=I-J
      SUM1=SUM1+B(J)*CC(M)
    2 SUM=SUM+B(J)*C(M)
      CC(I)=-(SUM1+B(I))
    1 C(I)=1.D0-(SUM+B(I))
      RETURN
      END
@ ELT FIT,1,670830, 47587
@ EOF @
      SUBROUTINE FIT(N,ORDER1,A,T,X)
C
C   N                          NUMBER OF POLYNOMIALS TO FIT
C   ORDER1                     ORDER PLUS ONE OF POLYNOMIALS
C   A          (ORDER1,N)      COEFFICIENTS OF POLYNOMIALS
C   T                          POINT AT WHICH TO EVALUATE POLYNOMIALS
C   X*         (N)             VALUE OF POLYNOMIALS AT 'T'
C
      INTEGER ORDER1
      DOUBLE PRECISION A(5,9),T1,X(9),T
      T1=1.0D0
      DO 100 J=1,N
100   X(J)=A(1,J)
      DO 200 I=2,ORDER1
      T1=T*T1
      DO 200 J=1,N
200   X(J)=X(J)+T1*A(I,J)
      RETURN
      END
@ ELT FKDIFF,1,670830, 47588
@ EOF @
      SUBROUTINE FKDIFF(ARRAY,DIFF,K,M,N)
      DOUBLE PRECISION ARRAY,DIFF,FK,BINC,FI,SUM
      DIMENSION ARRAY(100,3),DIFF(20,3),BINC(20)
      IF(K-2)1,2,2
    1 DIFF(K,N)=ARRAY (M,N)-ARRAY(M-1,N)
      GO TO 4
    2 FK=K
      BINC(1)=FK
      SUM=ARRAY(M,N)-BINC(1)*ARRAY(M-1,N)
      DO 3 I=2,K
      FI=I
      BINC(I)=(BINC(I-1)*(FK-FI+1.0D0))/FI
      L=M-I
    3 SUM=SUM+((-1.0D0)**I*BINC(I)*ARRAY(L,N))
      DIFF(K,N)=SUM
    4 RETURN
      END
@ ELT FRCS,1,671009, 48541
@ EOF @
      SUBROUTINE FRCS
      DOUBLE PRECISION X,XD,XDD,XXDD,DIFF,CDEL,TT,T,TREQ,TOUT,THETG0,
     *                 THDOT1,THDOT2,DMIN,ETIME,GM,AE,R,RSQ,RQ,THETG,
     *                 C,DAY1,CENTER,DAY2,VARD(9),EQ,TTRANS,DAY,FDAY,
     *                 DSTART,DSQRT,TC1,T1
```

```
      DOUBLE PRECISION TOL1,TOL2,TC
      DOUBLE PRECISION XYZ,EMASS
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     *               CDEL,TT,T,TREQ,TOUT,K,NORD
      COMMON/LIMITS/TOL1,TOL2,TC,ISCT,ISWT(30),SW(20),ORDER,L1,L2,MODE
      COMMON/CONST1/THETG0(10),THDOT1,THDOT2,DMIN,ETIME,IYBEG
      COMMON/CONST2/GM,AE,R,RSQ,RQ,THETG,TC1
      COMMON/CONST3/EMASS(2),XYZ(4),LS
      COMMON/COFIT/C(5,9),DAY1,CENTER,DSTART
      COMMON/TIMES/TAB,COWL,FORCS
      LOGICAL ISWT,SW
      EQUIVALENCE(VARD(9),EQ)
      CALL CLOCK(XE)
      RSQ=0.D0
      DO 5 I=1,3
    5 RSQ=RSQ+X(K,I)**2
      R=DSQRT(RSQ)
      RQ=RSQ*R
      IF(ISWT(21))  GO TO 10
      IY1=IYBEG-59
      DAY2=DSTART+T*TC1
      TTRANS=DAY2-CENTER
      T1=1.D0
      DO 100 J=1,9
  100 VARD(J)=C(1,J)
      DO 200 I=2,5
      T1=TTRANS*T1
      DO 200 J=1,9
  200 VARD(J)=VARD(J)+T1*C(I,J)
      DAY=AINT(DAY2)
      FDAY=DAY2-DAY
      THETG=THETG0(IY1)+THDOT1*DAY+THDOT2*FDAY+EQ
      CALL EGRAV
      IF(.NOT.ISWT(22)) GO TO 8
C     COMPUTE LUNAR GRAVITY EFFECTS
      DO 9 I=1,4
    9 XYZ(I)=VARD(I)
      LS=1
      CALL SLGRAV
    8 IF(.NOT.ISWT(23)) GO TO 16
C     COMPUTE SOLAR GRAVITY EFFECTS
      DO 14 I=5,8
      IN=I-4
   14 XYZ(IN)=VARD(I)
      LS=2
      CALL SLGRAV
   16 IF(.NOT. ISWT(24))  GO TO 18
      DO 7 I=5,7
      IN=I-4
    7 XYZ(IN)=VARD(I)
      CALL DRAG
   18 IF(.NOT. ISWT(25))  GO TO 20
C     COMPUTE EFFECTS DUE TO SOLAR RADIATION
      DO 19 I=5,8
      IN=I-4
   19 XYZ(IN)=VARD(I)
      CALL SHADOW (IND)
      IF(IND .EQ.0)  GO TO 20
      CALL SOLRAD
20    DO 6 I=1,3
```

```
  6 XDD(K,I)=XXDD(I)*TT
    GO TO 25
 10 DO 15 I=1,3
    XXDD(I)=-X(K,I)/RQ*GM
 15 XDD(K,I)=XXDD(I)*TT
 25 CONTINUE
    CALL CLOCK(XR)
    FORCS=FORCS + XR - XE
    IF(R.GT.1.D0)  GO TO 30
    WRITE(3,11)
 11 FORMAT (1H0,67HRADIUS VECTOR OF SATELLITE LESS THAN EARTH RADIUS
   *-RUN TERMINATED.)
    STOP
 30 RETURN
    END
@ ELT HEMINT,1,670830, 47590
@ EOF @
    SUBROUTINE HEMINT
C   N=ORDER OF INTERPOLATION POLYNOMIAL - UPPER BOUND =40.
C   X=ARRAY CONTAINING N PLUS ONE EQUI-SPACED DATA POINTS
C   XD=ARRAY FOR VELOCITIES - SPECIFICATIONS AS FOR X
C   H= INTERVAL BETWEEN POINTS
C   TI=TIME AT WHICH POINT IS TO BE INTERPOLATED
C   T= START TIME  T ZERO
C   FX= INTERPOLATED POSITION VALUES
C   FXD= INTERPOLATED VELOCITY VALUES
    DOUBLE PRECISION FX,FXD,FXDD,TI
    DOUBLE PRECISION X,XD,XDD,XXDD,DIFF,CDEL,TT,T,TREQ,TOUT
    DOUBLE PRECISION STEPD,H,TEMP4,OLDT,TM,TEMP3
    DOUBLE PRECISION TOL1,TOL2,TC
    DOUBLE PRECISION HX,HXB,HXB2
    DOUBLE PRECISION SMI,SJMI,SMJ,FJ,SHSQ,COFF
    DOUBLE PRECISION ONE,TWO,THREE,VJX,ZJX,WJX
    DOUBLE PRECISION HDX,HDD,HDBX,HDDB,HXBB,HDBB,HDDBB
    DOUBLE PRECISION SMJSQ,SJMIQ,SMISQ,AQ,SMIQ
    DOUBLE PRECISION FN,FI,LS,NF,S1,S,SH
    DOUBLE PRECISION SMI1,SMI2,FIM1,FNI2,A
    LOGICAL ISWT,SW
    COMMON/INTERP/FX(60,3),FXD(60,3),FXDD(60,3),TI,K1,M,L
    COMMON/LIMITS/TOL1,TOL2,TC,ISCT,ISWT(30),SW(20),ORDER,L1,L2,MODE
    COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
   1              CDEL,TT,T,TREQ,TOUT,K,N
    COMMON/TIMING/STEPD(90,3),H,TEMP4,OLDT,TM,XM,YM,ZM,XS,
   *              IENT,ITERS,INCOWL,ICH
    DIMENSION A(20)
    IF(ISCT .GT. M) GO TO 2
    M=ISCT
    SW(11)=.TRUE.
  2 CONTINUE
    IF(M.LT.5)  SW(11)=.TRUE.
    IF(.NOT.SW(10))  GO TO 3
    TEMP3=H
    H=-CDEL
  3 CONTINUE
    DO 1 J=1,3
    FX(L,J)=0.D0
    FXD(L,J)=0.D0
  1 FXDD(L,J)=0.D0
    LS=1.0D0
    NF=1.0D0
```

```
      S1=0.D0
      FN=M
      S=(T1-T)/H
      SH=S *H
      SHSQ=SH**2
      DO 4 I=1,M
      FI=I
      SMI1=S-(FI-1.0D0)
      LS=LS*SMI1
      NF=NF*FI
    4 S1=S1+1.0D0/SMI1
      NF=NF/FN
      A(1)=((-1.D0)**M*LS)/(NF*S)*(-1.D0)
      DO 5 I=2,M
      FI=I
      SMI1=S-(FI-1.D0)
      SMI2=S-(FI-2.0D0)
      FNI2=(FN-1.D0)-(FI-2.D0)
      FIM1=FI-1.0D0
      A(I)=-1.0D0*A(I-1)*(SMI2/SMI1)*(FNI2/FIM1)
    5 CONTINUE
      J=0
      FJ=J
   10 I=0
      SMI=0.D0
      SMIQ=0.D0
      SJMI=0.D0
      SJMIQ=0.D0
      LN=K1-J
   15 FI=I
      IF (J .EQ. I) GO TO 20
      SMI= SMI + 1.0D0/(S-FI)
      SMIQ=SMIQ+1.D0/(S-FI)**2
      SJMI= SJMI + 1.0D0/(FJ-FI)
      SJMIQ=SJMIQ + 1.D0/(FJ-FI)**2
   20 I=I+1
      IF(I .LT. M) GO TO 15
      SMISQ=SMI**2
      SMJSQ=SJMI**2
      SMJ=S-FJ
      AQ=A(J+1)**3
      IF(SW(11)) GO TO 50
      COFF=A(J+1)**2
      HX=COFF-2.D0*SMJ*SJMI*COFF
      HXB=H*SMJ*COFF
      HXB2=SMJ*COFF/H
   25 DO 30 II=1,3
      FX(L,II)=FX(L,II) + HX*X(LN,II)+HXB*XD(LN,II)
   30 FXD(L,II)=FXD(L,II) + HX*XD(LN,II)+HXB2*XDD(LN,II)
      J=J+1
      FJ=J
      IF(J .LT. M) GO TO 10
      IF(.NOT.SW(10)) RETURN
      H=TEMP3
      SW(10)=.FALSE.
      RETURN
   50 COFF = 3.D0 * AQ/H
      ONE= 3.D0 * SMJSQ + SJMIQ
      TWO= SMJ -3.D0 * SMJ**2 * SJMI
      THREE= 3.D0 * SMISQ - SMIQ
```

```
      VJX= 1.D0 + 3.D0/2.D0 * SMJ**2 * ONE  - 3.D0 * SMJ * SJMI
      ZJX=  H**2/2.D0 * SMJ**2
      WJX=  H*TWO
      HX=(1.D0 + 3.D0/2.D0*SMJ**2 * (3.D0 * SMJSQ + SJMIQ)
     *    - 3.D0 * SMJ*SJMI)*AQ
      HDX= (VJX* SMI + SMJ * ONE - SJMI)* COFF
      HDD= (ONE + 6.D0 * SMI * SMJ * ONE - 6.D0 * SMI*SJMI
     *      + VJX * THREE) * COFF/H
      HXB = (SMJ -3.D0*SMJ**2*SJMI) * H * AQ
      HDBX= (3.D0 * TWO * SMI + 1.D0 - 6.D0 * SMJ * SJMI)* AQ
      HDDB = ( -2.D0 * SJMI + 2.D0 * SMI -12.D0* SMJ * SJMI * SMI
     *       + TWO * THREE) * COFF
      HXBB = 0.5D0*H**2*SMJ**2*AQ
      HDBB = (3.D0/2.D0 * SMJ **2 * SMI  + SMJ)*H*AQ
      HDDBB= (1.D0 +6.D0 * SMJ *SMI +3.D0/2.D0 * SMJ**2 * THREE)*AQ
      DO 60 II=1,3
      FX(L,II)=FX(L,II) + HX * X(LN,II) + HXB*XD(LN,II)+ HXBB*XDD(LN,I)
     1 /H**2
      FXD(L,II)=FXD(L,II)+HDX * X(LN,II) + HDBX * XD(LN,II)
     *          + HDBB * XDD(LN,II)/H**2
      FXDD(L,II)=FXDD(L,II) + HDD* X(LN,II) + HDDB * XD(LN,II)
     *           + HDDBB * XDD(LN,II)/H**2
   60 CONTINUE
      J=J+1
      FJ=J
      IF(J .LT. M) GO TO 10
      H=TEMP3
      SW(10)=.FALSE.
      SW(11)=.FALSE.
      RETURN
      END
@ ELT INPUT,1,671013, 34113
@ EOF @
      BLOCK DATA
C
      DOUBLE PRECISION DRAD,DTWOPI,DRSEC,THETG0,THDOT1,THDOT2,GM,AE,CS
      DOUBLE PRECISION DMIN,TC1,R,RSQ,RQ,THETG,ETIME
      DOUBLE PRECISION ALT,DEN,HM,WE,ESQ,B,RHO1,RHO2,RHO3,RHO4,CN,CDP,
     *                 CAPR,EMASS,XYZ,AREA,SATMAS,CD
      DOUBLE PRECISION PSUN,CSUBR,SIGMA,F
C
      COMMON/CONST/DRAD,DTWOPI,DRSEC,RAD,RSEC
      COMMON/CONST1/THETG0(10),THDOT1,THDOT2,DMIN,ETIME,IYBEG
      COMMON/CONST2/GM,AE,R,RSQ,RQ,THETG,TC1
      COMMON/CONST3/EMASS(2),XYZ(4),LS
      COMMON/CONST4/PSUN,CSUBR,SIGMA,F
      COMMON/PERTS/ALT(41),DEN(41,2),HM(40,2),WE,ESQ,B,CN,CDP,RHO1,RHO2,
     *         RHO3,RHO4,CAPR,AREA,SATMAS,CD,MD
      COMMON/CSUN/ASUN,PMOON,EMOON,TASUN(10)
      COMMON/FMODEL/CS(20,23),INDEX1,INDEX3
C
C   CONVERSION FROM DEGREES TO RADIANS
C   TWO PI IN RADIANS
      DATA DRAD/.017453292519943296D0/,DTWOPI/6.2831853071795864D0/,
C   CONVERSION FROM SECONDS OF ARC TO RADIANS - DOUBLE AND SINGLE PREC.
C   CONVERSION FROM SECONDS OF ARC TO RADIANS
     *   DRSEC/.4848136811095360-5/,RAD/.0174532925/,RSEC/.484813681E-5/
C
C    RIGHT ASCENSION OF GREENWICH AT JAN 0.0 FOR 1960-1969 IN DEGREES
C
```

```
      DATA THETGO /98.6740065D0, 99.4209347D0, 99.1822166D0,
     .              98.9434986D0, 98.7047825D0, 99.4517117D0,
     .              99.2129936D0, 98.9742765D0, 98.7355595D0,
     .              99.4824896D0/,
C    MEAN ADVANCE IN RT ASC OF GREENWICH PER MEAN SOLAR DAY IN DEGREES
C    MINUTES IN 5 DAYS
     *    THDOT1/.9856473354D0/,DMIN/7200.D0/
C    GRAVITATIONAL CONSTANT TIMES MASS OF EARTH IN CANONICAL UNITS
C    SEMI-MAJOR AXIS OF REFERENCE ELLIPSOID IN VANGUARD UNITS OF LENGTH
      DATA GM/1.D0/,AE/1.D0/
C
C     RATIO OF MASS OF SUN TO MASS OF EARTH
C
      DATA EMASS(2)/332948.55D0/
C
C     RATIO OF MASS OF MOON TO MASS OF EARTH
C
      DATA EMASS(1)/.01230012300/
C
C    COMMON BLOCK /CSUN/ CONTAINS CONSTANTS NEEDED TO COMPUTE SOLAR AND
C    LUNAR EPHEMERIDES
C
C     SEMI-MAJOR AXIS OF EARTH'S ORBIT AROUND SUN IN METERS
C
      DATA ASUN   /.1496E12/,
C
C     SEMI LATUS RECTUM OF MOON'S ORBIT IN METERS
C
     .      PMOON  /.384750902E9/,
C
C     ECCENTRICITY OF MOON'S ORBIT
C
     .      EMOON  /.054900489/,
C
C     TRUE ANOMALY OF SUN AT JAN 0.0 FOR 1960-1969 IN DEGREES
C
     .      TASUN  /3.5727587, 2.8430634, 3.0989676, 3.3548717,
     .              3.6107759, 2.8810806, 3.1369848, 3.3928890,
     .              3.6487942, 2.9190979/
C
C     COMMO$ BLOCK CO$ST3 - CSUBR=RADIATIO$ CO$STA$T
C                           PSUN=SOLAR RADIATION PRESSURE IN DYNES/CM**
      DATA PSUN,CSUBR/4.5D-05,2.0D0/
C
C
C     COMMON BLOCK PERTS PARAMETERS FOR ATMOSPHERIC DENSITY AND DRAG.
C     ALTITUDE IN KM
      DATA (ALT(I),I=1,41)/0.D0,.2D+02,.4D+02,.6D+02,.8D+02,.1D+03,
     *                     .12D+03,.14D+03,.16D+03,.18D+03,.2D+03,
     *                     .22D+03,.24D+03,.26D+03,.28D+03,.3D+03,
     *                     .32D+03,.34D+03,.36D+03,.38D+03,.4D+03,
     *                     .42D+03,.44D+03,.46D+03,.48D+03,.5D+03,
     *                     .52D+03,.54D+03,.56D+03,.58D+03,.6D+03,
     *                     .64D+03,.68D+03,.72D+03,.76D+03,.8D+03,
     *                     .84D+03,.88D+03,.92D+03,.96D+03,.1D+04/
C     ATMOSPHERIC DENSITY IN SUNLIGHT - GMS/KM**3
      DATA (DEN(I,1),I=1,41)/.1225D+13,.8891D+11,.39957D+10,.30592D+09,
     *                       .1999D+08,.4974D+06,.249D+05,.3961D+04,
     *                       .1255D+04,.5442D+03,.2799D+03,.1596D+03,
     *                       .9728D+02,.6219D+02,.4119D+02,.2805D+02,
```

```
     *                           .1953D+02,.1385D+02,.9976D+01,.7282D+01,
     *                           .5376D+01,.4033D+01,.3034D+01,.2299D+01,
     *                           .1754D+01,.1349D+01,.1038D+01,.8043D+00,
     *                           .626D+00,.4892D+00,.3838D+00,.2388D+00,
     *                           .1507D+00,.9658D-01,.6289D-01,.417D-01,
     *                           .291D-01,.2075D-01,.15D-01,.111D-01,
     *                           .84D-02/
C    ATMOSPHERIC DENSITY IN EARTH'S SHADOW
      DATA (DEN(I,2),I=1,41)/.1225D+13,.8891D+11,.39957D+10,.30592D+09,
     *                           .1999D+08,.4974D+06,.249D+05,.4104D+04,
     *                           .1282D+04,.5109D+03,.2327D+03,.1159D+03,
     *                           .6174D+02,.3467D+02,.2031D+02,.1231D+02,
     *                           .7675D+01,.4894D+01,.3178D+01,.2094D+01,
     *                           .1397D+01,.9625D+00,.6552D+00,.4497D+00,
     *                           .3109D+00,.2166D+00,.152D+00,.1076D+00,
     *                           .7685D-01,.5545D-01,.4047D-01,.2244D-01,
     *                           .1325D-01,.8397D-02,.5713D-02,.4143D-02,
     *                           .315D-02,.252D-02,.208D-02,.1765D-02,
     *                           .151D-02/
C    DRAG CONSTANTS
      DATA WE,CN,CAPR,CD/.588336903074954198D-01,298.25D0,
     * 6378.166D0,2.3D0/
C    AREA AND MASS OF SATELLITE
      DATA AREA,SATMAS/.91483870D+04,.63502935D+05/
C
C    DIFFERENTIAL CORRECTION PARAMETERS AND ANGLE OF ATM. BULDGE
      DATA RHO1,RHO2,RHO3,RHO4/ 3*0.D0,30.D0/
      DATA MD/6/
C
C  COMMON BLOCK /FMODEL/ CONTAINS THE GRAVITATIONAL COEFFICIENTS AND
C  THEIR INDEXES
C
C     INDEXES FOR COEFFICIENTS
      DATA INDEX1,INDEX3 /16,16/
C
C-----C(2,0) THRU C(20,0)
      DATA (CS(I,1),I=2,20) /-1082.645D-6,2.546D-6,1.649D-6,.210D-6,
     .                       -.646D-6,.333D-6,.270D-6,.530D-7,.540D-7,
     .                       -.302D-6,.357D-6,.114D-6,.179D-6,6*0.D0/
C
C-----C(2,1) THRU C(20,1)
C
      DATA (CS(I,2),I=2,20) /0.D0,2.091D-6,-.543D-6,-.677D-7,-.037D-6,
     .                       .144D-6,-.052D-6,.076D-6,.649D-7,-.313D-7,
     .                       -.923D-7,0.D0,-.788D-8,6*0.D0/
C
C-----C(2,2) THRU C(20,2)
C
      DATA (CS(I,3),I=2,20) /1.536D-6,.251D-6,.738D-7,.102D-6,.858D-8,
     .                       .363D-7,.2135D-8,-.277D-9,-.624D-8,0.D0,
     .                       -.490D-8,8*0.D0/
C
C-----C(3,3) THRU C(20,3)
C
      DATA (CS(I,4),I=3,20) /.782D-7,.509D-7,-.172D-7,-.112D-8,.352D-8,
     .                       -.374D-9,0.D0,-.379D-9,10*0.D0/
C
C-----C(4,4) THRU C(20,4)
C
      DATA (CS(I,5),I=4,20) /-.112D-8,-.206D-8,-.167D-9,-.323D-9,
```

```
                               -.277D-9,0.D0,-.436D-10,10*0.D0/
C
C-----C(5,5) THRU C(20,5)
C
      DATA (CS(I,6),I=5,20) /.384D-9,-.253D-9,.269D-10,-.959D-11,12*0./
     */
C
C-----C(6,6) THRU C(20,6)
C
      DATA (CS(I,7),I=6,20) /-.932D-11,-.145D-10,-.475D-12,12*0.D0/
C
C-----C(7,7) THRU C(20,7)
C
      DATA (CS(I,8),I=7,20) /.102D-11,-.444D-13,12*0.D0/
C
C-----C(8,8) THRU C(20,8)
C
      DATA (CS(I,9),I=8,20) /-.316D-12,12*0.D0/
C
C-----C(9,9) THRU C(20,9)
C
      DATA (CS(I,10),I=9,20) /6*0.D0,-.241D-18,5*0.D0/
C
C-----C(10,10) THRU C(20,10)
C
      DATA (CS(I,11),I=10,20) /11*0.D0/
C
C-----C(11,11) THRU C(20,11)
C
      DATA (CS(I,12),I=11,20) /3*0.D0,.947D-21,6*0.D0/
C
C-----C(12,12) THRU C(20,12)
C
      DATA (CS(I,13),I=12,20) /-.2783D-18,-.110D-18,.504D-19,-.114D-19,
                               5*0.D0/
C
C-----C(13,13) THRU C(20,13)
C
      DATA (CS(I,14),I=13,20) /-.216D-19,0.D0,-.117D-20,5*0.D0/
C
C-----C(14,14) THRU C(20,14)
C
      DATA (CS(I,15),I=14,20) /-.1931D-21,.114D-22,5*0.D0/
C
C-----C(15,15) THRU C(28,15)
C
      DATA (CS(I,16),I=15,20) /6*0.D0/
C
C-----C(16,16) THRU C(20,16)
C
      DATA (CS(I,17),I=16,20) /5*0.D0/
C
C-----C(17,17) THRU C(20,17)
C
      DATA (CS(I,18),I=17,20) /4*0.D0/
C
C-----C(18,18) THRU C(20,18)
C
      DATA (CS(I,19),I=18,20)/3*0.D0/
C
```

```
C-----C(19,19) THRU C(20,19)
C
      DATA (CS(I,20),I=19,20) /2*0.D0/
C
C-----C(20,20)
C
      DATA CS(20,21) /0.D0/
C
C-----S(20,0) THRU S(2,0)
C
      DATA (CS(I,23),I=1,19) / 19*0.D0/
C
C-----S(20,1) THRU S(2,1)
C
      DATA (CS(I,22),I=1,19) /6*0D0,.280D-8,0.D0,-.040D-6,.088D-7,
     .                         -.779D-7,.784D-8,.045D-6,.114D-6,-.212D-7,
     .                         -.882D-7,-.445D-6,.287D-6,0.D0/
C
C-----S(20,2) THRU S(2,2)
C
      DATA (CS(I,21),I=1,19) /8*0.D0,-.023D-8,0.D0,-.250D-8,.242D-8,
     .                         .320D-8,.162D-7,-.455D-7,-.375D-7,.148D-6,
     .                         -.184D-6,-.872D-6/
C
C-----S(20,3) THRU S(3,3)
C
      DATA (CS(I,20),I=1,18) /10*0.D0,.175D-9,0.D0,.404D-10,.254D-9,
     .                         .643D-9,.231D-9,-.114D-7,.226D-6/
C
C-----S(20,4) THRU S(4,4)
C
      DATA (CS(I,19),I=1,17) /10*0.D0,-.654D-10,0.D0,-.157D-10,-.217D-9,
     .                         -.196D-8,.498D-9,.486D-8/
C
C-----S(20,5) THRU S(5,5)
C
      DATA (CS(I,18),I=1,16) /12*0.D0,.214D-10,.191D-10,-.370D-9,
     .                         -.146D-8/
C
C-----S(20,6) THRU S(6,6)
C
      DATA (CS(I,17),I=1,15) /12*0.D0,.888D-11,.473D-11,-.361D-10/
C
C-----S(20,7) THRU S(7,7)
C
      DATA (CS(I,16),I=1,14) /12*0.D0,.158D-12,.018D-10/
C
C-----S(20,8) THRU S(8,8)
C
      DATA (CS(I,15),I=1,13) /12*0.D0,.130D-12/
C
C-----S(20,9) THRU S(9,9)
C
      DATA (CS(I,14),I=1,12)/5*0.D0,-.483D-18,6*0.D0/
C
C-----S(20,10) THRU S(10,10)
C
      DATA (CS(I,13),I=1,11) /11*0.D0/
C
C-----S(20,11) THRU S(11,11)
```

```
C
      DATA (CS(I,12),I=1,10) /6*0.D0,-.473D-21,3*0.D0/
C
C-----S(20,12) THRU S(12,12)
C
      DATA (CS(I,11),I=1,9) /5*0.D0,.1068D-19,-.150D-19,.933D-19,
                                    .718D-20/
C
C-----S(20,13) THRU S(13,13)
C
      DATA (CS(I,10),I=1,8) /5*0.D0,-.928D-21,0.D0,.282D-19/
C
C-----S(20,14) THRU S(14,14)
C
      DATA (CS(I,9),I=1,7) /5*0.D0,-.558D-22,-.4138D-22/
C
C-----S(20,15) THRU S(15,15)
C
      DATA (CS(I,8),I=1,6) /6*0.D0/
C
C-----S(20,16) THRU S(16,16)
C
      DATA (CS(I,7),I=1,5) /5*0.D0/
C
C-----S(20,17) THRU S(17,17)
C
      DATA (CS(I,6),I=1,4) /4*0.D0/
C
C-----S(28,18) THRU S(18,18)
C
      DATA (CS(I,5),I=1,3) /3*0.D0/
C
C-----S(20,19) THRU S(19,19)
C
      DATA (CS(I,4),I=1,2) /2*0.D0/
C
C-----S(20,20)
C
      DATA CS(1,3) /0.D0/
C
      END
@ ELT OUTPUT,1,670830, 47591
@ EOF @
      SUBROUTINE OUTPUT
      DOUBLE PRECISION TOUT,TX,TXD,EV,EVD,X,XD,XDD,XXDD,DIFF,CDEL,TT,T
     1OUT ,TOL1,TOL2,TC,BEGTIM,ENDTIM,BEGT2,DSQRT,TREQ
      DOUBLE PRECISION FX,FXD,FXDD,TI,TIME
      DOUBLE PRECISION DRAG,DN
      DOUBLE PRECISION RSAVE
      LOGICAL ISWT,SW
      INTEGER ORDER
      DIMENSION TX(3),TXD(3)
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     1                 CDEL,TT,T,TREQ,TOUT,K,N
      COMMON/INTERP/FX(60,3),FXD(60,3),FXDD(60,3),TI,K1,M,M1
      COMMON/LIMITS/TOL1,TOL2,TC,ISCT,ISWT(30),SW(20),ORDER,L1,L2,MODE
      COMMON/OPT/BEGTIM,ENDTIM,BEGT2
      COMMON/DDRAG/DRAG(3)
      COMMON/BRIEF/RSAVE(10)
  101 FORMAT(1H0,21HNORM OF ERROR VECTORS,8X2HE=D24.16,5X5HEDOT=D24.16)
```

```
107 FORMAT(1H0,15X1HT27X1HX30X1HY28X1HZ)
108 FORMAT(4(5X,D24.16))
109 FORMAT(14X5HR X V,15X,3(D24.16,5X))
110 FORMAT (1H0,14X,20HNORM OF DRAG VECTOR=D24.16)
    TOUT=T/TC
    IF(.NOT. ISWT(7)) GO TO 57
    IF(TOUT.LT.BEGTIM.OR.(TOUT.GT.ENDTIM.AND.TOUT.LT.BEGT2))RETURN
 57 WRITE(3,107)
    IF(ISWT(8)) GO TO 20
    WRITE(3,108) TREQ,(FX(M1,I),I=1,3)
    CON=DSQRT((FX(M1,2)*FXD(M1,3)-FX(M1,3)*FXD(M1,2))**2
   *          +(FX(M1,3)*FXD(M1,1)-FX(M1,1)*FXD(M1,3))**2
   *          +(FX(M1,1)*FXD(M1,2)-FX(M1,2)*FXD(M1,1))**2)
    WRITE(3,108) CON,(FXD(M1,I),I=1,3)
    IF(ISWT(12)) GO TO 26
    DN=0.D0
    DO 23 I=1,3
 23 DN=AMAX1(DABS(DRAG(I)),DN)
    IF(DN.LE.0.D0)  GO TO 24
    WRITE(3,110) DN
 24 CONTINUE
    RETURN
 26 TIME=T
    T=TI
    CALL TRANS(TX,TXD)
    T=TIME
    EV=0.D0
    EVD=0.D0
    DO 22 I=1,3
    EV=AMAX1(DABS(FX(M1,I)-TX(I)),EV)
 22 EVD=AMAX1(DABS(FXD(M1,I)-TXD(I)),EVD)
    RSAVE(3)=EV
    WRITE(3,101) EV,EVD
    RETURN
 20 CONTINUE
    WRITE(3,108)TOUT,(X(K,I),I=1,3)
    WRITE(3,109)(XD(K,I),I=1,3)
    CON=DSQRT((X(K,2)*XD(K,3)-X(K,3)*XD(K,2))**2+(X(K,3)*XD(K,1)-X(K
   1)*XD(K,3))**2+(X(K,1)*XD(K,2)-X(K,2)*XD(K,1))**2)
    WRITE(3,108)CON   ,XXDD
    IF(.NOT. ISWT(12))  RETURN
    CALL TRANS(TX,TXD)
 25 EV=0.D0
    EVD=0.D0
    DO 32 I=1,3
    EV=AMAX1(DABS(X(K,I)-TX(I)),EV)
 32 EVD=AMAX1(DABS(XD(K,I)-TXD(I)),EVD)
    RSAVE(3)=EV
    WRITE(3,101)EV,EVD
    RETURN
    END
@ ELT PCCOFF,1,670830, 47592
@ EOF @
    SUBROUTINE PCCOFF (SIGMA,GAMMA,SIGMAS,GAMMAS,M)
C      CALCULATES PREDICTOR CORRECTOR COEFFICIENT M GIVEN M-1 COEFFICIENT
    DOUBLE PRECISION SIGMA,GAMMA,SIGMAS,GAMMAS,S1,S2,S3,S4,H,X1,X2,FM,
   1FI
    DIMENSION SIGMA(63),GAMMA(63),SIGMAS(63),GAMMAS(63),H(63)
    IF(M-2)1,2,3
  1 SIGMA(M)=1.0D0
```

```
      SIGMAS(M)=1.0D0
      GAMMA(M)= 1.0D0
      GAMMAS(M)=1.0D0
      H(M)= 1.0D0
      GO TO 10
    2 SIGMA(M)=0.0D0
      SIGMAS(M)= -1.0D0
      GAMMA(M)=0.5D0
      GAMMAS(M)=-0.5D0
      H(M)= 1.5D0
      GO TO 10
    3 S1=0.0D0
      S2=0.0D0
      S3=0.0D0
      S4=0.0D0
      FM=M
      H(M)=H(M-1)+1.0D0/FM
      J=M-1
      DO 4 I=1,J
      FI=I
      X1=1.0D0/(FI+1.0D0)
      X2=2.0D0/(FI+2.0D0)
      K=M-I
      L=I+1
      S1=S1+X1*GAMMA(K)
      S2=S2+X1*GAMMAS(K)
      S3=S3+X2*H(L)*SIGMA(K)
    4 S4=S4+X2*H(L)*SIGMAS(K)
      GAMMA(M)=1.0D0-S1
      GAMMAS(M)=-S2
      SIGMA(M)=1.0D0-S3
      SIGMAS(M)=-S4
   10 RETURN
      END
@ ELT RESUME,1,670830, 47593
@ EOF @
      SUBROUTINE RESUME
      DOUBLE PRECISION TOL1,TOL2,TC
      DOUBLE PRECISION RSAVE,TEMP
      LOGICAL ISWT,SW
      DOUBLE PRECISION STEPD,TEMP3,TEMP4,OLDT,TM
      DOUBLE PRECISION X,XD,XDD,XXDD,DIFF,CDEL,TT,T,TREQ,TOUT
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     1              CDEL,TT,T,TREQ,TOUT,K,N
      COMMON/LIMITS/TOL1,TOL2,TC,ISCT,ISWT(30),SW(20),ORDER,L1,L2,MODE
      COMMON/TIMING/STEPD(90,3),TEMP3,TEMP4,OLDT,TM,XM,YM,ZM,XS,
     *              IENT,ITERS,INCOWL,ICH
      COMMON/TIMES/TAB,COWL,FORCS
      COMMON/BRIEF/RSAVE(10)
C     INSERT CODING FOR RESUME
      IF(.NOT. SW(13))  GO TO 1
      IF(MODE.EQ.3 .OR. MODE.EQ.4)  GO TO 58
      TEMP3=TEMP4
    1 J=1
   92 IF(DABS(TEMP3-STEPD(J,1)) .GT. 1.D-13)  GO TO 90
      STEPD(J,2)= STEPD(J,2) + DBLE(ZM)
      STEPD(J,3)= STEPD(J,3)+ TM
      IF(SW(13))  GO TO 58
      RETURN
   90 J=J+1
```

```
      IF(J.LE.IENT) GO TO 92
      IF(IENT.GT.89) GO TO 190
      IENT= IENT + 1
      STEPD(IENT,1)=TEMP3
      STEPD(IENT,2)=DBLE(ZM)
      STEPD(IENT,3)= TM
      IF(.NOT. SW(13))  RETURN
   58 ZM=YM-XS
      XS=YM
      TAB=TAB/60.D0
      COWL=COWL/60.D0
      FORCS=FORCS/60.D0
      ZM=ZM/60.
      WRITE(3,360) TAB,COWL,FORCS,ZM
  360 FORMAT (1H1, //////, 47X, 27HTIMING BREAKDOWN BY ROUTINE //,
     1 17X, 5HTABLE, 24X, 5HCSTEP, 24X, 4HFRCS, 25X,
     2 9HTOTAL RUN /(4(5X,E24.8)))
      WRITE(3,362) (RSAVE(I),I=1,3)
  362 FORMAT (1H0 // 32X, 13HINITIAL ORDER 10X, 12HINITIAL STEP 10X,
     * 11HFINAL ERROR / 33X, 3(D12.5,10X))
      AITER = FLOAT(ITERS)/FLOAT(INCOWL)
      WRITE (3,365) AITER
  365 FORMAT (1H0 //// 41X, 37HAVERAGE NUMBER OF ITERATIONS IN CSTEP//
     1  48X, E24.8)
      WRITE(3,366) INCOWL
  366 FORMAT (1H0//// 50X, 28HTOTAL NUMBER OF COWELL STEPS//,57X,I12)
      IF(MODE.EQ.1 .OR. MODE.EQ.2  .AND. IENT.GT.1)  GO TO 76
      STEPD(1,1)=CDEL/TC
      STEPD(1,2)=DBLE(ZM)
      STEPD(1,3)=T/TC
      WRITE(3,370) (STEPD(1,I),I=1,3)
      RETURN
   76 IT=IENT-1
      DO 80 I1=1,IENT
      STEPD(I1,1)=STEPD(I1,1)/TC
      STEPD(I1,2)=STEPD(I1,2)/60.D0
   80 STEPD(I1,3)=STEPD(I1,3)/TC
   78 DO 77 I=1,IT
      I2=I
      DO 77 LR=I2,IT
      IF(STEPD(LR+1,1).GT.STEPD(I,1))  GO TO 77
      DO 79 J=1,3
      TEMP= STEPD (LR+1,J)
      STEPD(LR+1,J)=STEPD(I,J)
   79 STEPD(I,J)=TEMP
   77 CONTINUE
      WRITE (3,370) ((STEPD(I,J),J=1,3),I=1,IENT)
  370 FORMAT (//////, 22X, 9HSTEP SIZE 27X, 8HRUN TIME, 21X,
     * 10HORBIT TIME (//, 20X, D24.16,5X,D24.16,5X,D24.16))
      RETURN
C     ERROR EXIT
  190 IF(.NOT. SW(13)) RETURN
      WRITE(3,290)
  290 FORMAT (49H0STEP CHANGES GREATER THAN 90. RESUME INCOMPLETE.)
      RETURN
      END
@ ELT RK,1,670830, 47596
@ EOF @
      SUBROUTINE RK
C  SHANKS R-K ROUTINE ORDER 8
```

```
      DOUBLE PRECISION X,XD,XDD,H,T,TIME,F,RK1,RK2,RK3,RK4,RK5,RK6,RK7,X
     11,XD1,RK8,RK9,RK10,XXDD,DIFF,CDEL,TT,TOL1,TOL2,TC,TREQ,TOUT
      DOUBLE PRECISION H1,H2
      DOUBLE PRECISION CSTEPT
      INTEGER ORDER
      LOGICAL ISWT,SW
      DIMENSION                   F(6),RK1(6),RK2(6),RK3(6),RK4(6),RK5(6
     1),RK6(6),RK7(6),RK8(6),RK9(6),RK10(6),X1(3),XD1(3)
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     1               CDEL,TT,T,TREQ,TOUT,K,N
      COMMON/RKT/CSTEPT
      COMMON/LIMITS/TOL1,TOL2,TC,ISCT,ISWT(30),SW(20),ORDER,L1,L2,MODE
      COMMON/RKST/H1,H2
      IF(SW(1)) GO TO 110
      H=-H2
      LL=3
      DO 73 I=1,3
      X(K,I)=X(K+1,I)
   73 XD(K,I)=XD(K+1,I)
   82 IF(CSTEPT.LE.(T+H)) GO TO 120
      GO TO 61
  110 LL=1
      H=H1
      IF(SW(12))H=H2
      DO 74 I=1,3
      X(K,I)=X(K-1,I)
   74 XD(K,I)=XD(K-1,I)
  130 IF(CSTEPT.GT.(T+H)) GO TO 120
   61 LL=2
      H=CSTEPT-T
  120 TIME=T
      DO 1 I=1,3
      X1(I)=X(K,I)
    1 XD1(I)=XD(K,I)
      DO 50 L=1,10
      DO 2 I=1,3
      F(I)=XD(K,I)
    2 F(I+3)=XXDD(I)
      GO TO (10,20,30,40,52,60,70,80,90,100),L
   10 DO 3 I=1,6
    3 RK1(I)=H*F(I)
      DO 4 I=1,3
      X(K,I)=X1(I)+RK1(I)*4.0D0/27.0D0
    4 XD(K,I)=XD1(I)+RK1(I+3)*4.0D0/27.0D0
      T=TIME+H*4.0D0/27.0D0
      GO TO 49
   20 DO 5 I=1,6
    5 RK2(I)=H*F(I)
      DO 6 I=1,3
      X(K,I)=X1(I)+(RK1(I)+3.0D0*RK2(I))/18.0D0
    6 XD(K,I)=XD1(I)+(RK1(I+3)+3.0D0*RK2(I+3))/18.0D0
      T=TIME+2.0D0*H/9.0D0
      GO TO 49
   30 DO 7 I=1,6
    7 RK3(I)=H*F(I)
      DO 8 I=1,3
      X(K,I)=X1(I)+(RK1(I)+3.0D0*RK3(I))/12.0D0
    8 XD(K,I)=XD1(I)+(RK1(I+3)+3.0D0*RK3(I+3))/12.0D0
      T=TIME+H/3.0D0
      GO TO 49
```

```
40 DO 9 I=1,6
 9 RK4(I)=H*F(I)
   DO 12 I=1,3
   X(K,I)=X1(I)+(RK1(I)+3.0D0*RK4(I))/8.0D0
12 XD(K,I)=XD1(I)+(RK1(I+3)+3.0D0*RK4(I+3))/8.0D0
   T=TIME+H/2.0D0
   GO TO 49
52 DO 13 I=1,6
13 RK5(I)=H*F(I)
   DO 14 I=1,3
   X(K,I)=X1(I)+(13.0D0*RK1(I)-27.0D0*RK3(I)+42.0D0*RK4(I)+8.0D0*RK5(
  1I))/54.0D0
14 XD(K,I)=XD1(I)+(13.0D0*RK1(I+3)-27.0D0*RK3(I+3)+42.0D0*RK4(I+3)+8.
  1D0*RK5(I+3))/54.0D0
   T=TIME+2.0D0*H/3.0D0
   GO TO 49
60 DO 15 I=1,6
15 RK6(I)=H*F(I)
   DO 16 I=1,3
   X(K,I)=X1(I)+(389.0D0*RK1(I)-54.0D0*RK3(I)+966.D0*RK4(I)-824.D0*RK
  15(I)+243.0D0*RK6(I))/4320.0D0
16 XD(K,I)=XD1(I)+(389.0D0*RK1(I+3)-54.D0*RK3(I+3)+966.D0*RK4(I+3)-82
  14.0D0*RK5(I+3)+243.0D0*RK6(I+3))/4320.0D0
   T=TIME+H/6.0D0
   GO TO 49
70 DO 17 I=1,6
17 RK7(I)=H*F(I)
   DO 21 I=1,3
   X(K,I)=X1(I)+(-231.0D0*RK1(I)+81.0D0*RK3(I)-1164.D0*RK4(I)+656.D0*
  1RK5(I)-122.0D0*RK6(I)+800.0D0*RK7(I))/20.0D0
21 XD(K,I)=XD1(I)+(-231.0D0*RK1(I+3)+81.D0*RK3(I+3)-1164.D0*RK4(I+3)+
  1656.0D0*RK5(I+3)-122.0D0*RK6(I+3)+800.0D0*RK7(I+3))/20.0D0
   T=TIME+H
   GO TO 49
80 DO 22 I=1,6
22 RK8(I)=H*F(I)
   DO 23 I=1,3
   X(K,I)=X1(I)+(-127.0D0*RK1(I)+18.0D0*RK3(I)-678.D0*RK4(I)+456.D0*R
  1K5(I)-9.0D0*RK6(I)+576.0D0*RK7(I)+4.0D0*RK8(I))/288.0D0
23 XD(K,I)=XD1(I)+(-127.0D0*RK1(I+3)+18.D0*RK3(I+3)-678.D0*RK4(I+3)+4
  156.0D0*RK5(I+3)-9.0D0*RK6(I+3)+576.0D0*RK7(I+3)+4.0D0*RK8(I+3))/28
  28.0D0
   T=TIME+5.0D0*H/6.0D0
   GO TO 49
90 DO 24 I=1,6
24 RK9(I)=H*F(I)
   DO 25 I=1,3
   X(K,I)=X1(I)+(1481.0D0*RK1(I)-81.0D0*RK3(I)+7104.D0*RK4(I)-3376.D0
  1*RK5(I)+72.0D0*RK6(I)-5040.0D0*RK7(I)-60.0D0*RK8(I)+720.0D0*RK9(I)
  2)/820.0D0
25 XD(K,I)=XD1(I)+(1481.0D0*RK1(I+3)-81.D0*RK3(I+3)+7104.D0*RK4(I+3)-
  13376.0D0*RK5(I+3)+72.0D0*RK6(I+3)-5040.0D0*RK7(I+3)-60.0D0*RK8(I+3
  2)+720.0D0*RK9(I+3))/820.0D0
   T=TIME+H
   GO TO 49
100 DO 26 I=1,6
26 RK10(I)=H*F(I)
   GO TO 51
49 CALL FRCS
50 CONTINUE
```

```
   51 DO 18 I=1,3
      X(K,I)=X1(I) +(41.0D0*(RK1(I)+RK10(I))+27.0D0*(RK4(I)+RK6(I))+272.
     10D0*RK5(I)+216.0D0*(RK7(I)+RK9(I)))/840.0D0
   18 XD(K,I)=XD1(I) +(41.0D0*(RK1(I+3)+RK10(I+3))+27.0D0*(RK4(I+3)+RK6(
     1I+3))+272.0D0*RK5(I+3)+216.0D0*(RK7(I+3)+RK9(I+3)))/840.0D0
      CALL FRCS
      GO TO (130,81,82),LL
   81 RETURN
      END
@ ELT RYMDI,1,670830, 47596
@ EOF @
C NAME          SUBROUTINE RYMDI
C
C LANGUAGE      FORTRAN IV
C
C MACHINE       UNIVAC 1107/1108
C
C
C PURPOSE       TO SEPARATE PACKED SIX-DIGIT DECIMAL DATES INTO
C               TWO-DIGIT YEAR, MONTH, AND DAY
C
C
C CALLING SEQUENCE       CALL RYMDI(YMD, Y, M, D)
C
C SYMBOL        TYPE        DESCRIPTION
C
C YMD(1)        I           INPUT - DATE TO BE SEPARATED
C
C Y(1)          I           OUTPUT - TWO-DIGIT YEAR
C
C M(1)          I           OUTPUT - TWO-DIGIT MONTH
C
C D(1)          I           OUTPUT - TWO-DIGIT DAY
C
C
C ROUTINES REQUIRED         NONE
C
C TAPES REQUIRED            NONE
C
C CARDS REQUIRED            NONE
C
C
        SUBROUTINE RYMDI (YMD,Y,M,D)
C
C     PURPOSE - TO UNPACK YYMMDD TO YY - MM - DD
C     INPUT   YMD = YEAR MONTH DAY PACKED
C     OUTPUT    Y = YEAR
C               M = MONTH
C               D = DAY
C
        INTEGER YMD,Y,M,D
C
        Y=YMD/10000
        I=YMD/100
        M=I-Y*100
        D=YMD-I*100
        RETURN
        END
@ ELT SHADOW,1,671009, 48541
@ EOF @
```

```
      SUBROUTINE SHADOW(IND)
      DOUBLE PRECISION X,XD,XDD,XXDD,DIFF,CDEL,TT,T,TREQ,TOUT
      DOUBLE PRECISION EMASS,XYZ,COSPSI,PROJ,REARTH,DSQRT
      DOUBLE PRECISION PSUN,CSUBR,SIGMA,F
      DOUBLE PRECISION GM,AE,R,RSQ,RQ,THETG,TC1
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     *              CDEL,TT,T,TREQ,TOUT,K,N
      COMMON/CONST3/EMASS(2),XYZ(4),LS
      COMMON/CONST4/PSUN,CSUBR,SIGMA,F
      COMMON/CONST2/GM,AE,R,RSQ,RQ,THETG,TC1
C    COMPUTE DOT PRODUCT  - GET ANGLE BETWEEN SUN AND SATELLITE
      COSPSI=(X(K,1)/R)*XYZ(1)+(X(K,2)/R)*XYZ(2)+(X(K,3)/R)*XYZ(3)
C    DETERMINE WHETHER SATELLITE IS IN SUNLIGHT
      IF(COSPSI .LT. 0.D0)  GO TO 10
      IND=1
      RETURN
   10 PROJ=DSQRT((X(K,2)*XYZ(3)-X(K,3)*XYZ(2))**2
     *          +(X(K,3)*XYZ(1)-X(K,1)*XYZ(3))**2
     *          +(X(K,1)*XYZ(2)-X(K,2)*XYZ(1))**2)
      REARTH=1.D0-F*((X(K,3)+XYZ(3)*DSQRT(RSQ-1.D0))**2)
      IND=1
      IF(PROJ .LT. REARTH)  IND=0
      RETURN
      END
@ ELT SLGRAV,1,671013, 34112
@ EOF @
      SUBROUTINE SLGRAV
      DOUBLE PRECISION EMASS,XYZ,GM,AE,R,RSQ,THETG,TC1,RSAT,RS
      DOUBLE PRECISION X,XD,XDD,XXDD,DIFF,CDEL,TT,T,TREQ,TOUT,DSQRT
      DOUBLE PRECISION DXX(3)
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     *              CDEL,TT,T,TREQ,TOUT,K,N
      COMMON/CONST2/GM,AE,R,RSQ,RQ,THETG,TC1
      COMMON/CONST3/EMASS(2),XYZ(4),LS
      DO 10 I=1,3
   10 XYZ(I)=XYZ(I)*XYZ(4)
      RSAT=((X(K,1)-XYZ(1))**2+(X(K,2)-XYZ(2))**2+(X(K,3)-XYZ(3))**2)
      RSAT=(DSQRT(RSAT))*RSAT
      RS=XYZ(4)**3
      DXX(1)=GM*EMASS(LS)*(((X(K,1)-XYZ(1))/RSAT)+(XYZ(1)/RS))
      DXX(2)=GM*EMASS(LS)*(((X(K,2)-XYZ(2))/RSAT)+(XYZ(2)/RS))
      DXX(3)=GM*EMASS(LS)*(((X(K,3)-XYZ(3))/RSAT)+(XYZ(3)/RS))
      DO 3 I=1,3
    3 XXDD(I)=XXDD(I)-DXX(I)
      RETURN
      END
@ ELT SOLRAD,1,671009, 48542
@ EOF @
      SUBROUTINE SOLRAD
      DOUBLE PRECISION X,XD,XDD,XXDD,DIFF,CDEL,TT,T,TREQ,TOUT
      DOUBLE PRECISION EMASS,XYZ
      DOUBLE PRECISION PSUN,CSUBR,SIGMA,F,RSS,DSQRT
      DOUBLE PRECISION SOL(3)
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     *              CDEL,TT,T,TREQ,TOUT,K,N
      COMMON/CONST3/EMASS(2),XYZ(4),LS
      COMMON/CONST4/PSUN,CSUBR,SIGMA,F
      DO 1 I=1,3
    1 XYZ(I)=XYZ(I)*XYZ(4)
      RSS=DSQRT((XYZ(1)-X(K,1))**2+ (XYZ(2)-X(K,2))**2
```

```
     *         +(XYZ(3)-X(K,3))**2)
      DO 2 I=1,3
      SOL(I)=SIGMA*((XYZ(I)-X(K,I))/RSS)
    2 XXDD(I)=XXDD(I)-SOL(I)
      RETURN
      END
@ ELT SUMS,1,670830, 47597
@ EOF @
      SUBROUTINE SUMS
      DOUBLE PRECISION X,XD,XDD,XXDD,DIFF,CDEL,TT,T,TREQ,TOUT
      DOUBLE PRECISION S1,S2,PX,PXD,CX,CXD,CTOL,SAVE
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     1              CDEL,TT,T,TREQ,TOUT,K,N
      COMMON/COWS/S1(3),S2(3),PX(63),PXD(63),CX(63),CXD(63),CTOL,SAVE(60
     1          ,3),ITER
      DO 1 I=1,3
      S1(I)=XD(K,I)*CDEL + CXD(2)*XDD(K,I)
      S2(I)=X(K,I)- CX(3) * XDD(K,I)
      DO 1 J=1,N
      S1(I)=S1(I) - CXD(J+2) * DIFF(J,I)
    1 S2(I)=S2(I) -CX(J+3) * DIFF(J,I)
      DO 2 I=1,3
      S1(I)= XDD(K,I) + S1(I)
    2 S2(I)= S1(I) + S2(I)
      RETURN
      END
@ ELT TABLE,1,670830, 47598
@ EOF @
      SUBROUTINE TABLE
      DOUBLE PRECISION X,XD,XDD,XXDD,TOUT,T,TIME,TOUT1,TT,DIFF,TOL1,TOL2
     1,CSTEPT,FJ,TC,CDEL,TREQ
      DOUBLE PRECISION GM,AE,R,RSQ,RQ,THETG,TC1
      DOUBLE PRECISION CS,DAY1,CENTER,DSTART
      INTEGER ORDER
      LOGICAL ISWT,SW
      COMMON/RKT/CSTEPT
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     1              CDEL,TT,T,TREQ,TOUT,K,N
      COMMON/LIMITS/TOL1,TOL2,TC,ISCT,ISWT(30),SW(20),ORDER,L1,L2,MODE
      DOUBLE PRECISION THETG0,THDOT1,THDOT2,DMIN,ETIME
      COMMON/CONST1/THETG0(10),THDOT1,THDOT2,DMIN,ETIME,IYBEG
      COMMON/CONST2/GM,AE,R,RSQ,RQ,THETG,TC1
      COMMON/COFIT/CS(5,9),DAY1,CENTER,DSTART
  102 FORMAT(1H1,42X33HRUNGE-KUTTA NUMERICAL INTEGRATION/1H0,48X20HINI
     1AL INPUT VALUES/1H020X1HX36X1HY36X1HZ)
  103 FORMAT(3(10X,D24.16))
  104 FORMAT(5X5HTIME=D24.16,2X8HDELTA T=D24.16,14X6HORDER=I2)
  105 FORMAT(1H0,42X33HRUNGE-KUTTA NUMERICAL INTEGRATION/1H0,45X29HPREDI
     1CTED EXTRAPOLATED VALUES/1H020X1HX36X1HY36X1HZ)
  106 FORMAT(1H0,42X33HRUNGE-KUTTA NUMERICAL INTEGRATION/1H0,45X29HCORRE
     1CTED EXTRAPOLATED VALUES/1H020X1HX36X1HY36X1HZ)
  110 FORMAT(1H032HINITIAL COWELL ORDER CHANGED TO I3)
      IF(SW(12)) GO TO 40
      WRITE(3,102)
    1 WRITE(3,103)(X(1,I),I=1,3),(XD(1,I),I=1,3),(XXDD(I),I=1,3)
   28 TOUT1=CDEL/TC
      TOUT=T/TC
      WRITE(3,104)TOUT,TOUT1,ORDER
      SW(1)=.TRUE.
      SW(2)=.TRUE.
```

```
      TIME=T
   24 KK=K
   26 KK1=N
   27 DO 10 J=KK,KK1
      FJ=J
      CSTEPT=TIME+FJ*CDEL
      IF(CSTEPT*TC1+DSTART .LE. DAY1)  GO TO 3
      CALL EPHQAN
    3 K=J+1
      CALL RK
      IF(SW(12)) GO TO 10
      IF(ISWT(8))  GO TO 11
      ISWT(8)=.TRUE.
      CALL OUTPUT
      ISWT(8)=.FALSE.
      GO TO 10
   11 CALL OUTPUT
   10 CONTINUE
   25 DO 5 J=1,KK1
      CALL CKDIFF(J)
    5 CONTINUE
      IF(MODE.EQ.4) GO TO 31
    7 SW(3)=.TRUE.
    8 SW(4)=.FALSE.
      SW(5)=.FALSE.
      SW(7)=.FALSE.
      SW(9)=.FALSE.
      CALL TEST
      IF(SW(9)) GO TO 31
   30 IF(SW(4)) GO TO 29
      IF(SW(5)) GO TO 16
      IF (SW(7)) GO TO 2
      GO TO 6
    2 SW(2)=.FALSE.
      GO TO 24
   16 SW(2)=.FALSE.
      GO TO 8
   29 WRITE(3,111)
  111 FORMAT(1H068HINITIAL TABLE FAILED TOLERANCES--PROCEDURE RESTART WI
     1TH NEW STEPSIZE)
      T=TIME
      K=1
      CALL FRCS
      GO TO 28
    6 IF(SW(2))GO TO 31
      WRITE(3,110)ORDER
   31 SW(3)=.FALSE.
      RETURN
   40 IF(.NOT. ISWT(15))  GO TO 41
      WRITE(3,105)
      GO TO 1
   41 IF(SW(14)) WRITE(3,105)
      IF(.NOT.SW(14)) WRITE(3,106)
      GO TO 1
      END
@ ELT TABLEB,1,671013, 34119
@ EOF @
      SUBROUTINE TABLEB
      DOUBLE PRECISION X,XD,XDD,TC,CDEL,XXDD,TOUT,TIME,TOUT1,FJ,TT,
     1T,DIFF,TOL1,TOL2,CSTEPT,FX,FXD,TI,FI1,FACT,STEPD,TEMP3
```

```
     2,TEMP4,OLDT,TM,FXDD,TREQ
      DOUBLE PRECISION RSAVE,TP,TPP,TMP
      LOGICAL ISWT,SW
      INTEGER ORDER
      COMMON/TIMING/STEPD(90,3),TEMP3,TEMP4,OLDT,TM,XM,YM,ZM,XS,
     *              IENT,ITERS,INCOWL,ICH
      COMMON/RKT/CSTEPT
      COMMON/LIMITS/TOL1,TOL2,TC,ISCT,ISWT(30),SW(20),ORDER,L1,L2,MODE
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     1              CDEL,TT,T,TREQ,TOUT,K,N
      COMMON/INTERP/FX(60,3),FXD(60,3),FXDD(60,3),TI,K1,M,M1
      COMMON/TIMES/TAB, COWL, FORCS
      COMMON/BRIEF/RSAVE(10)
      EQUIVALENCE (RSAVE(4),TP)
  120 FORMAT(1H0,10X,22HSTEP-SIZE DOUBLING TO D19.12,6H AT T=,D19.12,
     1 2X,6HORDER=,I2)
  122 FORMAT(1H0,10X,22HSTEP-SIZE HALVING  TO D19.12,6H AT T=,D19.12,
     1 2X,6HORDER=,I2)
  126 FORMAT (1H0,40X,33HPOINTS COMPUTED USING RUNGE-KUTTA)
      CALL CLOCK (XE)
      TOUT=T/TC
      TOUT1=CDEL/TC
C     IF OPTIMIZATION OF STEP USE INTERPOLATION
      IF(ISWT(16))  GO TO 40
C     IF HALVING STEP USE INTERPOLATION
      IF (.NOT. SW(6)) GO TO 26
C     IF INSUFFICIENT POINTS USE RUNGE KUTTA
      IF (ISCT .LT. 2*(N+1)) GO TO 40
      WRITE(3,120) TOUT1,TOUT,ORDER
      DO 70 I=1,3
      XDD(K,I)=XDD(K,I)*4.D0
      DO 70 J=1,N
      N1=K-J
C     DOUBLE STEP BY SELECTING EVERY OTHER POINT
      N2=K-2*J
      X(N1,I)=X(N2,I)
      XD(N1,I)=XD(N2,I)
   70 XDD(N1,I)=XDD(N2,I) * 4.D0
      GO TO 25
C     M ORDER OF HERMITE INTERPOLATION -DEPENDENT ON COWELL ORDER AND STEP
   26 M=(ORDER + 1)/2
      M=MAX0(5,M)
      IF(TEMP3 .GE. 0.5D0)  M=MAX0(8,M)
      M=M+2
      IF(ISCT .LT. M) GO TO 40
      KK=K
      WRITE(3,122) TOUT1,TOUT,ORDER
      K1=K
      M2=M-2
      TEMP3=(-1.D0)*TEMP3
      DO 37 J=2,M2
      FJ=J
C     HALVE STEP BY INTERPOLATING FOR MIDPOINTS
      FI1=2.D0*(FJ-1.D0) + 1.D0
      TI = T-FI1*CDEL
      M1=M-J
      CALL HEMINT
   37 CONTINUE
      TEMP3=(-1.D0)*TEMP3
      M2= M-2
```

```
      DO 30 J1=1,M
      J2=K-J1+1
      XDD(J2,1)=XDD(J2,1)/4.D0
      XDD(J2,2)=XDD(J2,2)/4.D0
   30 XDD(J2,3)=XDD(J2,3)/4.D0
      DO 35 J=2,M2
C     INDEX TO MOVE AN ARRAY POINT BACKWARD TO A NEW POSITION
      N1=KK-(M-J)
C     INDEX TO INSERT INTERPOLATED POINT INTO INTERMEDIATE SLOTS IN ARRAY
      N2=KK- 2*(M-J)+ 1
      K=N2 + 1
      DO 32 I=1,3
      X(N2,I)=X(N1,I)
      XD(N2,I)=XD(N1,I)
      XDD(N2,I)=XDD(N1,I)
      X(K,I)=FX(J,I)
   32 XD(K,I)=FXD(J,I)
      CALL FRCS
   35 CONTINUE
C     REJECT LAST COMPUTED POINT
      K=KK-1
      T=T-TEMP3
      GO TO 25
   40 TPP=T-2.D0*TEMP3-DBLE(N+1)*CDEL
C     DETERMINE IF THERE ARE ENOUGH POINTS AT THE LAST STEPSIZE TO PRODUCE
C     N<1 POINTS AT THE NEW STEPSIZE
      IF(TPP.LT.TP)  GO TO 50
      ASSIGN 60 TO L
      IF(SW(6))  GO TO 100
C     IF DECREASING STEP - REJECT LAST COMPUTED POINT
      K=K-1
      T=T-TEMP3
      TPP=TPP-TEMP3
      TT=TEMP3**2
      CALL FRCS
      TT=CDEL**2
      TOUT=T/TC
      GO TO 100
   60 M1=0
C     HERMITE INTERPOLATION ORDER
      M=MAX0(4,ORDER-1)
      IF(M.GT.ISCT)  GO TO 50
C     UPPER INDEX OF ARRAY POINTS TO BE INTERPOLATED [K  TO K1P]
      K1P=K-ISCT
C     ADJUST TIME SO THAT INTERPOLATION WILL NOT OCCUR NEAR THE END POINT
C     OF THE ARRAY. THESE ARE NOT AS ACCURATELY INTERPOLATED.
      TIME=T-TEMP3
      K1=K
C     TIME OF NEXT TO LAST POINT OF PARTIAL ARRAY SATISFYING HERMITE ORDER
C     WITHIN WHICH INTERPOLATION MUST OCCUR
      TMP=T-DBLE(M-2)*TEMP3
C     IF TIME OF NEXT TO LAST POINT OF PARTIAL ARRAY OVERLAPS POINTS PRODUCED
C     AT ANOTHER STEPSIZE - SET IT TO WITHIN TPP- TIME OF STEP PRIOR TO CHANGE
      IF(TMP.LT.TPP)  TMP=TPP+TEMP3
      KN=N+1
      TEMP3=(-1.D0)*TEMP3
   62 M1=M1+1
      FJ=M1
C     INTERPOLATION TIME
      TI=TIME-FJ*CDEL
```

```
      IF(TI.LT.TMP)  GO TO 65
      CALL HEMINT
   64 GO TO 62
C     UPDATE TIMES FOR NEXT PARTIAL ARRAY
   65 K1=K1-(M-3)
      T=T-DBLE(M-3)*DABS(TEMP3)
      TMP=TMP-DBLE(M-3)*DABS(TEMP3)
      IF(K1-M .GE. K1P)  GO TO 67
C     IF TIME OF LAST POINT OF PARTIAL ARRAY OVERLAPS STEP CHANGE - ADJUST TIME
C     AND INDEX TO POINT BEFOR THE CHANGE
      T=T+DBLE(M-K1+K1P-1)*DABS(TEMP3)
      TMP=TMP+DBLE(M-K1+K1P-1)*DABS(TEMP3)
      K1=K1P+M-1
   67 M1=M1-1
      IF(TMP.LT.TPP)  TMP=TPP+DABS(TEMP3)-.5D-13
C     IF N<1 POINTS HAVE NOT BEEN PRODUCED CONTINUE INTERPOLATING
      IF(M1.LT.N+1)  GO TO 62
      TEMP3=(-1.D0)*TEMP3
C     DROP LAST POINT SINCE INTERPOLATION WAS FROM NEXT TO LAST POINT
      KK=K-1
      T=TIME
      DO 68 I=1,N
      K=KK-I
      DO 66J=1,3
C     STORE INTERPOLATED VALUES
      X(K,J )=FX(I,J )
   66 XD(K,J)=FXD(I,J)
      T=T-CDEL
      CALL FRCS
   68 CONTINUE
      T=TIME
      K=KK
      CALL FRCS
      GO TO 25
   50 IF(.NOT.ISWT(16))  GO TO 52
      ASSIGN 45 TO L
      IF(SW(6))  GO TO 100
      ASSIGN 41 TO L
      GO TO 100
   52 IF(SW(6))  GO TO 43
      WRITE(3,122) TOUT1,TOUT,ORDER
   41 CONTINUE
C     COMPUTE BACKPOINTS USING RUNGE KUTTA
C
      K=K-1
      T=T-TEMP3
      GO TO 45
   43 FACT=4.D0
      WRITE(3,120) TOUT1,TOUT,ORDER
      XDD(K,1)= XDD(K,1)*FACT
      XDD(K,2)= XDD(K,2)*FACT
      XDD(K,3)= XDD(K,3)*FACT
   45 IF(SW(6).AND..NOT.ISWT(16))  GO TO 46
      CALL FRCS
   46 SW(1)=.FALSE.
      WRITE(3,126)
      TIME=T
      KK=K
      DO 10 J=1,N
      FJ=J
```

```
      CSTEPT=TIME-FJ*CDEL
      K=KK-J
      CALL RK
   10 CONTINUE
      K=KK
      T=TIME
      SW(6)=.TRUE.
C
C     COMPUTE NEW DIFFERENCES AND NEW SUMS
C
   25 DO 42 J=1,N
      CALL CKDIFF(J)
   42 CONTINUE
      CALL SUMS
      ISCT=N+1
      TP=T-FLOAT(N)*CDEL
      CALL CLOCK (XR)
      TAB = TAB + XR - XE
      RETURN
C     THIS SECTION FORMATS A MESSAGE TO INDICATE THE TYPE OF STEP CHANGE
  100 IF(  SW(6))  GO TO 101
      WRITE(3,121) TOUT1,TOUT,ORDER
  121 FORMAT (1H0,10X,21HSTEP SIZE DECREASE TO D19.12, 6H AT T= D19.12
     * 2X, 6HORDER=, I2)
      GO TO L (41,60)
  101 WRITE (3,123) TOUT1,TOUT,ORDER
  123 FORMAT (1H0,10X,21HSTEP SIZE INCREASE TO D19.12, 6H AT T= D19.12
     * 2X, 6HORDER=, I2)
      GO TO L (45,60)
      END
@ ELT TEST,1,671013, 34117
@ EOF @
      SUBROUTINE TEST
      DOUBLE PRECISION X,XD,XDD,XXDD,DIFF,CDEL,TT,T,TREQ
      DOUBLE PRECISION STEPD,TEMP3,TEMP4,OLDT,TM
      DOUBLE PRECISION CSAVE,TOUT,SUM,SUM1,FN1,OPTST
      DOUBLE PRECISION S1,S2,PX,PXD,CX,CXD,CTOL,SAVE
      DOUBLE PRECISION TOL1,TOL2,TOL3,TOL4,TC
      LOGICAL ISWT,SW,TEMP0,TEMP1,TEMP2
      INTEGER ORDER
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     1               CDEL,TT,T,TREQ,TOUT,K,N
      COMMON/LIMITS/TOL1,TOL2,TC,ISCT,ISWT(30),SW(20),ORDER,L1,L2,MODE,
      COMMON/TIMING/STEPD(90,3),TEMP3,TEMP4,OLDT,TM,XM,YM,ZM,XS,
     *               IENT,ITERS,INCOWL,ICH
      COMMON/COFFS/CSAVE(2)
      COMMON/ODEL/NL1,NL2
      COMMON/OPTIM/TOL3,TOL4
      COMMON/COWS/S1(3),S2(3),PX(63),PXD(63),CX(63),CXD(63),CTOL,SAVE(60
     1              ,3),ITER
  104 FORMAT(1H064HESTIMATE OF TRUNCATION ERROR INDICATES A COWELL ORDER
     1 CHANGE TO ,I3, 9H   TIME= , D19.12)
  107 FORMAT (1H0,64H TRUNCATION ERROR WILL ALLOW AN OPTIMIZATION OF ORD
     *ER CHANGE TO ,I3,9H    TIME=,D19.12)
  315 FORMAT (1H0,47X,31HORDER FAILED L1 - INCREASE STEP)
  355 FORMAT (1H0,47X,31HORDER FAILED L2 - DECREASE STEP)
      SW(18)=.FALSE.
C     DIFFERENCE TO BE TESTED
      N1=N-1
      FN1=N1
```

```
      TOUT=T/TC
      IF(SW(3))  GO TO 5
C     RESTORE POSITION COEFFICIENTS - THE ORDER MAY BE CHANGED
      PX(N+3)=CSAVE(1)
      CX(N+3)=CSAVE(2)
    5 SUM=0.D0
      DO 1 I=1,3
    1 SUM=SUM + DABS(DIFF(N1,I))
C     IF 1ST STEP OPTIMIZATION HAS NOT OCCURRED -- DO NOT ALLOW ORDER TO VARY.
      IF(((ISWT(16)).AND. SW(20)))  GO TO 2
C     AFTER MODE 1 STEP CHANGE -- ADJUST ORDER
C     SW[15] WILL THEN BE SET TO FALSE UNTIL ANOTHER STEP CHANGE OCCURS
C     IF ORDER OPTIMIZATION -- ORDER OPTIMIZATION CAN OCCUR IF LOCAL ERROR
C     SATISFIES TOLLS.
      IF(MODE.EQ.1.AND.SW(15))  GO TO 6
      IF(.NOT. ISWT(17))  GO TO 2
    6 IF(TOL1-SUM) 2,2,7
    7 IF(TOL2-SUM) 8,8,2
    8 IF(SW(3))  GO TO 2
      SW(15)=.FALSE.
      DO 101 J=1,N1
      N2=N-J
      SUM1=0.D0
      DO 102 I=1,3
  102 SUM1=SUM1+DABS(DIFF(N2,I))
      NN=N2+2
      IF(SUM1.GT. TOL4)  GO TO 103
  101 CONTINUE
C     SMALLEST ORDER -- TO SATISFY THE SPECIFIED TOLERANCE --SIGMA
  103 LORD=NN+2
      N5=LORD-3
      IF(LORD.LT.NL1.OR.LORD.GT.NL2)  GO TO 2
      IF(LORD .GE. ORDER)  GO TO 2
      IF((DABS(DIFF(N5,1))+DABS(DIFF(N5,2))+DABS(DIFF(N5,3))) .GT. TOL2)
     *  GO TO 2
C     COMPUTE APPROXIMATE LOCAL ERROR FOR NEW ORDER
      SUM=0.D0
      DO 4 I=1,3
    4 SUM=SUM+DABS(DIFF(N5,I))
      ORDER=LORD
      N=ORDER-2
      N1=N-1
      FN1=N1
      IF(MODE .NE. 1)  GO TO 9
C     MODE 1 - CHECK L1 LESS THAN ORDER LESS THAN L2
      IF(ORDER.GE.L1)  GO TO 18
      WRITE(3,107) LORD,TOUT
      WRITE(3,315)
      GO TO 10
   18 IF(ORDER.LE.L2)  GO TO 9
      WRITE(3,107) LORD,TOUT
      WRITE(3,355)
      GO TO 50
    9 WRITE(3,107) LORD,TOUT
C
C
C
    2 CONTINUE
      IF (SUM .GE. TOL2) GO TO 3
      GO TO (10,20,30),MODE
```

```
C      ENTRY MODE 1 - DOUBLE STEP--INCREASE ORDER
   10 IF(ORDER .GT. L1)  GO TO 30
      ORDER= L1 + (L2-L1)/2
      N= ORDER - 2
C      ENTRY MODE 2 - DOUBLE STEP
   20 SW(6)=.TRUE.
      OPTST=(TOL3/SUM*CDEL**FN1)**(1.D0/FN1)
      TEMP0=TEMP1
      TEMP1=TEMP2
      TEMP2=SW(6)
      ISCT1=ISCT2
      ISCT2=ISCT3
      ISCT3=ISCT
      IF(ICH .LT. 3)  GO TO 24
C     TEST FOR INCREASE-DECREASE LOOP.
      IF((ISCT1+ISCT2+ISCT3) .GT. 10)  GO TO 24
      IF((TEMP2.AND..NOT.TEMP1.AND.TEMP0).OR.(.NOT.TEMP2.AND.TEMP1.AND.
     * .NOT.TEMP0)) GO TO 110
   24 TEMP3=TEMP4
      IF(ICH.LT. 3)  ICH=ICH+1
      CDEL = CDEL * 2.0D0
      TT=CDEL**2
      TEMP4=CDEL
      IF(SW(3))  GO TO 21
      IF(.NOT. ISWT(16))  GO TO 25
C     STEP OPTIMIZATION -- USE COMPUTED STEP SIZE
      CDEL=OPTST
      TEMP4=CDEL
      NSAVE=N
C     SET INDEX TO PRODUCE SUFFICIENT POINTS SO THAT ORDER CAN INCREASE IN MODE 1
      N=L2-2
      IF(.NOT. SW(20))  GO TO 25
      SW(20)=.FALSE.
      N=NSAVE
   25 TT= CDEL**2
      CALL CLOCK(YM)
      ZM=YM-XM
C     COMPUTE REQUIRED BACKPOINTS
      CALL TABLEB
      SW(15)=.TRUE.
      IF(ISWT(16)) N=NSAVE
      CALL CLOCK (XM)
      GO TO 40
C      ENTRY MODE 3 - DECREASE ORDER
   30 ORDER=ORDER-1
      SW(18)=.TRUE.
      IF(ORDER.LT.NL1) GO TO 130
      N=ORDER - 2
      IF(SW(3)) GO TO 22
      WRITE (3,104) ORDER,TOUT
      GO TO 100
C
    3 IF (SUM .LE. TOL1) GO TO 100
      GO TO (50,60,70),MODE
C      ENTRY MODE 1 - HALVING STEP--DECREASE ORDER
   50 IF(ORDER .LT. L2)  GO TO 70
      ORDER=L1+(L2-L1)/2
      N=ORDER-2
C      ENTRY MODE 2 - HALVE STEP
   60 SW(6)=.FALSE.
```

```
      OPTST=(TOL3/SUM*CDEL**FN1)**(1.D0/FN1)
      TEMP0=TEMP1
      TEMP1=TEMP2
      TEMP2=SW(6)
      ISCT1=ISCT2
      ISCT2=ISCT3
      ISCT3=ISCT
      IF(ICH .LT. 3)  GO TO 61
      IF((ISCT1+ISCT2+ISCT3) .GT. 10)  GO TO 61
      IF((TEMP2.AND..NOT.TEMP1.AND.TEMP0).OR.(.NOT.TEMP2.AND.TEMP1.AND.
     * .NOT.TEMP0)) GO TO 110
   61 TEMP3=TEMP4
      IF(ICH.LT. 3)  ICH=ICH+1
      CDEL= CDEL/2.0D0
      TT=CDEL**2
      TEMP4=CDEL
      IF(SW(3))  GO TO 21
      IF(.NOT. ISWT(16))  GO TO 65
      CDEL=OPTST
      TEMP4=CDEL
      NSAVE=N
      N=L2-2
   65 TT= CDEL**2
      CALL CLOCK(YM)
      ZM=YM-XM
   66 CALL TABLEB
      SW(15)=.TRUE.
   67 IF(ISWT(16)) N=NSAVE
      CALL CLOCK (XM)
      K=K+1
      SW(8) = .TRUE.
      GO TO 40
C     ENTRY MODE 3 - INCREASE ORDER
   70 ORDER=ORDER+1
      SW(18)=.TRUE.
      IF(ORDER.GT.NL2) GO TO 130
      N= ORDER-2
      IF(SW(3)) GO TO 23
      IF(ISCT.LT. ORDER-1)  GO TO 170
      WRITE (3,104) ORDER,TOUT
      K=K-1
      T=T-CDEL
      DO 72 I=1,N1
      DO 72 J=1,3
   72 DIFF(I,J)= SAVE(I,J)
      CALL CKDIFF (N)
      CALL SUMS
      K=K+1
      SW(8) = .TRUE.
      GO TO 100
C     TERMINATION
   40 TM=T-OLDT
      OLDT=T
      CALL RESUME
      SW(18)=.TRUE.
C
C
  100 IF(SW(3))  RETURN
      CSAVE(1)=PX(N+3)
      CSAVE(2)=CX(N+3)
```

```
      PX(N+3)=0.D0
      CX(N+3)=0.D0
C     IF A STEP OR ORDER CHANGE HAS OCCURRED -- RETURN
      IF(SW(18))  RETURN
C
C OPTIMIZE STEP FOR FIRST COWELL POINT
      IF(SW(20))  GO TO 27
      RETURN
   21 SW(4)=.TRUE.
      ISCT3=0
      ICH=0
      TEMP2=.FALSE.
      TEMP3=0.D0
      RETURN
   22 IF(ORDER .LE. L1)  GO TO 10
      SW(5)=.TRUE.
      RETURN
   23 IF(ORDER .GE. L2)  GO TO 50
      SW(7)=.TRUE.
      RETURN
   27 CONTINUE
C     FIRST COWELL POINT -- OPTIMIZE STEP
      OPTST=(TOL3/SUM*CDEL**FN1)**(1.D0/FN1)
      IF(OPTST .LT. CDEL)  GO TO 300
      PX(N+3)=CSAVE(1)
      CX(N+3)=CSAVE(2)
      IF(MODE.EQ.2)  GO TO 20
      ORDER=L1+(L2-L1)/2
      N=ORDER-2
      GO TO 20
  300 RETURN
C     ERROR EXIT
  110 SW(19)=.TRUE.
      MODE=4
      WRITE(3,210) MODE
  210 FORMAT (68H0 POSSIBLE HALVING AND DOUBLING LOOP.  MODE WILL BE CHA
     *NGED TEMP. TO  I3)
      IF(.NOT.TEMP1)  RETURN
      TEMP3=TEMP4
      CDEL=CDEL/2.D0
      TEMP4=CDEL
      SW(6)=.FALSE.
      GO TO 65
  130 WRITE (3,230) TOUT
  230 FORMAT (1H0, 48HORDER CYCLE DOES NOT CONVERGE WITHIN SET LIMITS.
     *        1H0, 40X, 11HFINAL TIME= D24.12)
      SW(9)=.TRUE.
      RETURN
  170 ORDER=L1+(L2-L1)/2
      N=ORDER-2
      GO TO 60
      END
@ ELT TNODE,1,670830, 47605
@ EOF @
      SUBROUTINE TNODE
      DOUBLE PRECISION AT,Z,FJ,X,XD,XDD,XXDD,DIFF,CDEL,TT,T,TOL1,TOL2,
     1,TI,PRO,PRO1,FX,FXD,A,OUTT,TREQ,TOUT,FXDD
      DOUBLE PRECISION XNODE,XDNODE,TNOD,CC
      DOUBLE PRECISION FND,D,DD,C,D1,D2,SUM,SUM1,ANS,ANS1
      DOUBLE PRECISION ANS2,DDD,D3,SUM2
```

```
      INTEGER ORDER
      LOGICAL ISWT,SW
      DIMENSION D(100,3),DD(100,3),DDD(100,3),D1(20,3),D2(20,3),D3(20,
     1,ANS(3),ANS1(3)
      DIMENSION AT(20),Z(20),A(20),OUTT(20)
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     1              CDEL,TT,T,TREQ,TOUT,K,N
      COMMON/NODE/XNODE(200,3),XDNODE(200,3),TNOD(200),C(20),CC(20),
     1KK,J1,NDIFF,NEXT,KEXT,INODE
      COMMON/LIMITS/TOL1,TOL2,TC,ISCT,ISWT(30),SW(20),ORDER,L1,L2,MODE
      COMMON/INTERP/FX(60,3),FXD(60,3),FXDD(60,3),TI,K1,M,M1
      FND=NEXT
      PRO=1.D0
      INODE=INODE+1
      DO 1 I=1,8
      J=I-1
      L=K-J
      Z(I)=X(L,3)
      PRO=-Z(I)*PRO
      FJ=J
      AT(I)=(T-FJ*CDEL)
      OUTT(I)=AT(I)/TC
    1 CONTINUE
      TI=0.D0
      DO 2 J=1,8
      PRO1=1.D0
      DO 3 I=1,8
      IF(J.EQ.I) GO TO 3
      PRO1=PRO1*(Z(J)-Z(I))
    3 CONTINUE
      A(J)=-PRO/(Z(J)*PRO1)
    2 TI=TI+A(J)*AT(J)
      K1=K
      M=8
      M1=1
      SW(10)=.TRUE.
      CALL HEMINT
      DO 4 I=1,3
      XNODE(INODE,I)=FX(1,I)
    4 XDNODE(INODE,I)=FXD(1,I)
      TNOD(INODE)=TI
      IF(ISWT(18)) RETURN
      IF (INODE.LT.NDIFF) GO TO 10
      DO 11 I=1,3
      DO 11 J=KK,J1
      J2=2+NEXT*(J-1)
      DDD(J,I)=    (TNOD(J2)-TNOD(J2-1))
      D(J,I)=    (XNODE(J2,I)-XNODE(J2-1,I))
   11 DD(J,I)=    (XDNODE(J2,I)-XDNODE(J2-1,I))
      DO 12 I=1,3
      DO 12 J=1,KEXT
      CALL FKDIFF(D,D1,J,J1,I)
      CALL FKDIFF(DD,D2,J,J1,I)
      CALL FKDIFF(DDD,D3,J,J1,I)
   12 CONTINUE
      IF(SW(14)) GO TO 20
      DO 14 I=1,3
      SUM=0.D0
      SUM1=0.D0
      SUM2=0.0D0
```

```
      DO 15 J=1,KEXT
      SUM2=SUM2+C(J)*D3(J,1)
      SUM=SUM+C(J)*D1(J,I)
   15 SUM1=SUM1+C(J)*D2(J,I)
      J2=(J1-1)*NEXT+1
      ANS(I)=(SUM+D(J1,I))*FND+XNODE(J2,I)
      ANS1(I)=(SUM1+DD(J1,I))*FND+XDNODE(J2,I)
   14 CONTINUE
      ANS2=(SUM2+DDD(J1,1))*FND+TNOD(J2)
      INODE=INODE+(NEXT-1)
      T=ANS2
      TNOD(INODE)=ANS2
      DO 16 I=1,3
      XNODE(INODE,I)=ANS(I)
      XDNODE(INODE,I)=ANS1(I)
      X(1,I)=ANS(I)
   16 XD(1,I)=ANS1(I)
      X(1,3)=0.0D0
      SW(12)=.TRUE.
      NDIFF=NDIFF+NEXT
      J1=J1+1
      KK=J1
      IF(ISWT(15)) RETURN
      SW(14)=.TRUE.
   10 CONTINUE
      RETURN
   20 CONTINUE
C     CORRECTING EXTRAPOLATED VALUES
      DO 21 I=1,3
      SUM= 0.D0
      SUM1=0.D0
      SUM2=0.D0
      DO 22 J=1,KEXT
      SUM2=SUM2+CC(J)*D3(J,I)
      SUM=SUM+CC(J)*D1(J,I)
   22 SUM1=SUM1+CC(J)*D2(J,I)
      J2=(J1-2)*NEXT+1
      X(1,I)=(SUM+D(J1,I))*FND+XNODE(J2,I)
   21 XD(1,I)=(SUM1+DD(J1,I))*FND+XDNODE(J2,I)
      X(1,3)=0.0D0
      T=(SUM2+DDD(J1,1))*FND+TNOD(J2)
      INODE=INODE-1
      J=INODE
      TNOD(J)=T
      DO 23 I=1,3
      XNODE(J,I)=X(1,I)
   23 XDNODE(J,I)=XD(1,I)
      SW(14)=.FALSE.
      SW(12)=.TRUE.
      RETURN
      END
@ ELT TRANS,1,670830, 47606
@ EOF @
      SUBROUTINE TRANS(TX,TXD)
      DOUBLE PRECISION X,XD,XDD,XXDD,DIFF,CDEL,TT,T,TREQ,TOUT
      DOUBLE PRECISION ELEM,TX,TXD,DSIN,DCOS,DSQRT
      DOUBLE PRECISION SING,COSG,SINH,COSH,SINI,COSI,SINE,COSE
      DOUBLE PRECISION N,B,P,Q,L,E,DE,C,D,TEP
      COMMON/WORKER/X(200,3),XD(200,3),XDD(200,3),XXDD(3),DIFF(60,3),
     1              CDEL,TT,T,TREQ,TOUT,K,M
```

```
      COMMON/EMS/ELEM(6),TEP
      DIMENSION TX(3),TXD(3)
      DIMENSION P(3),Q(3)
      SING=DSIN(ELEM(5))
      COSG=DCOS(ELEM(5))
      SINH=DSIN(ELEM(6))
      COSH=DCOS(ELEM(6))
      SINI=DSIN(ELEM(3))
      COSI=DCOS(ELEM(3))
      N=1.0D0/(DSQRT(ELEM(1)**3))
      B=ELEM(1)*DSQRT(1.0D0-ELEM(2)**2)
      P(1)=COSG*COSH-SING*SINH*COSI
      P(2)=SING*COSH*COSI+COSG*SINH
      P(3)=SING*SINI
      Q(1)=-(SING*COSH+COSG*SINH*COSI)
      Q(2)=COSG*COSH*COSI-SING*SINH
      Q(3)=COSG*SINI
    2 L=ELEM(4)+N*(T-TEP)
      L=DMOD(L,6.2831853071795864D0)
      E=L
    3 DE= L-E+ELEM(2)*DSIN(E)
      IF(DABS(DE)-0.2D-14.LE.0.0D0) GO TO 5
      E=E+DE/(1.0D0-ELEM(2)*DCOS(E+0.5D0*DE))
      GO TO 3
    5 SINE= DSIN(E)
      COSE=DCOS(E)
      C=COSE-ELEM(2)
      D=1.0D0-ELEM(2)*COSE
      D=N/D
      DO 10 I=1,3
      TX(I)   =ELEM(1)*P(I)*C+ B*Q(I)*SINE
   10 TXD(I)   =D*(B*COSE*Q(I)-ELEM(1)*SINE*P(I))
      RETURN
      END
@ ELT TRANS2,1,670830, 47607
@ EOF @
      SUBROUTINE TRANS2 (X,XD)
C     TO COMPUTE ELEMENTS FROM POSITION AND VELOCITY VECTORS
      DOUBLE PRECISION X,XD,ELEM,RSQ,VSQ,RRD,P,PSQ,DATAN2,DSQRT,ES
     1INE,ECOSE,R,E,Y,DATAN,SINH,COSH,SINI,DSIN,Q,SINU,SINV,COSU,COSV
     2,TEP
      DIMENSION X(3),XD(3),P(3)
      COMMON/EMS/ELEM(6),TEP
      RSQ=0.D0
      VSQ=0.D0
      RRD=0.D0
      PSQ=0.D0
      P(1)=X(2)*XD(3)-X(3)*XD(2)
      P(2)=X(3)*XD(1)-X(1)*XD(3)
      P(3)=X(1)*XD(2)-X(2)*XD(1)
      DO 1 I=1,3
      RSQ=RSQ+X(I)**2
      VSQ=VSQ+XD(I)**2
      RRD=RRD+X(I)*XD(I)
    1 PSQ=PSQ+P(I)**2
      Q=DSQRT(PSQ)
      R=DSQRT(RSQ)
      ELEM(1)= R/(2.D0-R*VSQ)
      SINV= Q*RRD/R
      COSV= (PSQ/R)-1.D0
```

```
      ELEM(2)=DSQRT(SINV**2+COSV**2)
      SINV=SINV/ELEM(2)
      COSV=COSV/ELEM(2)
      ESINE=R*SINV/(ELEM(1)*DSQRT(1.D0-ELEM(2)**2))
      ECOSE=R*COSV/ELEM(1)+ELEM(2)
      E=DATAN2(ESINE,ECOSE)
      ELEM(4)=E-ELEM(2)*ESINE
      Y=DSQRT(1.0D0-(P(3)/Q)**2)/(P(3)/Q)
      ELEM(3)=DATAN(Y)
      SINI=DSIN(ELEM(3))
      SINH=P(1)/(Q*SINI)
      COSH=-P(2)/(Q*SINI)
      ELEM(6)=DATAN2(SINH,COSH)
      SINU=X(3)/(R*SINI)
      COSU=(X(1)*COSH+X(2)*SINH)/R
      ELEM(5)=DATAN2(SINU,COSU)-DATAN2(SINV,COSV)
      IF(ELEM(4).LT.0.0D0) ELEM(4)=ELEM(4)+6.2831853071795864D0
      IF(ELEM(5).LT.0.0D0) ELEM(5)=ELEM(5)+6.2831853071795864D0
      IF(ELEM(6).LT.0.0D0) ELEM(6)=ELEM(6)+6.2831853071795864D0
      RETURN
      END
@ ELT YMDAY,1,670830, 47607
@ EOF @
      DOUBLE PRECISION FUNCTION YMDAY(IYMD,IHM,SEC)
      DOUBLE PRECISION SEC
      IY=(IYMD/10000)*10000+101
      IHMS=IHM *100
      CALL DIFF(IY,0,IYMD,IHMS,ID,IS)
      YMDAY=86400*(ID+1)+IS
      YMDAY=(YMDAY+SEC)/8.64D4
      RETURN
      END



@ ELT EPHQAN,1,670913, 46904
@ EOF @
      SUBROUTINE EPHQAN
      DOUBLE PRECISION DUMMY,C,DAY1,CENTER,D,TIME(11),VAR(11)
      DOUBLE PRECISION F1
      DIMENSION A0(9,11)
      DOUBLE PRECISION DSTART
      COMMON/CONST1/DUMMY(14),IYBEG
      COMMON/COFIT/C(5,9),DAY1,CENTER,DSTART
      IY1=IYBEG-59
      CENTER=DAY1+2.5D0
      DO 5 I=1,11
      F1=I-1
      D=DAY1+.5D0*F1
      TIME(I)=D-CENTER
    5 CALL EPHEM(IY1,D,A0(1,I))
      DO 10 I=1,9
      DO 15 J=1,11
   15 VAR(J)=A0(I,J)
   10 CALL COEFF(VAR,TIME,11,4,C(1,I))
      DAY1=DAY1+5.D0
      RETURN
      END
```

APPENDIX A

RUNGE KUTTA INTEGRATION FORMULAS

$$K_0 = h\ f(X_0,\ Y_0) \tag{1}$$

$$K_1 = h\ f\left(X_0 + \frac{4}{27}h,\ Y_0 + \frac{4}{27}K_0\right) \tag{2}$$

$$K_2 = h\ f\left(X_0 + \frac{2}{9}h,\ Y_0 + \frac{1}{18}(K_0 + 3K_1)\right) \tag{3}$$

$$K_3 = h\ f\left(X_0 + \frac{1}{3}h,\ Y_0 + \frac{1}{12}(K_0 + 3K_2)\right) \tag{4}$$

$$K_4 = h\ f\left(X_0 + \frac{1}{2}h,\ Y_0 + \frac{1}{8}(K_0 + 3K_3)\right) \tag{5}$$

$$K_5 = h\ f\left(X_0 + \frac{2}{3}h,\ Y_0 + \frac{1}{54}(13K_0 - 27K_2 + 42K_3 + 8K_4)\right) \tag{6}$$

$$K_6 = h\ f\left(X_0 + \frac{1}{6}h,\ Y_0 + \frac{1}{4320}(389K_0 - 54K_2 + 966K_3 - 824K_4 + 243K_5)\right) \tag{7}$$

$$\begin{aligned} K_7 = {} & h\ f\left(X_0 + h,\ Y_0 + \frac{1}{20}(-231K_0 + 81K_2 - 1164K_3\right. \\ & \left. + 656K_4 - 122K_5 + 800K_6)\right) \end{aligned} \tag{8}$$

$$\begin{aligned} K_8 = {} & h\ f\left(X_0 + \frac{5}{6}h,\ Y_0 + \frac{1}{288}(-127K_0 + 18K_2 - 678K_3\right. \\ & \left. + 456K_4 - 9K_5 + 576K_6 + 4K_7)\right) \end{aligned} \tag{9}$$

$$K_9 = h\,f\left(X_0 + h,\ Y_0 + \frac{1}{820}\,(1481K_0 - 81K_2 + 7104K_3 - 3376K_4 + 72K_5 - 5040K_6 - 60K_7 + 720K_8)\right) \qquad (10)$$

$$Y(X_0 + h) = Y(X_0) + \frac{1}{840}\,(41K_0 + 27K_3 + 272K_4 + 27K_5 + 216K_6 + 216K_8 + 41K_9)$$

Referecnce: Shanks, E. (1966): "Solutions of Differential Equations by Evaluations of Functions," Math. of Compnt., Vol. 20, pp 21-38.

APPENDIX B

FORCE MODEL

The equations of motion used in the program are of the form

$$\ddot{\bar{X}} = \ddot{\bar{X}}_{\oplus} + \ddot{\bar{X}}_{☾} + \ddot{\bar{X}}_{\odot} + \ddot{\bar{X}}_{D} + \ddot{\bar{X}}_{*}$$

where

(A) $\ddot{\bar{X}}_{\oplus}$ - the earth's gravitational effects

(B) $\ddot{\bar{X}}_{☾}$, $\ddot{\bar{X}}_{\odot}$ - lunar and solar gravitational effects

(C) $\ddot{\bar{X}}_{D}$ - earth's atmospheric drag effects

(D) $\ddot{\bar{X}}_{*}$ - solar radiation effects.

The computation of the required earth, lunar and solar ephmeris quantities required by these perturbative accelerations is described in Section (E).

In what follows, the formulation used to compute $\ddot{\bar{X}}(t)$, given $\bar{X}(t)$, $\dot{\bar{X}}(t)$ and t is detailed. We remark that the unit of length = 6378.166 km and unit of time = 806.81242 secs.

(A) EARTH'S GRAVITATIONAL PERTURBATION

The earth's gravitational perturbation is given by

$$\ddot{\bar{X}}_{\oplus} = -\frac{\mu \bar{X}}{r^3} + \frac{\partial \mu}{\partial \bar{X}}$$

where U, the disturbing potential, expressed in the spherical coordinates of the satellite r, λ, ψ, is given by

Reference for section (A) and (E): WRDC Final Technical Report, "Unified Geodetic Parameter Program (GEOPS)", December 1964 to March 1966, Vol. 1, Mathematical Analysis by Kahler and Wells. pp 24-28, appendices A and B.

$$U = \frac{\mu}{r}\left[\sum_{n=2}^{N}\sum_{m=0}^{n}\left(\frac{b}{r}\right)^{n}(C_{nm}\cos m\lambda + S_{nm}\sin m\lambda)\,P_n^m(\sin\psi)\right]$$

where

b = mean radius of earth (unit of length)

r = geocentric radius of satellite

λ = longitude of satellite measured eastward, given by $\lambda = \alpha - \theta_g$, where α, the right ascension is given by $\alpha = \tan^{-1}(y/x)$, and θ_g is the right ascension of Greenwich at time t, referenced to true equinox (see Section E).

Sin ψ = sine of the geocentric latitude, given by $\sin\psi = z/r$

$C_{n,m}$, $S_{n,m}$ = the unnormalized harmonic coefficients (stored as data arrays with a maximum index of 20).

P_n^m = the m^{th} derivative of the n^{th} Legendre polynomial

μ = product of gravitational constant and mass of the earth.

The perturbative force, in rectangular coordinates is given by

$$\frac{\partial U}{\partial \overline{X}} = \frac{\partial U}{\partial r}\frac{\partial r}{\partial \overline{X}} + \frac{\partial U}{\partial \lambda}\frac{\partial \lambda}{\partial \overline{X}} + \frac{\partial U}{\partial \psi}\frac{\partial \psi}{\partial \overline{X}}$$

where

$$\frac{\partial U}{\partial r} = -\frac{\mu}{r}\left[\sum_{n=1}^{N}\sum_{m=0}^{n} b^n (n+1)\,\frac{\{CC_{nm} + SS_{nm}\}}{r^{n+1}}\,P_n^m(\sin\psi)\right]$$

$$\frac{\partial U}{\partial \lambda} = \frac{\mu}{r}\left[\sum_{n=0}^{N} \sum_{m=0}^{n} \frac{\{SC_{nm} - CS_{nm}\}}{r^n} P_n^m (\text{Sin } \psi)\right]$$

$$\frac{\partial U}{\partial \psi} = \frac{\mu}{r}\left[\sum_{n=0}^{N} \sum_{m=0}^{n} \left\{P_n^{m+1} (\text{Sin } \psi) - m \tan \psi \, P_n^m (\text{Sin } \psi)\right\} \cdot \frac{\{CC_{nm} + SS_{nm}\}}{r^n}\right]$$

where

$$CC_{nm} = C_{nm} \text{ Cos } m\lambda$$

$$SS_{nm} = S_{nm} \text{ Sin } m\lambda$$

$$SC_{nm} = S_{nm} \text{ Cos } m\lambda$$

$$CS_{nm} = C_{nm} \text{ Sin } m\lambda$$

and are computed using the relations

$$\text{Cos } m\lambda = 2 \text{ Cos } \lambda \text{ Cos}(m-1)\lambda - \text{Cos}(m-1)\lambda$$

$$\text{Sin } m\lambda = 2 \text{ Cos } \lambda \text{ Sin}(m-1)\lambda - \text{Sin}(m-1)\lambda$$

and where the P_n^m (Sin ψ) are given recursively by

$$P_n^o = \left[(2n - 1) \text{ Sin } \psi \, P_{n-1}^o - (n - 1) \, P_{n-2}^o\right] / n \qquad (m = 0)$$

$$P_n^n = (2n - 1) \text{ Cos } \psi \, P_{n-1}^{n-1} \qquad (m = n)$$

$$P_n^m = P_{n-2}^m + (2n - 1) \text{ Cos } \psi \, P_{n-1}^{m-1} \qquad (m \neq 0, \ m \neq n)$$

Finally, the position partials of r, λ, ψ are given by

$$\frac{\partial r}{\partial \bar{X}} = \left(\frac{x}{r}, \frac{y}{r}, \frac{z}{r}\right)$$

$$\frac{\partial \lambda}{\partial \bar{X}} = \left(\frac{\partial \lambda}{\partial x}, \frac{\partial \lambda}{\partial y}, \frac{\partial \lambda}{\partial z}\right)$$

where

$$\frac{\partial \lambda}{\partial x} = \frac{-y}{x^2 + y^2}, \quad \frac{\partial \lambda}{\partial y} = \frac{x}{x^2 + y^2}, \quad \frac{\partial \lambda}{\partial z} = 0$$

$$\frac{\partial \psi}{\partial \bar{X}} = \left(\frac{\partial \psi}{\partial x}, \frac{\partial \psi}{\partial y}, \frac{\partial \psi}{\partial z}\right)$$

where

$$\frac{\partial \psi}{\partial x} = \frac{-xz}{r^2 \sqrt{x^2 + y^2}}, \quad \frac{\partial \psi}{\partial y} = \frac{-yz}{r^2 \sqrt{x^2 + y^2}}, \quad \frac{\partial \psi}{\partial z} = \frac{\sqrt{x^2 + y^2}}{r^2}$$

(B) LUNAR AND SOLAR GRAVITATIONAL EFFECTS

The moon's gravitational perturbation is given by

$$\ddot{\overline{X}} = -\mu M_{\mathbb{C}} \left[\frac{\overline{X} - \overline{X}_{\mathbb{C}}}{r_{\mathbb{C}s}^{3}} + \frac{\overline{X}_{\mathbb{C}}}{r_{\mathbb{C}}^{3}} \right]$$

where

$\overline{X}$ = satellite position vector

$\overline{X}_{\mathbb{C}}$ = moon's position vector expressed in the geocentric system (Section E)

$M_{\mathbb{C}}$ = the lunar mass expressed in earth masses

$r_{\mathbb{C}s}$ = satellite-moon radius vector

$r_{\mathbb{C}}$ = moon radius vector, and

μ = product of gravitational constant and mass of the earth.

The solar gravitational effect is computed precisely as above, with $\overline{X}_{\mathbb{C}}$, $M_{\mathbb{C}}$, $r_{\mathbb{C}s}$, $r_{\mathbb{C}}$ replaced by $\overline{X}_{\odot}$, $M_{\odot}$, $r_{\odot s}$, $r_{\odot}$.

(C) ATMOSPHERIC DRAG EFFECTS

The drag effect on the accelerations is given by

$$\ddot{\overline{X}}_{DRAG} = -\frac{C_D A}{2M} \left[1 + \rho_1\right] \left[1 + \rho_2 t\right] \rho(h) \; |\overline{V}_r| \overline{V}_r$$

where

C_D = atmospheric drag constant

A = cross-sectional area of satellite in CM^2

M = mass of satellite in gms.

ρ_1, ρ_2 = atmospheric density correction parameters

$\rho(h)$ = atmospheric density at satellite height h

$\overline{V}_r$ = relative velocity vector of the satellite, given by $(\dot{x} - w_e y, \; \dot{y} - w_e x, \; \dot{z})$, where w_e is the angular velocity of the earth in rad./c.u.t.

θ = the angle denoting the lag between the atmospheric bulge angle and local noon.

The atmospheric density $\rho(h)$ is given by

$$\rho(h) = \rho_N(h) + \left[\rho_D(h) - \rho_N(h)\right] \text{Cos}^m (\psi/2)$$

where ρ_N (h), ρ_D (h) are night and day density values computed from a set of tabular values ρ_N (h_i), ρ_D (h_i) as follows:

A table look-up is performed to obtain the index i satisfying

$$h_i < h < h_{i+1},$$

from which ρ_N (h) or ρ_D (h) are computed by

$$\rho_j (h) = \rho_j(h_i)\, e_{xp} \left[\frac{h_i - h}{H_j}\right]$$

$$H_j = h_i - h_{i+1}/\ln \left[\rho_j (h_{i+1})/\rho_j (h_i)\right]$$

where $j = 1$ for night density, and $j = 2$ for day density.

Also, Cos ψ is given by

$$\text{Cos}\, \psi = \overline{X}^* \cdot \overline{X}^*_{Bulge}$$

where * denotes unit vector and

$$\overline{X}^*_{Bulge} = \left(X^*_{\odot} \text{ Cos } \theta - Y^*_{\odot} \text{ Sin } \theta,\ Y^*_{\odot} \text{ Cos } \theta + X^*_{\odot} \text{ Sin } \theta,\ Z^*_{\odot} \right)$$

where $(X^*_{\odot}, Y^*_{\odot}, Z^*_{\odot})$ is the unit vector of the sun's position.

(D) SOLAR RADIATION EFFECTS

The effects due to solar radiation are given by

$$\ddot{\overline{X}}_* = -\frac{\sigma A}{M}\frac{(\overline{X}_\odot - \overline{X})}{|\overline{X}_\odot - \overline{X}|}$$

where

A = cross-sectional area of satellite in cm^2

M = mass of satellite in gms

$\overline{X}_\odot$ = sun's position vector

σ = solar radiation constant dyne/cm^2

$\overline{X}$ = satellite position vector

The sunlight-shadow determination is given by

$\overline{X}^* . \overline{X}^*_\odot \geq 0$ = the satellite is on the solar side of a plane normal to the earth-sun line. No shadowing can occur.

$\overline{X}^* . \overline{X}^*_\odot < 0$ = the satellite will be shadowed only if the distance between the satellite and the earth-sun line is less than the earth's radius, hence if

$$|\overline{X} \times \overline{X}^*_\odot| < T$$

where

$$T = 1 - f\left(Z + Z_\odot \sqrt{r^2 - 1}\right)^2,$$

then the satellite is in the earth's shadow. Otherwise, it is in sunlight.

Note: * denotes unit vector, and f is the flattening coefficient of the earth.

(E) EPHEMERIS QUANTITIES

The right ascension of Greenwich at the time of interest (t) is computed from

$$\theta g = \theta g_0 + (t - t_0)\dot{\theta}_1 + (t - t_0)\dot{\theta}_2 + \Delta\mu$$

where

θg_0 = right ascension of Greenwich at Jan 0.0 for the year of interest

$\dot{\theta}_1$ = 0.9856473354 deg/mean solar day

$\dot{\theta}_2$ = 360.9856473354 deg/mean solar day

$\Delta\mu$ = equation of equinoxes

$(t - t_0)$ = time in days from Jan 0.0 for the year of interest.

The equation of equinoxes $\Delta\mu$ is obtained in the process of computing the lunar and solar ephemerides.

The ephemeris quantities are computed at ten equal intervals over a five day period and least squares fit to a fourth order polynomial in time about the midpoint of the five day period. The positions are determined for intermediate points by evaluating the polynomial at the required time.

To calculate the earth-centered slant range and inertial unit vectors to the sun and moon and the equation of the equinoxes at a given time, the time interval from epoch (1900 Jan 0.5) is denoted by T when measured in Julian centuries, by $D = 3.6525T$ and $d = 10000D$.

Angular variables:

ϵ - mean obliquity of the ecliptic

$L_{\odot}$ - geometric mean longitude of the sun, mean equinox of date

T - mean longitude of solar perigee

$L_{☾}$ - mean longitude of the moon, measured in the ecliptic from the mean equinox of date to the mean ascending node of the lunar orbit, and then along the orbit

T' - the mean longitude of lunar perigee, measured in the ecliptic from the mean equinox of date to the mean ascending node of the lunar orbit, and then along the orbit

Ω - the longitude of the mean ascending node of the lunar orbit on the ecliptic; measured from the mean equinox of date.

ℓ - $L_{☾}$ minus the mean longitude of the lunar perigee, measured in the ecliptic from the mean equinox of date to the mean ascending node of the lunar orbit, and then along the orbit

ℓ' - geometric mean longitude of the sun minus the mean longitude of perigee of the sun

F - $L_{☾}$ minus the longitude of the mean ascending node of the lunar orbit on the ecliptic; measured from the mean equinox of date

D - $L_{☾}$ minus the geometric mean longitude of the sun.

$\Delta\psi$ - nutation in longitude

$\Delta\epsilon$ - nutation in obliquity

ϵ - true obliguity of the ecliptic

$\Delta\mu$ - equation of equinoxes or nutation in right ascension

$\epsilon_{\odot}$ - eccentricity of sun in an earth centered coordinate system

$a_{\odot}$ - semi major axis of earth's orbit about the sun (cul)

$r_{\odot}$ - radius vector to sun

$\ell_{\odot}$ - apparent longitude of sun

$X_{\odot}, Y_{\odot}, Z_{\odot}$ - inertial unit vectors to sun

$a_{☾}$ - semi major axis of lunar orbit (cul)

$r_{☾}$ - radius vector to moon

$e_{☾}$ - eccentricity of lunar orbit

$X_{☾}, Y_{☾}, Z_{☾}$ - inertial unit vectors to moon

$\lambda_{☾}$ - apparent longitude of moon

$\phi_{☾}$ - apparent latitude of moon

ν - true anomaly of the sun at Jan 0.0 for the year of interest

$\dot{\theta}_1$ - deg/mean solar day.

$$\epsilon = 23°.452294 - 0°.0130125T - 0.164 \times 10^{-5} T^2$$

$$L_{\odot} = 279° 41' 48''.04 + .12960276813'' \times 10^9 T + 1''.089 T^2$$

$$T = 281° 13' 15''.00 + 6189''.03T + 1''.63T^2$$

$$L_{☾} = 270°.434164 + 13°.1763865268d - .85 \times 10^{-4} D^2$$

$$T' = 334°.329556 + 0°.1114040803d - .7739 \times 10^{-3} D^2$$

$$\Omega = 259°.183275 - 0°.0529539222d + .1557 \times 10^{-3} D^2$$

$$\ell = L_{☾} - T'$$

$$\ell' = L_{\odot} - T$$

$$F = L_{☾} - \Omega$$

$$D = L_{☾} - L_{\odot}$$

$$\begin{aligned}\Delta\psi = {} & .2088 \sin 2\Omega - (17''.2327 + .01737T) \sin \Omega \\ & - 1.273 \sin (F - D + \Omega) - .2037 \sin (2F + 2\Omega) \\ & + .1259 \sin (\ell') - .0496 \sin (\ell' + 2F - 2D + 2\Omega) \\ & + .0214 \sin (-\ell' + 2F - 2D + 2\Omega) + .0675 \sin (\ell) \\ & - .0342 \sin (2F + \Omega) - .0261 \sin (\ell + 2F + 2\Omega) \\ & + .0114 \sin (-\ell + 2F + 2\Omega) + .0124 \sin (2F - 2D + \Omega) \\ & - .0149 \sin (\ell - 2D)\end{aligned}$$

$$\Delta\epsilon = 9''.2106 \cos \Omega - .0904 \cos 2\Omega$$

$$+ .552 \cos 2(F - D + \Omega) + .0884 \cos 2(F + \Omega)$$

$$+ .0183 \cos(2F + \Omega) + .0216 \cos(\ell' + 2F - 2D + 2\Omega)$$

$$\epsilon = \epsilon_0 + \Delta\epsilon$$

$$\Delta\mu = \Delta\psi \cos \epsilon$$

$$\epsilon_{\odot} = .01675104 - .418 \times 10^{-4}\, T$$

$$\ell_{\odot} = L_{\odot} + 2e_{\odot} \sin(\dot{\theta}_1 (t - t_0) - \nu)$$

$$X_{\odot} = \cos(\ell_{\odot})$$

$$Y_{\odot} = \sin(\ell_{\odot}) \cos \epsilon$$

$$Z_{\odot} = \sin(\ell_{\odot}) \sin \epsilon$$

$$r_{\odot} = a_{\odot} (1 - e_{\odot}^2)/[1 + e_{\odot} \cos(\ell_{\odot} - T)]$$

$$\begin{aligned}
\lambda_{☾} = \Big[& 206265\,L + 22640 \sin(\ell) - 4586 \sin(\ell - 2D) \\
& + 2370 \sin(2D) + 769 \sin(2\ell) - 668 \sin(\ell') \\
& - 412 \sin(2F) - 212 \sin(2\ell - 2D) \\
& - 206 \sin(\ell + \ell' - 2D) + 192 \sin(\ell + 2D) \\
& - 165 \sin(\ell' - 2D) + 148 \sin(\ell - \ell') \\
& - 125 \sin(D) - 109 \sin(\ell + \ell') - 55 \sin(2F - 2D) \\
& - 45 \sin(\ell + 2F) + 40 \sin(\ell - 2F) - 38 \sin(\ell - 4D) \\
& + 36 \sin(3\ell) - 31 \sin(2\ell - 4D) + 28 \sin(\ell - \ell' - 2D) \\
& - 24 \sin(\ell' + 2D) + 19 \sin(\ell - D) \\
& + 18 \sin(\ell' + D)\Big] /206265
\end{aligned}$$

$$\phi_{☾} = \Big[18520 \sin(S) \Big\{1 - .00293 \sin(1.403808 - .0009242203T)\Big\} - 31 \sin(F - \ell - 2D) - 25 \sin(F - 2\ell) - 23 \sin(\ell' + F - 2D) + 21 \sin(F - \ell) - 526 \sin(F - 2D) + 44 \sin(\ell + F - 2D)\Big] /206265$$

$$X_{☾} = \cos\phi \cos\lambda$$

$$Y_{☾} = \cos\phi \sin\lambda \cos\epsilon - \sin\phi \sin\epsilon$$

$$Z_{☾} = \cos\phi \sin\lambda \sin\epsilon + \sin\phi \cos\epsilon$$

$$r_{☾} = a_{☾} (1 - e_{☾}^2) / (1 + e_{☾} \cos\ell)$$