Air Force Report No.
SSD-TR-67-XXX-Vol. _____

IBM Report No.
67-M50-007

PHASE I FINAL TECHNICAL REPORT (DRAFT)

(U) QUICK REACTION GUIDANCE AND TARGETING STUDY

Volume 4

~~Sections I through~~ APPENDICES

By

H.R. Robbins, P.P. Mooney, et al
Space Systems Center, FSD, IBM Corporation

JUNE 1967

Prepared For
Advanced Development Directorate
Space Guidance Program Division
Air Force Systems Command
Los Angeles Air Force Station, California

Appendix I

TGP - TRAJECTORY GENERATOR PROGRAM

1.1 INTRODUCTION

The Trajectory Generator Program (TGP) is a digital computer program written in FORTRAN IV to run on IBM 7090/7094 computers. The program is used in mission analysis and planning to generate a preliminary trajectory (or trajectories) from launch to orbit injection followed by several options for performing orbit transfer with or without rendezvous.

The program employs analytical equations for determining launch conditions and the atmospheric phase of ascent. The vacuum phase of ascent (to orbit injection or intercept) is generated with an explicit guidance algorithm. All orbital burns are considered as impulses and are planned by two different techniques to provide alternate solutions. The program is based on a spherical earth model which allows for the rapid generation of coast segments with Kepler-type equations.

This program was written to demonstrate feasibility of the Preliminary Trajectory Generation Mode of the On-Board Mission Planning Function (see Section V of the main report). TGP is used to generate initial trajectories for the Trajectory Optimizer Program (TOP) which is described in Appendix II.

1.2 PURPOSE AND USE

TGP was written to demonstrate preliminary feasibility of on-board trajectory generation for the QRGT concept. The program also demonstrates the capability of the generalized guidance algorithms, developed during this study, for mission planning purposes. TGP represents the "Explicit" aspect of the "Optimal-Explicit" approach to on-board trajectory planning which is outlined in Par. 5.4 of the main report.

This program is employed to generate preliminary trajectories for various missions that may employ the QRGT concept. These preliminary trajectories are specified a set of parameters which identify: launch conditions, ascent to orbit, and all major orbit burns. These parameter sets then become the input to the Trajectory Optimizer Program-TOP (Appendix II) where they are systematically varied in order to minimize some mission performance function (e.g. total ΔV or total mission time). TOP then represents the "Optimal" aspect of the "Optimal-Explicit" approach to trajectory planning.

1.3 PROGRAM FEATURES

The present version of TGP makes use of many subroutines generated in different subtasks of Task II during this phase of QRGT. This has resulted in some duplication and program inefficiency. These deficiencies can be corrected when a more general and sophisticated program (which will be necessary for detailed feasibility studies) is written during Phase II.

The present program employs fairly simple planning logic and launch routines in conjunction with quite sophisticated ascent and orbit maneuver routines to provide the following features:

- Direct ascent, ascent to orbit and orbit maneuvers (with or without rendezvous) can be generated.
- Spherical gravity model.
- Kepler arcs for coast phases.
- Launch conditions options for minimum out-of-plane angle or for a specified launch time.
- Maximum and minimum launch azimuth constraints included.
- Ascent to orbit or intercept with an optimal explicit guidance algorithm (UP 1 or UP 2 Subroutines, which are really two entries to a slightly modified version of OP-EX, the algorithm used for powered-flight guidance and described in Appendix III).
- Two-burn (RENDI Subroutine) or three-burn (VPG Subroutine) orbit transfers possible.
- Orbital burns treated as impulses.
- Mission can be planned from launch or from a parking orbit.
- The two burn rendezvous routine (RENDI) employs a search routine to locate local minimum- ΔV transfers.

1.4 PROGRAM OPERATION

The over-all operation of TGP is illustrated in Figure I-1. The program consists of an executive routine MAIN and seven major subroutines: LAUNCH, UP 1, UP 2, EPHEM, RVVR, RENDI and VPG. The MAIN program reads in all data, performs necessary conversions, sets up program paths, initializes and calls subroutines, and writes all output data. The LAUNCH subroutine is used to determine the launch time and launch conditions. The ascent to orbit or intercept phases is calculated in UP 1 and UP 2 respectively. Ephemeris for the spacecraft and target is updated with EPHEM. The RVVR routine is used to compute the required velocity necessary for an orbital burn. Two-burn orbital rendezvous maneuvers are generated with RENDI while VPG generates several different rendezvous trajectories with three (or more) burns per trajectory.

For each successful run, the printout includes all parameters necessary to specify each trajectory. These include: launch time and azimuth; ascent cut-off state, time, and ΔV remaining; position, velocity, time and ΔV at each orbital burn; and other related data.

1.4.1 MAIN PROGRAM

The executive routine MAIN represents the planning logic and prescriptions of the Preliminary Trajectory Generator Mode of the On-Board Mission Planning Function (see Section V of the main report). That is, it determines how a trajectory will be generated, the subroutines necessary and the linking of subroutines. In TGP this logic and prescriptions are dependent on the mission type but otherwise fixed. For example, all missions requiring rendezvous with an orbiting target (real or fictitious) are planned in the same way; only the results differ. Figure I-2 is a detailed math flow of MAIN. The symbols used may be found in Table I-1.

Referring to Figures I-1 and I-2, after reading-in and converting input data MAIN branches if a Surveillance case is to be planned (M=4), otherwise the EPHEM subroutine is called and certain parameters of the target's orbit are computed. If a parking orbit is required (M=3) the LAUNCH routine is used to determine the best time to launch; that is, when the out-of-plane angle with the target orbit is minimum. The launch time (t_L) is constrained to occur after a reference time (t_o) and before a maximum time (t_{Lf}) both of which are inputs. The outputs of LAUNCH are used with UP 1 to generate a optional ascent trajectory to a circular parking orbit.

Once in the parking orbit, the orbital burns necessary to achieve rendezvous with the target vehicle are generated as a two-burn maneuver with RENDI or as a set of possible three-burn maneuvers with VPG.

The results from RENDI and VPG are printed-out with the launch conditions and ascent phase. Selected options are then ready to be optimized with TOP (Appendix II).

When a direct ascent is called for (M=1, 2), MAIN branches after generating the target parameters. The problem here is to determine the launch conditions and an ascent trajectory so that the coast trajectory after cutoff intersects the target orbit when the target is at the intercept point. This is the case, for example, when a direct-ascent rendezvous is to be planned. The first step is to employ LAUNCH to generate launch conditions for minimum out-of-plane angle at the earliest possible launch time ($t_L = t_o$). From this, the intercept direction is determined. The EPHEM routine is then called to determine the target's position and velocity at the intercept point, and the time required to reach that point (t_f). The intercept position and time to intercept ($t_f - t_o$) are inputted to UP 2 which generates an ascent trajectory, if one exists, which will arrive at that position at the proper time. If the ΔV required on this trajectory is less than the maximum available, a solution has been found and the program writes the results and stops. If, on the other hand, the trajectory is not feasible, the program increments the launch time by an input constant (Δt_L) and proceeds to generate a new trajectory. The launch time is incremented in this fashion until a feasible trajectory results (i.e., one with $\Delta V \leq \Delta V_{max}$), or a maximum value for t_L (t_{Lmax}) is exceeded. In the latter case, the launch time is fixed at the "best" time found so far (i.e., corresponding to a minimum value of (ΔV), and the launch azimuth is now incremented from its minimum value (A_{Lmin}) by a fixed amount (ΔA_L, an input).

In this iteration the maximum and minimum values for launch azimuth are set to the desired value in LAUNCH. This results in the launch conditions and intercept point when the launch time and the launch azimuth are specified and will, in general, not result in the minimum out-of-plane angle with the target's orbit. If a solution can not be found for $A_L \leq A_{Lmax}$ the program stops. Within the above mentioned constraints of launch time and launch azimuth, this exit indicates that the target cannot be reached with a direct ascent trajectory.

In the case of surveillance missions (M=4), MAIN initially checks the possibility of a direct over-flight by proper cutoff conditions during ascent. This is done by determining the maximum inclination attainable from the launch site when launch azimuth restrictions are considered. If this maximum inclination exceeds the latitude of the point to be overflown, a first pass overflight is calculated. In this application the inputs to UP 1 are varied in an attempt to find an elliptic parking orbit which has the desired altitude over the point and gives overflight at the desired time. In this iteration the parking orbit parameters and overflight time are varied.

If a solution cannot be found or if the target's latitude exceeds the maximum possible inclination, a parking orbit and transfer maneuver is used. UP 1 is used to put the spacecraft into a circular parking orbit which is as close to the target as launch azimuth constraints allow. The EPHEM and RVVR routines are then used in an iteration loop to determine the time or the orbital burn and the overflight time (which defines an aim point for the burn). The iterations are continued until a suitable trajectory is found which results in a total ΔV (ascent to parking orbit and orbital maneuver) which is less than the vehicle's capability (ΔV_{max}).

1.4.2 LAUNCH SUBROUTINE

A math flow of the LAUNCH subroutine is presented in Figure I-3 and the corresponding symbols are in Table I-2. This routine is used to determine the launch conditions for: (a) a specified launch time, (b) the launch time which results in the minimum possible out-of-plane angle, or (c) a specified launch time and launch azimuth.

Referring to Figure I-3, option (a) corresponds to LCL positive. The specified launch time (t) is used to determine the location of the launch site relative to the target's ascending mode (ϕ_L). The launch azimuth (A_L) is then computed which will result in the minimum out-of-plane angle with the given target orbit plane. This results in a 90 degree range angle between launch and target plane intersection ($\Delta\theta$). The computed launch azimuth is compared with the allowed minimum and maximum values (A'_{Lmin} and A'_{Lmax}) and set equal to the appropriate limit when it is exceeded. If the launch azimuth is constrained the range angle $\Delta\theta$ is computed; otherwise it is set to 90 degrees. The launch conditions are then specified by generating unit vector along the launch radius ($\underline{r}_L$), in the launch plane defined by A_L ($\underline{v}_L$) and normal to the launch plane ($\underline{n}_L$). Finally, the out-of-plane angle between the launch and target planes (Δ i) is calculated.

Option (b), when LCL is negative, is used to determine the launch times (τ_1 and τ_2) when the minimum possible out-of-plane angle (Δ i) can be achieved. If the launch site latitude (λ_L) is less than the targets inclination (i_T) there are two opportunities each day when Δ i is zero; otherwise both opportunities coincide and result in a value of Δ i equal to the difference in λ_L and i_T. When this option is used, the launch unit vectors $\underline{r}_L$, $\underline{v}_L$, $\underline{n}_L$ are obtained by setting $t = \tau_1$ (or τ_2), LCL positive, and re-entering LAUNCH.

The last option (c) is just a special case of (a) where the launch azimuth limits (A'_{Lmin} and A'_{Lmax}) are both set equal to the desired launch azimuth, A_L.

1.4.3 UP 1 AND UP 2 SUBROUTINES

These two subroutines are really just two entry points to one routine. This is possible because of the large amount of commonality between the computations for ascent to a parking orbit (UP 1) and ascent to intercept (UP 2). This routine is based on the optimal-explicit guidance algorithm developed during this study which is discussed in Par. 6.4 of the main report and described in detail in Appendix III. This algorithm is used for the exoatmospheric phase of ascent and gives ascent trajectories with nearly minimal burning times (and therefore, minimum fuel consumption).

In order to initialize the explicit guidance equations, the vehicle's state at the "top of the atmosphere" (dynamic pressure less than 30 lbs/ft.) must be determined in terms of the launch conditions. Presently, both UP 1 and UP 2 determine these conditions by analytical equations which use three vehicle-dependent and payload-dependent parameters. These parameters, once determined, are then fixed and used for all launch azimuths. The three parameters, in conjunction with the launch conditions, are used to calculate the position and velocity at the beginning of explicit guidance by use of spherical trigonometry relations. This analytical technique of handling the atmospheric phase of ascent is very efficient computationally, but initial simulation results indicate prediction errors of the order of 1 - 3 seconds in the final injection time. This aspect of planning the ascent phase can and should be improved by a more accurate modeling of the atmospheric phase.

1.4.4 EPHEM SUBROUTINE

The ephemeris subroutine EPHEM, see Figure I-4 and Table I-3, is used to determine the spacecraft's or target's position and velocity ($\underline{r}$ and $\dot{\underline{r}}$) at a given epoch based on a given ephemeris ($\underline{r}_o$, $\dot{\underline{r}}_o$) and epoch (T_o). The program has two modes of operation: (1) Option A (NOP = 0) when the range angle (θ) between the epochs is specified, and (2) Option B (NOP = 1) when the transfer time (t) is specified. When Option A is employed, the transfer time (t_{of}) is calculated in EPHEM in addition to the position and velocity. This routine was formulated during the studies of orbit maneuvers and a description of its operation and the algorithms employed in it are given in Section VI of the main report.

1.4.5 RVVR SUBROUTINE

The Required Velocity Vector Routine (RVVR) is used to determine the velocity required to transfer between two positions in space. The slope (target of the flight path angle) at arrival at the second point ($^{m}{}_{a}$) is the constraint imposed to completely define the problem. Figure I-5 is a math flow of this subroutine and the symbols are defined in Table I-4.

This routine has five options available for specifying the slope m_a and thus, generally, five transfer options (KK = -1, 0, 1, 2, 3). These options were developed during this study as part of the generalized guidance equations subtask in order to study different techniques for generating two-burn orbit transfer maneuvers. The rationale and implementation of these options is discussed in Section VI of the main report.

For those cases where a 180 degree transfer is involved and the transfer plane becomes undefined, an input quantity, α, is used in the subroutine to define the plane. If this case is not known when the routine is entered, $\alpha = 0$ and the velocity at the first position is used to define the plane and an indicator is set (N180 = +1). The main program can then specify a value for α and re-enter the routine, if required. This subroutine calls its own time-of-flight routine for calculating the time on the transfer arc: TIME (Figure I-6 and Table I-5) for elliptic arcs and SIMTIM (Figure I-7 and Table I-6) for parabolic and hyperbolic arcs.

1.4.6 RENDI SUBROUTINE

The Rendezvous-Intercept routine (RENDI) is a trajectory planning program which is employed to find local minimum - ΔV two-burns orbit transfers which result in rendezvous with an orbiting target vehicle. The target may be fictitious as is the case of placing a payload in a synchronous equatorial orbit where a "target" is introduced to control the earth longitude of the final orbit injection.

The program uses the EPHEM and RVVR subroutines (paragraphs 1.4.4 and 1.4.5 above) to generate a two-burn rendezvous and a one-dimensional search routine (Subroutine BEST) to find local minimums of the total ΔV required. In TGP this routine is called to generate all local minimum ΔV rendezvous for which the first burn is allowed to occur between TSTART and TSTOP (two input times).

This routine represents the planning mode of operation of the Rendezvous-Intercept guidance algorithm which is discussed in detail in Section VI of the main report. Figure I-8 is a math flow of this subroutine and Table I-7 contains the symbols employed. The one-dimensional search routine (BEST), based on Powell's technique, is represented in Figure I-9 with a list of symbols in Table I-8.

1.4.7 VPG SUBROUTINE

The Variable Point Guidance (VPG) subroutine is a planning program used to generate a series of orbital burns which result in rendezvous with a specified target vehicle. This is an IBM program based on the technique developed in the Variable Point Guidance and Targeting Study, Reference (1).

This program was developed during this study based on earlier reports generated during the various VPG studies phases (especially references 1, 8, and 9 of Reference (1). The program incorporates many of the features reported in these earlier papers but differs in actual implementation from that reported in Reference (1). Some of the important features of the present version of the VPG subroutine are:

1. The spacecraft starts from a circular parking orbit.
2. The target (real or ficticious) is an arbitrary orbit (relative to the spacecraft).
3. All burns are considered as impulses.
4. Spherical gravity model assumed.
5. The flight path angle, relative to local horizontal, is unchanged by a burn (i.e. except for plane change the ΔV added is co-linear with the velocity).
6. All burns except the first occur on the line-of-nodes between the spacecraft and target orbits.
7. The first burn is positioned in the parking orbit to satisfy (5) and (6).
8. Eight options are generated for each input case. Four options correspond to rendezvous occurring at one node, the other four to rendezvous at the opposite node. The four options are Bielliptic Chase, Bielliptic Lob, Full Orbit Phasing Chase, and Full Orbit Phasing Lob.

9. Bielliptic maneuvers involve three major burns: the first at parking orbit altitude, the last at target altitude (on the line-of-nodes) and the second 180 degrees from the third. The altitude of the second burn is adjusted to insure proper phasing at the third burn. If this altitude is below the target orbit, it is a Chase solution, if above, it is a Lob solution.

10. Full Orbit Phasing maneuvers also involve three major burns: The first at the parking orbit and the second and third coincident at the target altitude. The second burn, at the target orbit, is adjusted to produce a phasing orbit such that after an integer number of revolutions the spacecraft and target are coincident. The third burn then matches the spacecraft's velocity to that of the target. If the period of the phasing orbit is less than that of the target, it is a Chase solution, otherwise it is a Lob solution.

11. The plane change to be effected at each burn is determined by optimally splitting the total change between the major burns. This routine includes an option (input) of also considering a pure plane change maneuver at target altitude.

One of the major differences between this program and that in Reference (1) is that the ascent to orbit is not included in the VPG subroutine. This phase of the mission is generated with the UP 1 subroutine (Paragraph 1.4.3 of this Appendix) because of the flexibility and performance it offers.

A detailed description of the philosophy underlying the Variable Point Guidance and Targeting technique as well as its implementation can be found in Reference (1). A math flow of the VPG subroutine can be found in Figure I-10 and the symbols are listed in Table I-9. VPG itself employs three subroutines: an optimal Plane Change Angle routine - PCA (Figure I-11 and Table I-10); a Required Velocity routine - REQ (Figure I-12 and Table I-11); and a Time-of-Flight routine - TOF (Figure I-13 and Table I-12). The latter two routines (REQ and TOF) were generated when VPG was initially altitude (LCO = +1, LCS = +1) the radius magnitude is also calculated internally to insure a 24 hour period ($\underline{R}_T$ is then used only to define the target's direction at the epoch, t_T).

If the mission is to start from a parking orbit (LCL = -1) the spacecraft's ephemeris is inputted:

$\underline{R}_P$ = Parking orbit position at $t = t_p$

$\underline{V}_P$ = Parking orbit velocity at $t = t_p$

t_p = Epoch time.

The type of mission to be planned is determined by the integer M:

M = 1 Intercept

M = 2 Direct Ascent Rendezvous

M = 3 Orbital Rendezvous

M = 4 Surveillance - from parking orbit

M = 5 Surveillance by first pass overflight (set in program if found possible)

For M = 4, the surveillance point is defined by:

λ_T = Latitude

Ω_T = Longitude relative to launch site

R_T = Overflight radius

δ_M = Error tolerance allowed in the overpass.

The additional input data required are: the circular parking orbit radius R_P (when LCL = +1); last possible launch time t_{Lmax}; last possible final time t_{fmax}; maximum ΔV capability of the vehicle ΔV_{max}; and increments of launch time and launch azimuth (Δt_L and ΔA_L) which are used when M = 1 or 2 to find a feasible direct ascent trajectory.

1.6 TEST CASES

Three test cases are presented in order to illustrate the performance of TGP. Table I-13 contains a description of the cases, and Table I-14 illustrates the results obtained with TGP.

written as an independent program. They were retained when VPG was converted into a subroutine for TGP (instead of using RVVR and EPHEM) because of the internal structure of VPG which would have required a major overhaul in order to make use of the more general RVVR and EPHEM routines.

1.5 PROBLEM SPECIFICATION

The coordinate system used to define the problem and in which all calculations are performed is the XYZ ECI (Earth Centered Inertial) system, illustrated in Figure I-14. The XY plane is the equatorial plane and the Z-axis is through the north pole. This figure also illustrates some of the parameters involved in the program, which are defined in Table I-1.

If the mission is to be planned from launch (LCL = +1) the launch site is specified by:

λ_L = Launch site latitude.

Ω_{Lo} = Launch site longitude relative to X-axis at t_o

t_o = Reference time.

A'_{Lmin} = Minimum allowable launch azimuth.

A'_{Lmax} = Maximum allowable launch azimuth.

Two alternative ways are provided for specifying the target orbit If LCO = -1, the target orbit is specified by its ephemeris at some epoch:

$\underline{R}_T$ = Target's position at $t = t_1$

$\underline{V}_T$ = Target's velocity at $t = t_1$

t_1 = Epoch time; or,

If a circular target orbit is desired (LCO = +1) the orbit is specified by:

$\underline{n}_c$ = Unit normal to target orbit

$\underline{R}_T$ = Target's position at $t = t_1$

t_t = Epoch time.

and the program computes the circular velocity from $|\underline{R}_T|$ and forms the velocity vector with $\underline{n}_c$ and $\underline{R}_T$. When the target orbit corresponds to a synchronous

Case 1 - This case requires planning the launch conditions and orbital maneuvers to achieve rendezvous with a fictitious target in a synchronous equatorial orbit stationary at launch site longitude. The program generates, with the LAUNCH subroutine, a launch at $t_L = 0$ at an azimuth of 90° which gives the minimum out-of-plane angle possible ($\Delta i = 28.5617° = \lambda_L$). The ascent to the circular parking orbit is calculated with UP 1 and requires 483.7 seconds with 14,444 feet/sec. of ΔV left for orbit maneuvers.

The orbit transfer necessary for rendezvous is generated as a two-burn maneuver with RENDI and as a series of three-burn maneuvers with VPG. In the case of RENDI, Table I-14 lists the results of the first local minimum ΔV found and the best local minimum in the time span allowed for the search ($t_f < 1$ day). The VPG results in Table I-14 represent one Bielliptic and one Full Orbit Phasing Option. Both are three-burn maneuvers and represent maneuvers which require less ΔV than is available (14,444 feet/sec.) and which, if possible, achieve rendezvous within 1 day (86,400 seconds).

The Bielliptic option requires 14,138 feet per second but final rendezvous is at 102,716 seconds. This is accomplished by gross phasing in the parking orbit for 7 full orbits after the first nodal crossing. The rendezvous is then effected by two 180 degree transfers, with the second burn being above the target orbit at 143,437,670 feet (target radius is 138,607,380 feet).

The second VPG option is a Full Orbit Phasing case which requires 14,032 feet/second of ΔV with rendezvous at 64,800 seconds. In this case, the phasing is accomplished with one orbit at initial altitude followed by a 180 degree transfer to target altitude. The second burn then puts the spacecraft into a phasing ellipse with a period of 38,990 seconds (target period is 86,400 seconds) and after one phasing orbit the third burn matches the velocities for rendezvous.

Case 2 - The target in this case is at Space Station Altitude (185 nm) and the problem is to plan a rendezvous mission from launch. The LAUNCH routine is used to find the most favorable launch conditions, which are t_L = 14,502 seconds and A_L = 74.9178 degrees. This results in the ascent trajectory intersecting the target orbit 90 degrees down range with zero out-of-plane angle ($i_T > \lambda_L$, so an in-orbit launch time opportunity is available twice per day). The ascent to a 100 nm parking orbit is then generated with UP 1 and results in an injection time of 15,089 seconds (585 seconds after launch) with 8,762 feet/sec. of ΔV remaining for the rendezvous maneuvers.

The orbital maneuver represented in Table I-14 is a VPG Full Orbit Phasing option which requires 6,057 feet/sec. of ΔV with rendezvous at 27,383 seconds (12,881 seconds after launch). The maneuver involves a 180 degree transfer to target altitude followed by one revolution in a phasing orbit. The period of the phasing orbit is greater than that of the target orbit (8,280 seconds vs. 5,485 seconds) so that this is a Lob option.

Case 3 - The final test case is the same as Case 2 except that a direct ascent rendezvous is to be planned. TGP finds a solution corresponding to the launch site 80.25 degrees beyond the target's ascending node ($t_L = 19,260$ seconds);this corresponds to approximately 80 minutes after the launch site has passed through the target orbit plane. The launch azimuth of 84.8443 degrees results in the intercept point being 90 degrees down range and an out-of-plane angle of 3.05 degrees. These conditions are inputted to UP 2 with the time that the target will arrive at the intercept point (2112 seconds after launch). An ascent trajectory is generated which achieves the desired final state and has 2,734 feet/sec. of ΔV left after the adaptation burn for rendezvous.

TABLE I-1

MAIN PROGRAM SYMBOLS

Math Flow Symbol	Description	Program Symbol
M	Run Type Variable	M
t_o	Reference Time	TREF
$\underrightarrow{R}_{TO}'$	Initial Position Vector (Target)	RTO
$\underrightarrow{V}_{TO}'$	Initial Velocity Vector (Target)	VTO
t_{TO}	Initial Time (Target)	TTO
μ	Gravitational Constant = $1.4082878\text{x}10^{16}$	GC
$\underrightarrow{r}$	Position Vector	RTN
$\underrightarrow{v}$	Velocity Vector	VTN
TOF	Time of Flight	TOF
MA	Slope at Arrival	SLT
a_T	Semi Major Axis of Target Orbit	A
h_T	Angular Momentum	ANG
$\underrightarrow{n}_T$	Unit Cross Product Vector (Target)	UNT
p_T	Target Semi Latus Rectum	SEMILR
e_T	Eccentricity	E
i_T	Target Inclination Angle	EYE
Ω_T	Longitude of Ascending Node	CLONG
AL_{MAX}	Maximum Allowable Launch Azimuth Angle	ALMAX

(Sheet 1 of 5)

Table I-1 Main Program Symbols (Continued)

Math Flow Symbol	Description	Program Symbol
AL_{MIN}	Minimum Allowable Launch Azimuth Angle	ALMIN
τ_1	1st Zero Out-of-Plane Launch Time	TAU1
τ_2	2nd Zero Out-of-Plane Launch Time	TAU2
λ_L	Launch Site Latitude	SLAT
Ω_{LO}	Launch Site Longitude at t_o	SLONG
t	Desired Launch Time	T
A_L	Launch Azimuth	AL
Δi	Out-of-Plane Angle	DELEYE
$\Delta\theta$	Range Angle to Target Plane Intersection	TANG
$\underset{\rightarrow}{r}_L$	Unit Vector Thru Launch Site	RL
$\underset{\rightarrow}{v}_L$	Unit Vector In Launch Plane	VL
$\underset{\rightarrow}{n}_L$	Unit Vector Normal to Launch Plane	UNITL
N	Program Counter = Number of Launch Time Calculations	N
$t'_{1,2,3,n}$	Launch Times	TP(N)
t_1	Best Launch Time	T_1
$\underset{\rightarrow}{R}_1$	Unit Vector Thru Launch Site at t_1	R_1
R_E	Earth Radius	RE
$\underset{\rightarrow}{n}_1$	Unit Vector Normal to Launch Plane at t_1	UNIT1
r_a	Apogee Radius	RA
r_p	Pergee Radius	RP
R_o	Parking Orbit Altitude	RPARK
Ω_L	Launch Site Longitude at Launch Time	XLONG

(Sheet 2 of 5)

Table I-1 Main Program Symbols (Continued)

Math Flow Symbol	Description	Program Symbol
ΔV^*	Current Max Value of ΔV_R	DVSTAR
θ_f'	Range Angle to Target	THETF
t_f'	Time of Insertion or Rendezvous	TFP
$\underrightarrow{R}_f$	Position Vector at t_f'	RF
$\underrightarrow{V}_f$	Velocity Vector at t_f'	VF
t^*	Time of Conditions with Highest Remaining ΔV	TSTAR
A_L^*	Launch Azimuth with Highest Remaining ΔV	ALSTAR
LC1	Program Indicator to Step Launch Azimuth	LC1
ΔA_L	Launch Azimuth Increment	AZ DEL
Δt_1	Launch Time Increment	TDEL
$t_{1\ MAX}$	Latest Possible Launch Time	TLMAX
i_{MAX}	Maximum Inclination Angle	EYEMAX
λT	Target Latitude	TLAT
ΔT	Elapsed Time	DELT
t_4	Transfer Time for Zero Flight Path Angle	T4
$\underrightarrow{R}_T$	Overpass Position Vector	RT
t_f	Time at Overpass Position	TF
e_f	Error in Time at Overpass Position	ERRF
t_{fMAX}	Last Possible Final Time	TFMAX

(Sheet 3 of 5)

Table I-1 Main Program Symbols (Continued)

Math Flow Symbol	Description	Program Symbol
W_E	Earth Rotational Rate	WE
t_2	Time at Injection into Parking Orbit	T2
$\underrightarrow{R}_2$	Parking Orbit Position or Cut-Off Vector at Time t_2	R2
$\underrightarrow{V}_2$	Parking Orbit or Cut-Off Velocity Vector at Time t_2	V2
ΔV_M	Remaining ΔV(Ft/Sec) Available	DELVM
ΔV_R	Remaining ΔV(Ft/Sec) Available	DELVR
$\underrightarrow{r}_f$	Target Position Vector at t_2	RTN
$\underrightarrow{v}_f$	Target Velocity Vector at t_2	VTN
$\underrightarrow{n}_2$	Unit Cross Product Vector of $\underrightarrow{R}_2$ and $\underrightarrow{V}_2$	CP
α	Cosine of Out-of-Plane Angle Between Parking Orbit and Target Orbit	ALPHA
$\underrightarrow{n}_N$	Unit Vector to Line-of-Nodes	CPPR
θ_o	Transfer Angle from Parking Orbit to Rendezvous	THETO
$\underrightarrow{R}_3$	Position Vector at Rendezvous	R3
$\underrightarrow{V}_3$	Velocity Vector at Rendezvous	V3
t_3	Time of Rendezvous	T3
$\underrightarrow{R}_{T3}$	Target Position Vector at Rendezvous	RT3
$\underrightarrow{V}_{T3}$	Target Velocity Vector at Rendezvous	VT3
Δf_3	Angular Position of Target Relative to Line-of-Nodes at t_3	DELF3
f_3	True Anomaly of Target at t_3	F3
f_{LN}	Angular Position of Line-of-Node Relative to Targets Perigee	FLN

Table I-1 Main Program Symbols (Continued)

Math Flow Symbol	Description	Program Symbol
ΔV_3	Velocity Required for Overflight	DELTV3
δ	Angular Overpass Error	DEL
$\delta *$	Maximum Allowable Overpass Error	DELSTR
Ω_{Lf}	Final Launch Longitude	XLONGF
Ω_{Tf}	Final Target Longitude	TLONGF
$\Delta \Omega_T$	Target Longitude Relative to Launch Site	TLONG
$\underset{\rightarrow}{n}_{DN}$	Unit Vector to Parking Orbits Descending Node	CPDN
ΔV_R	Remaining ΔV	DELVN
$\Delta \Omega$	Range Angle Error	DELONG
LCO	Parameter to Select Circular Orbit Option	LCO
LCS	Parameter to Select Synchronous Orbit Option	LCS
LCL	Parameter to Select Parking Orbit Data	LCL
LCT	Parameter to Select VPG Option	LCT
$\underset{\rightarrow}{R}_{po}$	Parking Orbit Position Vector	RPO
$\underset{\rightarrow}{V}_{po}$	Parking Orbit Velocity Vector	VPO
t_{po}	Parking Orbit Epoch Time	TPO
$\underset{\rightarrow}{R}_{c}$	Radius of Circular Orbit	RC
$\underset{\rightarrow}{V}_{c}$	Velocity of Circular Orbit	VC

(Sheet 5 of 5)

TABLE I-2

LAUNCH SUBROUTINE SYMBOL TABLE

Math Symbol	Description	FORTRAN Program Symbol
LCL	Option Indicator: +1 when launch time is specified; -1 determines best launch time(s)	LC
λ_L	Launch site latitude (degrees)	LAML
Ω_{LO}	Launch site longitude relative to x-axis at t_o (degrees)	OLO
Ω_T	Longitude of ascending node of target orbit (degrees)	OT
i_T	Inclination of target orbit plane to x-y plane ($i_T \geq 0$) (degrees)	IT
t	Desired launch time	T
t_0	Reference time (sec)	T0
$A'_{L\ max}$	Maximum permissable launch azimuth (degrees)	ALMAX
$A'_{L\ min}$	Minimum permissable launch azimuth (degrees)	ALMIN
τ_1	Time at which launch site is nearest target plane	T 1
τ_2	Time at which launch site is nearest target plane	T 2
A_L	Launch azimuth (degrees)	AL
Δi	Out of plane angle between launch plane and target orbit (degrees)	DELI
$\Delta\theta$	Range angle from launch site to intersection with target orbit	DELTH
$\underrightarrow{r}_L$	Unit vector through launch site	RL

(Sheet 1 of 2)

Table I-2 Launch Subroutine Symbol Table (Continued)

Math Symbol	Description	FORTRAN Program Symbol
$\underset{\rightarrow}{v}_L$	Unit vector in the launch plane and perpendicular to $\underset{\rightarrow}{r}_L$	RL
$\underset{\rightarrow}{n}_L$	Unit vector normal to launch plane ($\underset{\rightarrow}{n}_L = \underset{\rightarrow}{r}_L \times \underset{\rightarrow}{v}_L$)	NL
ω_E	Rate of the earth's rotation (rad/sec)	WE
ϕ_L	Longitude of launch site at time t relative to target orbit's ascending node	PHIL
ϕ_{L1}	Value of ϕ_L when $t = \tau_1$	PHIL1
ϕ_{L2}	Value of ϕ_L when $t = \tau_2$	PHIL2
A_{L1}	Value of A_L, at $t = \tau_1$, for minimum Δi	AL1
A_{L2}	Value of A_L, at $t = \tau_2$, for minimum Δi	AL2
Δi_1	Value of Δi at $t = \tau_1$	DELI1
Δi_2	Value of Δi at $t = \tau_2$	DELI2

(Sheet 2 of 2)

TABLE I-3
EPHEM (EPHEMERIS SUBROUTINE)
SYMBOL TABLE

Math Symbol	Description	FORTRAN Program Symbol
$\underrightarrow{r}_o$, $\dot{\underrightarrow{r}}_o$	Initial position and velocity (ft. and ft/sec)	RO, RODOT
TO	Initial time (sec)	TO
μ	Gravitational constant	U
θ	Transfer angle (radian)	TH
t	Time of transfer-input for option B (sec)	T
DLE	Convergence criteria	DLE
NOP	= 0 for Option A (angle input) = 1 for Option B (time input)	NOP
$\underrightarrow{r}$, $\dot{\underrightarrow{r}}$	Position and velocity at end of transfer	R, RDOT
tof	Time of flight (Option A only) (sec)	TOF
m_a	Arrival slope	MA
a	Semi-major axis (ft)	A
r_o	Magnitude of $\underrightarrow{r}_o$	ROM
$\underrightarrow{e}_{ro}$	Unit radial vector	ERO
V_{ro}	Initial radial velocity (ft/sec)	VRO
$\underrightarrow{C}$	Moment of momentum (ft^2/sec)	C
C	Magnitude of $\underrightarrow{C}$ (ft^2/sec)	CM
$V_{\theta o}$	Initial horizontal velocity (ft/sec)	VTHO
m_o	Initial slope	XMO
r	Magnitude of $\underrightarrow{r}$	RA
ΔE	Eccentric anomaly (radian)	DLEN

(Sheet 1 of 2)

Table I-3 EPHEM (Ephemeris Subroutine) Symbol Table (Continued)

Math Symbol	Description	FORTRAN Program Symbol
P	Period (sec)	P
V_{ra}	Radial velocity at arrival	VRA
$V_{\theta a}$	Horizontal velocity at arrival	VTHA

TABLE I-4

RVVR (REQUIRED VELOCITY VECTOR SUBROUTINE)

SYMBOL TABLE

Math Symbol	Description	FORTRAN Program Symbol
$\underset{\rightarrow}{r}_o, \underset{\rightarrow}{\dot{r}}_o$	Initial position and velocity (ft. and ft/sec)	RO, RODOT
$\underset{\rightarrow}{r}_a$	Aim point (ft)	RA
m_a	Slope at arrival	XMA
x_{kl}	Constant to determine 180° transfers	XK1
μ	Gravitation constant	U
NN	Number of complete orbits during transfer (input)	NN
r_{Ta}	Magnitude of target radius (Option A-3 only)	RTAM
KK	= -1 for Option B of Rendezvous-Intercept	KK
	0 for Option A-1 of Rendezvous-Intercept	
	1 for Option A-2 of Rendezvous-Intercept	
	2 for Option A-3 of Rendezvous-Intercept	
	3 for Option A-4 of Rendezvous-Intercept	
m_{Po}	Slope at $\underset{\rightarrow}{r}_o$ before velocity change	XPO
$\underset{\rightarrow}{e}_{\theta}^{R}$	Unit horizontal vector	ETHR
α	Plane change angle used in 180° transfers (radian)	ALF
N180	= 1 when 180° transfer has been performed	N180
	= 0 otherwise	
θ_a	Transfer angle (radian)	THA
tof	Time of flight (sec)	TA
$\underset{\rightarrow}{V}_r$	Velocity required for transfer (ft/sec)	VR
NSIG	Indicates type of conic transfer	NSIG
a	Semi-major axis (ft)	A, AX

(Sheet 1 of 2)

Table I-4 RVVR (Required Velocity Vector Subroutine) Symbol Table (Continued)

Math Symbol	Description	FORTRAN Program Symbol
r_o	Magnitude of $\underrightarrow{r}_o$ (ft)	ROM
r_a	Magnitude of $\underrightarrow{r}_a$ (ft)	RAM
$\underrightarrow{e}_r$	Unit radial vector	ER
$\underrightarrow{r}_{ar}$	Radial component of $\underrightarrow{r}_a$ (ft)	RAR
$\underrightarrow{r}_{a\theta}$	Horizontal component of $\underrightarrow{r}_o$ (ft)	RATH
$r_{a\theta}$	Magnitude of $\underrightarrow{r}_{a\theta}$ (ft)	RATHM
$\underrightarrow{V}_\theta$	Horizontal component of $\dot{\underrightarrow{r}}_o$ (ft/sec)	VTH
V_θ	Magnitude of $\underrightarrow{V}_\theta$ (ft/sec)	VTHM

(Sheet 2 of 2)

TABLE I-5

TIME SUBROUTINE SYMBOL TABLE

Math Symbol	Description	FORTRAN Program Symbol
V_θ^R	Horizontal component of velocity (ft/sec)	VTHR
V_r^R	Radial component of velocity (ft/sec)	VRR
μ	Gravitational constant	U
r_o	Magnitude of position vector (ft)	ROM
θ_a	Transfer angle (radian)	
$\cos \theta a$		CTHA
$\sin \theta a$		STHA
m_a	Slope at arrival	XMA
r_a	Magnitude of arrival point radius (ft)	RAM
NN	Number of complete orbits	NN
tof	Time of flight (sec)	TA
C	Magnitude of angular momentum	CA
a	Semi-major axis (ft)	A
ΔE	Eccentric anomaly (radian)	DE

TABLE I-6

SIMTIM SUBROUTINE SYMBOL TABLE

Math Symbol	Description	FORTRAN Program Symbol
θ	Transfer angle (radian)	TH
m_o	Departure slope	XMO
r_o	Departure radius (ft)	ROM
C	Magnitude of angular momentum	H
m_a	Arrival slope	MA
μ	Gravitational constant	U
tof	Time of flight (sec)	TOF, TOFN

TABLE I-7

RENDI - RENDEZVOUS-INTERCEPT SUBROUTINE

SYMBOL TABLE

Math Symbol	Description	FORTRAN Program Symbol
$\underset{\rightarrow}{r}_i$	Initial interceptor position (ft)	RI
$\dot{\underset{\rightarrow}{r}}_i$	Initial velocity of interceptor (ft/sec)	RIDOT
$\underset{\rightarrow}{r}_{Ti}$	Initial target position (ft)	RTI
$\dot{\underset{\rightarrow}{r}}_{Ti}$	Initial velocity of target (ft/sec)	RTIDOT
TSTOP	Maximum time at which first burn can occur (sec)	TSTOP
m_a	Arrival slope	MA
k_1	= -1 for RENDI Option B = 0 for RENDI Option A-1 = 1 for RENDI Option A-2 = 2 for RENDI Option A-3 = 3 for RENDI Option A-4	K1
NOIT	= 0 return when t + TSTART > TSTOP = 1 return after first minimum ΔV is attained or t + TSTART > TSTOP = -1 return when $\Delta V < \Delta V_M$ or t + TSTART > TSTOP	NOIT
TSTART	Reference time when $\underset{\rightarrow}{r}_i$, $\dot{\underset{\rightarrow}{r}}_i$, $\underset{\rightarrow}{r}_{Ti}$, and $\dot{\underset{\rightarrow}{r}}_{Ti}$ are given (sec)	TSTART
ΔV_M	Maximum allowable ΔV for trajectory (ft/sec)	DELVM
μ	Gravitational constant	U
ΔT	Amount by which the time of first burn is incremented in search for minimum ΔV (sec)	DELT

(Sheet 1 of 3)

Table I-7 RENDI - Rendezvous-Intercept Subroutine Symbol Table (Continued)

Math Symbol	Description	FORTRAN Program Symbol
t	Time of first burn relative to TSTART (sec)	T
$\underrightarrow{r}_o$, $\dot{\underrightarrow{r}}_o$	Interceptor position and velocity at time of first burn	RO, RODOT
$\underrightarrow{r}_T$, $\dot{\underrightarrow{r}}_T$	Target position and velocity at time of first burn	RT, RTDOT
$\underrightarrow{r}_{inp}$	Projection of $\underrightarrow{r}_o$ onto target plane (ft)	RINP
θ	Angle between $\underrightarrow{r}_{inp}$ and $\underrightarrow{r}_T$ (radian)	THH
DLIM	Lower limit for transfer angle (radian)	DLIM
ULIM	Upper limit for transfer angle (radian)	ULIM
P	Period of interceptor (sec)	P
θ_0	Past guess at transfer angle (radian)	TH0
θ_1	Present guess at transfer angle (radian)	TH1
$\underrightarrow{r}_{TA}$, $\dot{\underrightarrow{r}}_{TA}$	Target position and velocity after being transferred in orbit through angle θ_1 (or θ_0) from $\underrightarrow{r}_T$	RTA, RTADOT
m_{TA}	Target slope at $\underrightarrow{r}_{TA}$	MTA
T_{TA0} or T_{TA1}	Time of target transfer from $\underrightarrow{r}_T$ to $\underrightarrow{r}_{TA}$ resulting from transfer angle of θ_0 or θ_1 (sec)	TTA0 or TTA1
$\underrightarrow{r}_a$	Aim point (ft)	RA
m_a	Desired interceptor arrival slope	MA
T_{A0} or T_{A1}	Time of interceptor transfer from $\underrightarrow{r}_o$ to $\underrightarrow{r}_a$ corresponding to target transfer angle θ_0 or θ_1 (sec)	TA0 or TA1
VALUE	The function which is reduced to zero by regula falsi iterations (sec)	VALUE

(Sheet 2 of 3)

Table I-7 RENDI - Rendezvous-Intercept Subroutine Symbol Table (Continued)

Math Symbol	Description	FORTRAN Program Symbol
DERIV	Approximated derivative of VALUE	DERIV
θ_2	New value of θ_1 (radian)	TH2
$\underset{\rightarrow}{V}_r$	Velocity required by interceptor at $\underset{\rightarrow}{r}_o$ to transfer to $\underset{\rightarrow}{r}_a$ (ft/sec)	VR
T_A	Fixed transfer time (sec)	TA
m_{A0}	Past guess at arrival slope	MA0
m_{A1}	Present guess at arrival slope	MA1
m_{A2}	New guess at arrival slope	MA2
$\dot{\underset{\rightarrow}{r}}_a$	Arrival velocity of interceptor (ft/sec)	RADCT
ΔV	Total change in velocity (ft/sec)	DELV

(Sheet 3 of 3)

TABLE I-8

BEST-SUBROUTINE SYMBOL TABLE

Math Symbol	Description	FORTRAN Program Symbol
DV	Velocity change (ΔV)	DV
T	Time of first burn	T
TH	Transfer angle	TH
IPL	Counter to tell when to apply quadratic fit	IPL
DT	Time increment equal to 1/4 the interceptor or target period, whichever is smaller	DT
TMINSA	T at last minimum found	TMINSA
NOPT	= 1 when minimum has been found = 0 otherwise	NOPT
δT	Time increment = DT/10	SDT

TM(I)
I = 1, 4
	Storage for first burn times	TN(I)

DVEL(I)
I = 1, 4
	Storage for ΔV's corresponding to TM(I)	DVEL(I)

τ_I I = 1, 4	Normalized values of TM(I)	TM(I)
τ	Normalized value of time indicated by quadratic to give minimum ΔV	

TABLE I-9

VPG - VARIABLE POINT GUIDANCE SUBROUTINE

SYMBOL TABLE

Math Symbol	Description	FORTRAN Program Symbol
	I. Inputs	
a_T	Target semi-major axis (ft)	AT
e_T	Target eccentricity	ET
f_{LN}	Angular Separation of Line-of-Nodes and Target's Line-of-Apsides (deg)	FLN
Δf_3	Position of target relative to Line-of-Nodes at t_3 (deg)	DELF3
t_3	Time of spacecraft's first nodal crossing (sec)	T3
R_p	Radius of spacecraft's circular parking orbit (ft)	RP
Δi	Out-of-plane angle between spacecraft (S/C) and target orbits	DELI
$\underset{\to}{n}_2$	Unit vector normal to S/C parking orbit	EN2
$\underset{\to}{n}_N$	Unit normal along Line-of-Nodes (defines Node A, Node B defined by $-\underset{\to}{n}_N$)	ENN
μ	Gravitation constant	U
	II. Internal Symbols	
ΔR_{max}	Maximum altitude above target orbit allowed for intermediate aim point in Bielliptic option (ft)	DELRM
R^*_{min}	Minimum radius allowed for intermediate aim point in Bielliptic option (ft)	RMINS
$\Delta R_o{}^*$	Initial altitude increment, added to parking orbit radius, used for intermediate aim point in Bielliptic option (ft)	DELROS

(Sheet 1 of 4)

Table I-9 VPG - Variable Point Guidance Subroutine Symbol Table

(Continued)

Math Symbol	Description	FORTRAN Program Symbols
ϵ_α	Error tolerance used to stop iterations in optimum plane change angle routine (rad)	EPA
$S_{\Delta i}$	Sign of the Δi input to subroutine	SDELI
$\dot{\omega}_P$	Angular rate of S/C in its orbit (rad/sec)	WP
P_P	Period of S/C orbit (sec)	PP
V_P	Velocity in circular parking orbit (ft/sec)	VP
P_T	Period of target orbit (sec)	PT
h_T	Angular momentum of target orbit (ft^2/sec)	HT
R_{TA}	Target radius at Node A (ft), see $\underset{\rightarrow}{n}_N$	RTA
R_{TB}	Target radius at Node B (ft), see $\underset{\rightarrow}{n}_N$	RTB
R_{T3}	Target radius at $t = t_3$	RT3
V_{TA}	Target's velocity at Node A (ft/sec)	VTA
V_{TB}	Target's velocity at Node B (ft/sec)	VTB
V_{T3}	Target's velocity at $t = t_3$	VT3
γ_{TA}	Target's flight path angle at Node A (rad)	GAMTA
γ_{TB}	Target's flight path angle at Node B (rad)	GAMTB
R_i $i = 3, 7$	S/C radius at $t = t_i$, $i = 3, \cdots, 7$	R3, ···, R7
V_i^- $i = 3, 7$	S/C velocity at $t = t_i^-$, $i = 3, \cdots, 7$	V3M, ···, V7M and VM(I) I = 3, 7
V_i^+, $i = 3, 7$	S/C velocity at $t = t_i^+$, $i = 3, \cdots, 7$	V3PL, ···, V7PL and VPL(I) I = 3,
R^*_{max}	Maximum radius allowed for intermediate aim point in Bielliptic option	RMAXS

Table I-9 VPG - Variable Point Guidance Subroutine Symbol Table (Continued)

Math Symbol	Description	FORTRAN Program Symbol
ΔT	Time of flight (sec)	DELT
T_N	Time for target to reach rendezvous point for the first time (sec)	TN
R^*	Radius of intermediate aim point in Bielliptic option (ft)	RS
m^*	Slope of S/C orbit at R*	MS
m_6	Slope of target orbit at rendezvous point	M6
V_r	Radial component of required velocity (ft/sec)	VR
V_θ	Tangential component of required velocity (ft/sec)	VTH
γ_i i = 3, 7	Flight path angle of S/C orbit at $t = t_i$, i = 3, 7	GAM1, ..., and GAM(I) I = 3, 7
T_{T1}	Transfer time to intermediate aim point (sec)	TT1
$\Delta\theta$	Position of first orbital burn ($t = t_4$) relative to line-of-nodes (rad)	DELTH
γ^*	Flight path angle at R*	GAMS
T_{T2}	Transfer time on second arc in Bielliptic option (sec)	TT2
T_T	Total transfer time, relative to t_3, for S/C excluding integral number of phasing orbits	TT
X'	Non-integer number of full parking orbits (for phasing) before first burn	XP
ΔR^*	Increment to R* to correct phasing error (ft)	DELRS
P_{PE0}	Initial value of P_{PE} (sec)	PPEO
P_{PE}	Period of phasing ellipse in Full Orbit Phasing option	PPE

(Sheet 3 of 4)

Table I-9 VPG - Variable Point Guidance Subroutine Symbol Table (Continued)

Math Symbol	Description	FORTRAN Program Symbol
a_{PE}	Semi-major axis of phasing ellipse	APE
α_i i = 3, 7	Plane change angle at $t = t_i$ i = 3, ···, 7	ALFA3, ··· and AF(I) I = 3,7
ΔV_i i = 3, 7	Velocity impulse at $t = t_i$ i = 3, ···, 7	DV3, ···, DV7 and DELV(I) I =3,7
$\underrightarrow{n}_i$ i = 2, 7	Unit normal to plane defined by $\underrightarrow{R}_i$ and $\underrightarrow{V}_i$	EN(,I), I = 2,7
$\underrightarrow{e}_i$ i = 3, 6	Unit vector along position vector at $t = t_i$ i = 3, ···, 6	EVEC (,I), I=3,6
$\underrightarrow{R}_i$ i = 3, 6	Position vector at t_i i = 3, ···, 6	RVEC (,I), I=3,6
$\underrightarrow{e}_{Vi}$ i = 3, 6	Unit vector normal to $\underrightarrow{R}_i$ and $\underrightarrow{n}_i$	EVC (,I), I=3,6
$\underrightarrow{V}_i$, i = 3, 6	Velocity vector at $\underrightarrow{R}_i$	VVEC (,I), I=3,6
$\Delta \underrightarrow{V}_i$, i = 3, 6	Total velocity impulse at t_i i = 3, ···, 6	DLVL (,I), I = 3, 6
I	Integer, full orbits in parking orbit	I
J	Integer full orbits in phasing orbit (Full Orbit Phasing option)	J
K	Integer, full orbits of target before rendezvous	K

(Sheet 4 of 4)

TABLE I-10

PCA - PLANE CHANGE ANGLE SUBROUTINE

SYMBOL TABLE

Math Symbol	Description	FORTRAN Program Symbol
J, K, L	Controls for indices on various vectors	J, K, L
NM	Iteration Counter	NM
Δi	Total plane change angle	DELI
$V^{-I}(I)$	Velocity magnitude before I^th burn	VM (I)
$V^{+}(I)$	Velocity magnitude after I^th burn	VPL (I)
$\gamma(I)$	Flight path angle during I^th burn	GAM (I)
$\alpha(I)$	Plane change angle at I^th burn	ALFA (I)
$\Delta V_o(I)$	Impulse at I^th burn with no plane change	DELVO (I)
$\Delta V(I)$	Total impulse at I^th burn	DELV (I)

TABLE I-11

REQ SUBROUTINE
SYMBOL TABLE

Math Symbol	Description	FORTRAN Program Symbol
r_1	Radius of vehicle position (ft)	R1
r_2	Radius of aim point (ft)	R2
θ	Angle between vectors to vehicle position and aim point (radian)	TH
m	Slope at arrival at aim point	M
μ	Gravitational constant	U
V_θ	Horizontal component of required velocity (ft /sec)	VTH
V_r	Radial component of required velocity (ft/sec)	VR

TABLE I-12

TOF SUBROUTINE
SYMBOL TABLE

Math Symbol	Description	FORTRAN Program Symbol
r_1	Radius of vehicle position (ft)	R1
r_2	Radius of aim point (ft)	R2
$\Delta\theta$	Angle between vectors to vehicle position and aim point (radian)	DELTH
V_1	Velocity of vehicle (ft/sec)	V1
μ	Gravitational constant	U
γ_1	Slope of vehicle	GAM1
ΔT	Time of flight	DELT
h	Magnitude of angular momentum	H
a	Semi-major axis (ft)	A
m	Slope at arrival at aim point	XM
ΔE	Eccentric anomaly (radian)	DE

Table I-13

TEST CASES

Case	Initial Conditions	Final Orbit	Comments
1	Launch Latitude: λ_L = 28.5617° Launch Longitude: Ω_{LO} = 0 Reference Time: t_o = 0 Launch Azimuth Limits: $65^o \leq A_L \leq 115^o$	Semi-Major Axis: a_T = 138,607,380 ft. Eccentricity: $e_T = 0$ Inclination: $i_T = 0$	Rendezvous with fictitious target in synchronous, equatorial orbit at launch site longitude.
2	Same as Case 1	Semi-Major Axis: a_T = 22,060,185 ft. Eccentricity: $e_T = 0$ Inclination: $i_T = 32^o$ Longitude of Ascending Node: $\Omega_T = 0$	Rendezvous with vehicle in 185 n.m. circular parking orbit.
3	Same as Case 1 but no parking orbit used, and $\Omega_{LO} = 80^o$	Same as Case 2	Direction Ascent Rendezvous to target of Case 2

Table I-14

RESULTS OF TEST CASES

Case	Launch			Ascent			Orbital Maneuvers								
							RENDI				VPG				
	Launch Time (sec)	Launch Azimuth (deg)	Out-Of-Plane Angle With Target Orbit (deg)	Cutoff Time (sec)	Altitude At Cutoff (Parking Orbit for Cases 1 and 2) (n. m.)	ΔV Available for Orbital Burns (ft/sec)	Time of 1st Burn (sec)	Time of Last Burn (sec)	ΔV for Orbital Burns (ft/sec)	Note (See Comments)	Time of First Burn (sec)	Time of Second Burn (sec)	Time of Last Burn (sec)	ΔV for Orbital Burns (ft/sec)	Note (See Comments)
1	0	90	28.56	483	100	14,444	4032	25050	17697	(1)	38553	58382	102716	14138	(3)
							43488	75863	15489	(2)	6846	25810	64800	14032	(4)
2	14502	74.92	0	15089	100	8762					19103	27383	30125	6057	(5)
3	(6) 19260	84.84	3.05	19764	276	10127	(7) 21372	--	(7) 7378						

RENDI

(1) First Min- ΔV Found

(2) Best Min- ΔV Found

VPG

(3) Bielliptic Lob

(4) Full Orbit Phasing Chase

(5) VPG Full Orbit Phasing Lob

(6) Launch site is 80.25 degrees from Target's Ascending Node at Launch

(7) One Orbital Burn At Target Altitude to Match Velocities.

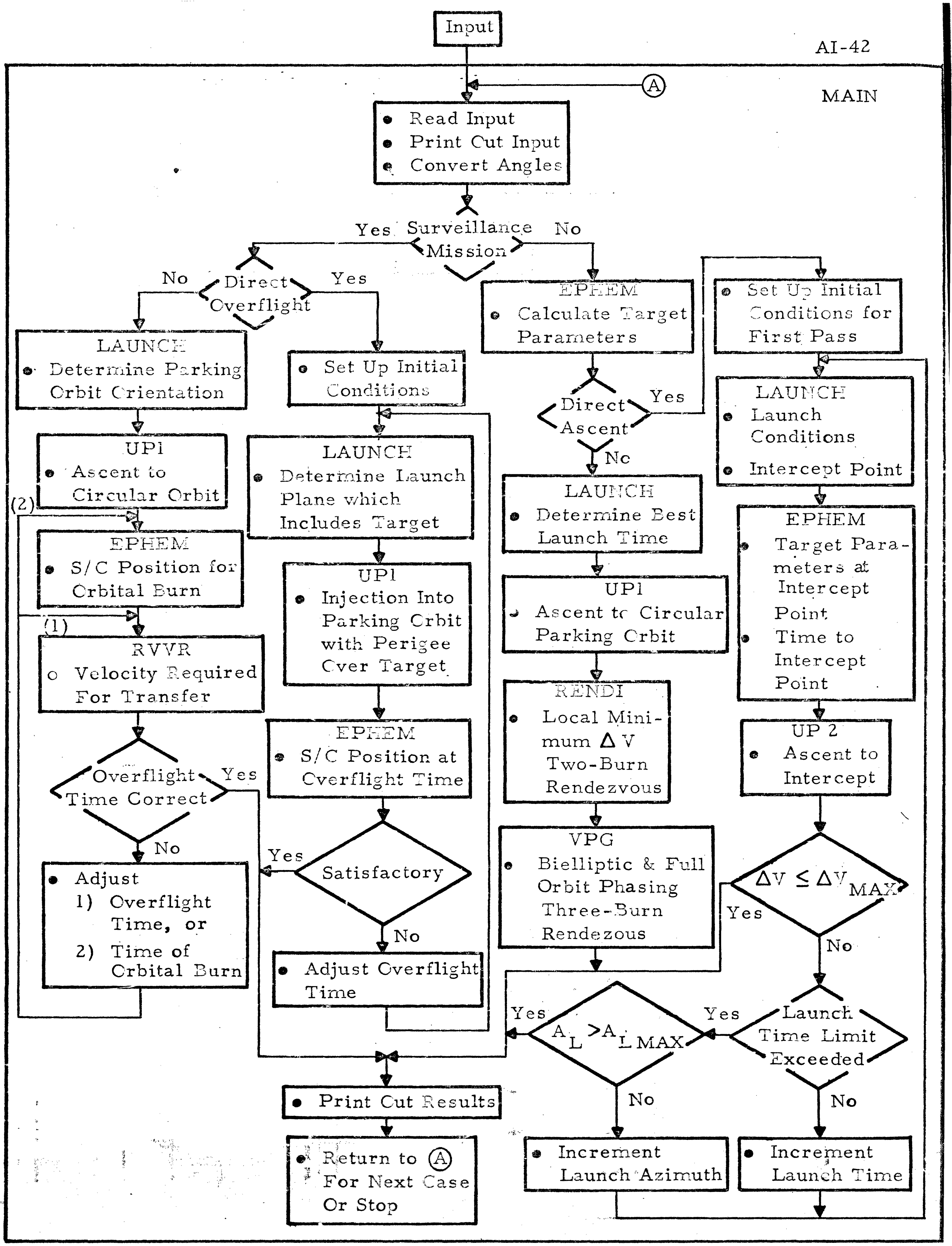
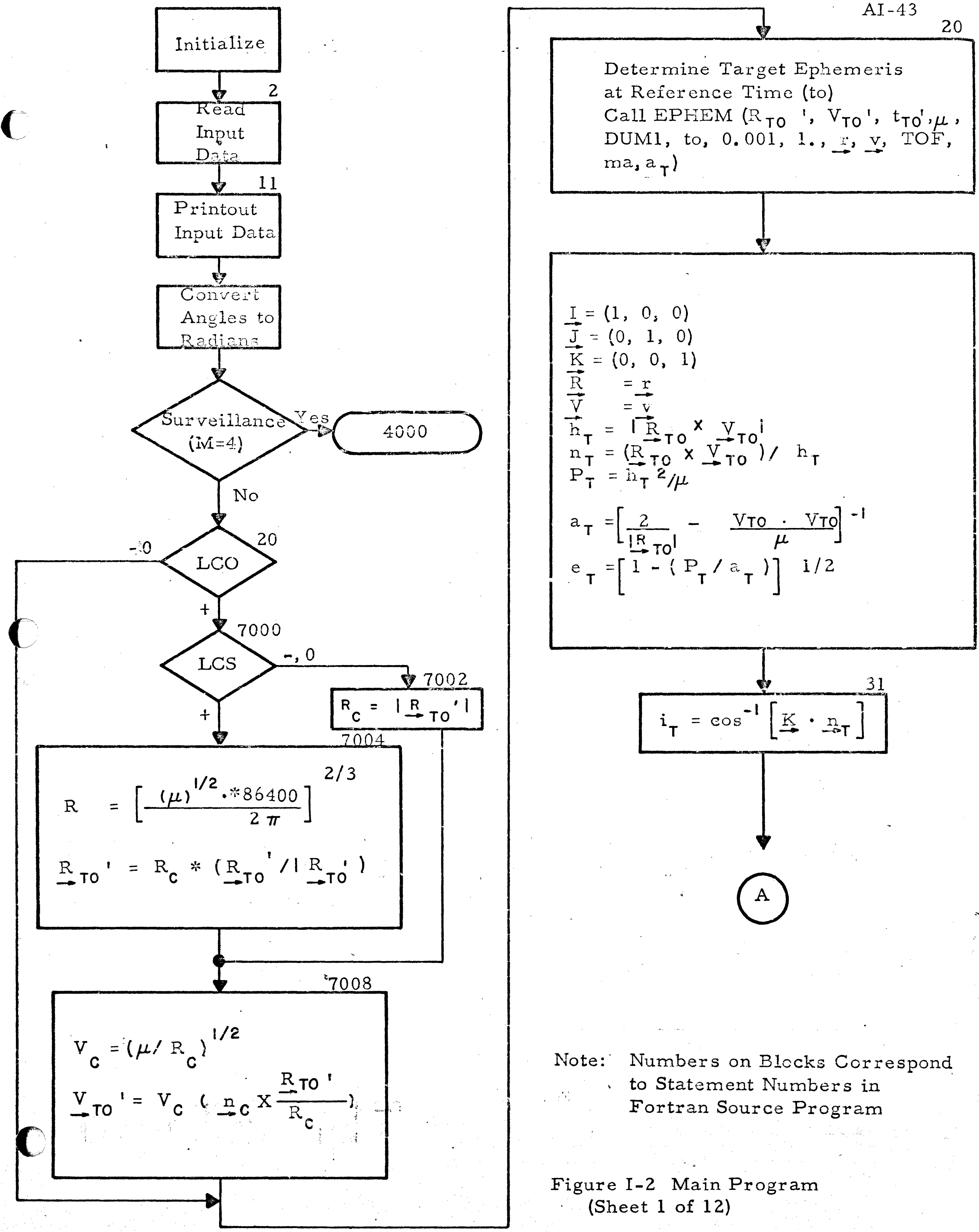

Input
MAIN
A
Read Input
Print Cut Input
Convert Angles
Yes
Surveillance Mission
No
No
Direct Overflight
Yes
LAUNCH
Determine Parking Orbit Crientation
Set Up Initial Conditions
UP1
Ascent to Circular Orbit
(2)
LAUNCH
Determine Launch Plane which Includes Target
EPHEM
S/C Position for Crbital Burn
UP1
Injection Into Parking Crbit with Perigee Cver Target
(1)
RVVR
Velocity Required For Transfer
EPHEM
S/C Position at Cverflight Time
Overflight Time Correct
Yes
No
Satisfactory
Yes
No
Adjust
1) Overflight Time, or
2) Time of Crbital Burn
Adjust Overflight Time
EPHEM
Calculate Target Parameters
Direct Ascent
Yes
Nc
LAUNCH
Determine Best Launch Time
UP1
Ascent tc Circular Parking Crbit
RENDI
Local Minimum Δ V Two-Burn Rendezvous
VPG
Bielliptic & Full Orbit Phasing Three-Burn Rendezous
Set Up Initial Conditions for First Pass
LAUNCH
Launch Conditions
Intercept Point
EPHEM
Target Parameters at Intercept Point
Time to Intercept Point
UP 2
Ascent to Intercept
ΔV ≤ ΔV MAX
Yes
No
Yes
A L >A L MAX
Yes
Launch Time Limit Exceeded
No
No
Print Cut Results
Return to A For Next Case Or Stop
Increment Launch Azimuth
Increment Launch Time

Figure I-1 Trajectory Generator Program


Initialize
2
Read Input Data
11
Printout Input Data
Convert Angles to Radians
Surveillance (M=4)
Yes
4000
No
20
LCO
-,0
+
7000
LCS
-, 0
+
7002
$R_C = |\vec{R}_{TO}'|$
7004
$R = \left[\frac{(\mu)^{1/2} \cdot * 86400}{2\pi}\right]^{2/3}$
$\vec{R}_{TO}' = R_C * (\vec{R}_{TO}' / |\vec{R}_{TO}'|)$
7008
$V_C = (\mu / R_C)^{1/2}$
$\vec{V}_{TO}' = V_C \left(\vec{n}_C X \frac{R_{TO}'}{R_C} \right)$
20
Determine Target Ephemeris at Reference Time (to)
Call EPHEM (R_{TO} ', V_{TO}', t_{TO}', μ, DUM1, to, 0.001, 1., $\vec{r}$, $\vec{v}$, TOF, ma, a_T)
$\vec{I} = (1, 0, 0)$
$\vec{J} = (0, 1, 0)$
$\vec{K} = (0, 0, 1)$
$\vec{R} = \vec{r}$
$\vec{V} = \vec{v}$
$h_T = |\vec{R}_{TO} \times \vec{V}_{TO}|$
$\vec{n}_T = (\vec{R}_{TO} \times \vec{V}_{TO}) / h_T$
$P_T = h_T^{\ 2} / \mu$
$a_T = \left[\frac{2}{|\vec{R}_{TO}|} - \frac{V_{TO} \cdot V_{TO}}{\mu} \right]^{-1}$
$e_T = \left[1 - (P_T / a_T) \right]^{1/2}$
31
$i_T = \cos^{-1} \left[\vec{K} \cdot \vec{n}_T \right]$
A

Note: Numbers on Blocks Correspond to Statement Numbers in Fortran Source Program

Figure I-2 Main Program
(Sheet 1 of 12)

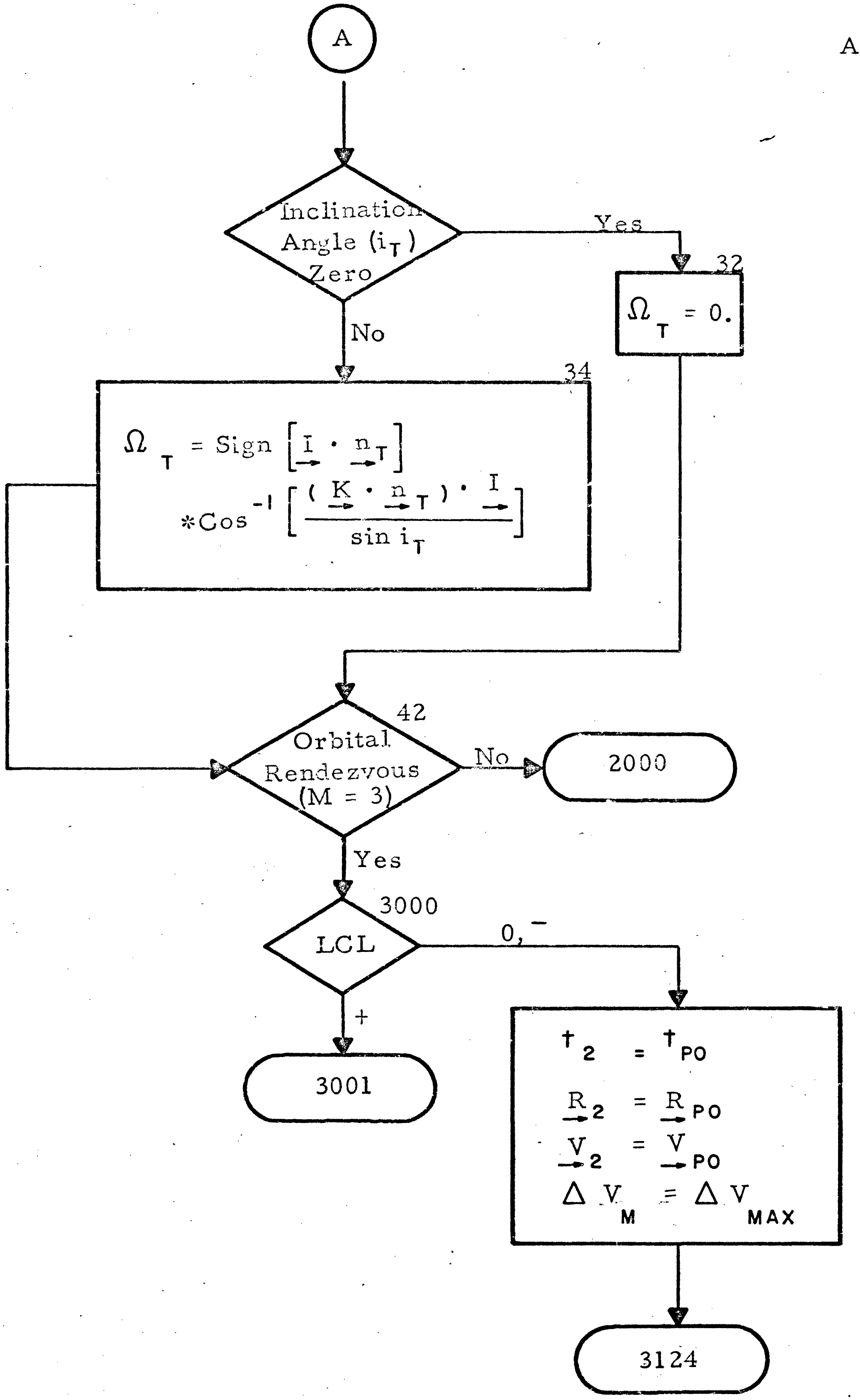

A
Inclination Angle (i_T) Zero
Yes
32
$\Omega_T = 0.$
No
34
$\Omega_T = \text{Sign}\left[\underset{\rightarrow}{I} \cdot \underset{\rightarrow}{n}_T\right] * \text{Cos}^{-1}\left[\frac{(\underset{\rightarrow}{K} \cdot \underset{\rightarrow}{n}_T) \cdot \underset{\rightarrow}{I}}{\sin i_T}\right]$
42
Orbital Rendezvous (M = 3)
No
2000
Yes
3000
LCL
0, −
+
3001
$t_2 = t_{PO}$
$\underset{\rightarrow}{R}_2 = \underset{\rightarrow}{R}_{PO}$
$\underset{\rightarrow}{V}_2 = \underset{\rightarrow}{V}_{PO}$
$\Delta V_M = \Delta V_{MAX}$
3124

Figure I-2 Main Program
(Sheet 2 of 12)

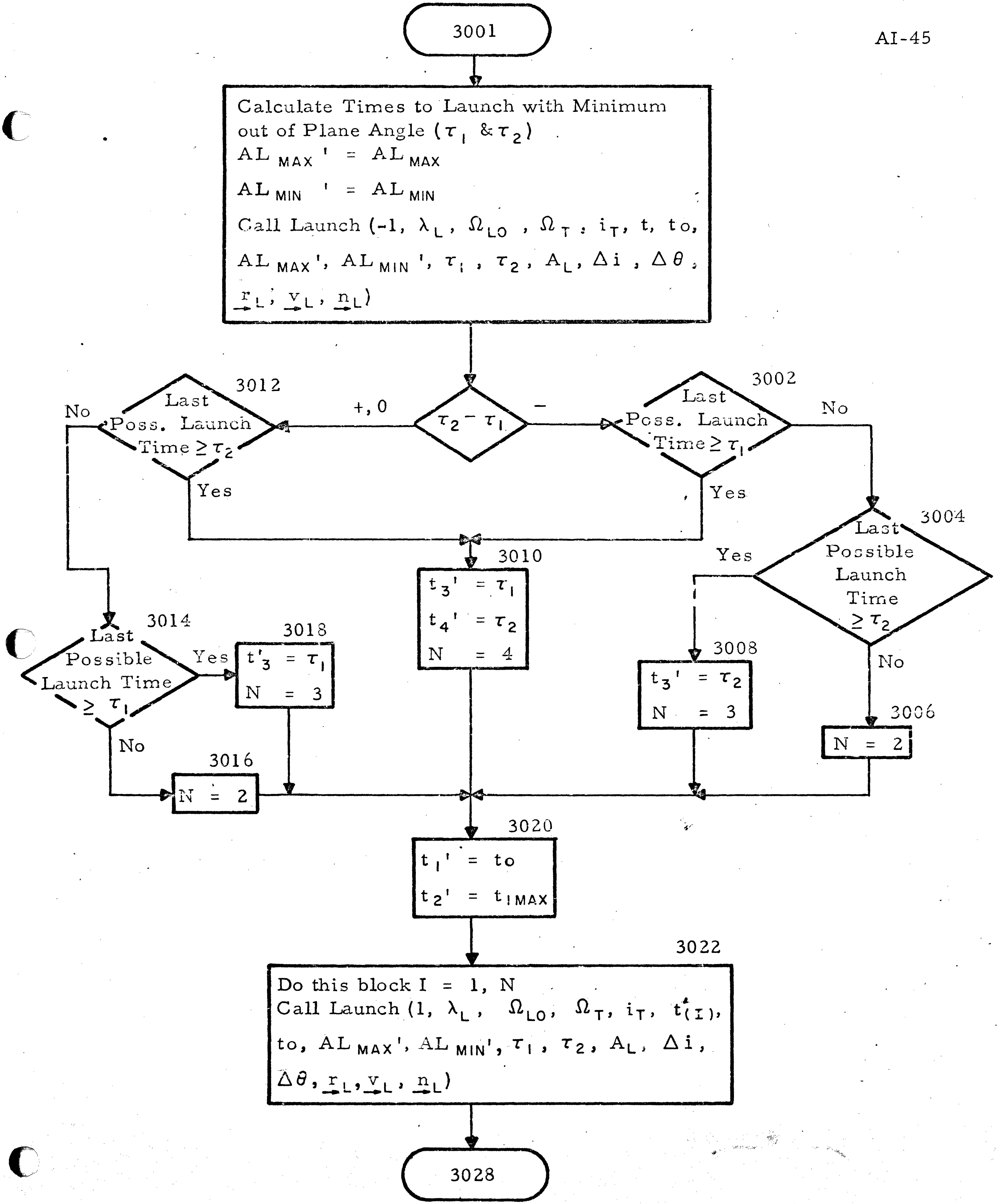

3001
Calculate Times to Launch with Minimum out of Plane Angle (τ_1 & τ_2)
$AL_{MAX}' = AL_{MAX}$
$AL_{MIN}' = AL_{MIN}$
Call Launch (-1, λ_L, Ω_{LO}, Ω_T : i_T, t, to, AL_{MAX}', AL_{MIN}', τ_1, τ_2, A_L, Δi, $\Delta\theta$, $\underline{r}_L$, $\underline{v}_L$, $\underline{n}_L$)
$\tau_2 - \tau_1$
+, 0
–
3012
Last Poss. Launch Time $\geq \tau_2$
No
Yes
3002
Last Poss. Launch Time $\geq \tau_1$
No
Yes
3004
Last Possible Launch Time $\geq \tau_2$
Yes
No
3010
$t_3' = \tau_1$
$t_4' = \tau_2$
N = 4
3014
Last Possible Launch Time $\geq \tau_1$
Yes
No
3018
$t'_3 = \tau_1$
N = 3
3016
N = 2
3008
$t_3' = \tau_2$
N = 3
3006
N = 2
3020
$t_1' = to$
$t_2' = t_{IMAX}$
3022
Do this block I = 1, N
Call Launch (1, λ_L, Ω_{LO}, Ω_T, i_T, $t_{(I)}$, to, AL_{MAX}', AL_{MIN}', τ_1, τ_2, A_L, Δi, $\Delta\theta$, $\underline{r}_L$, $\underline{v}_L$, $\underline{n}_L$)
3028

Figure I-2 Main Program
(Sheet 3 of 12)

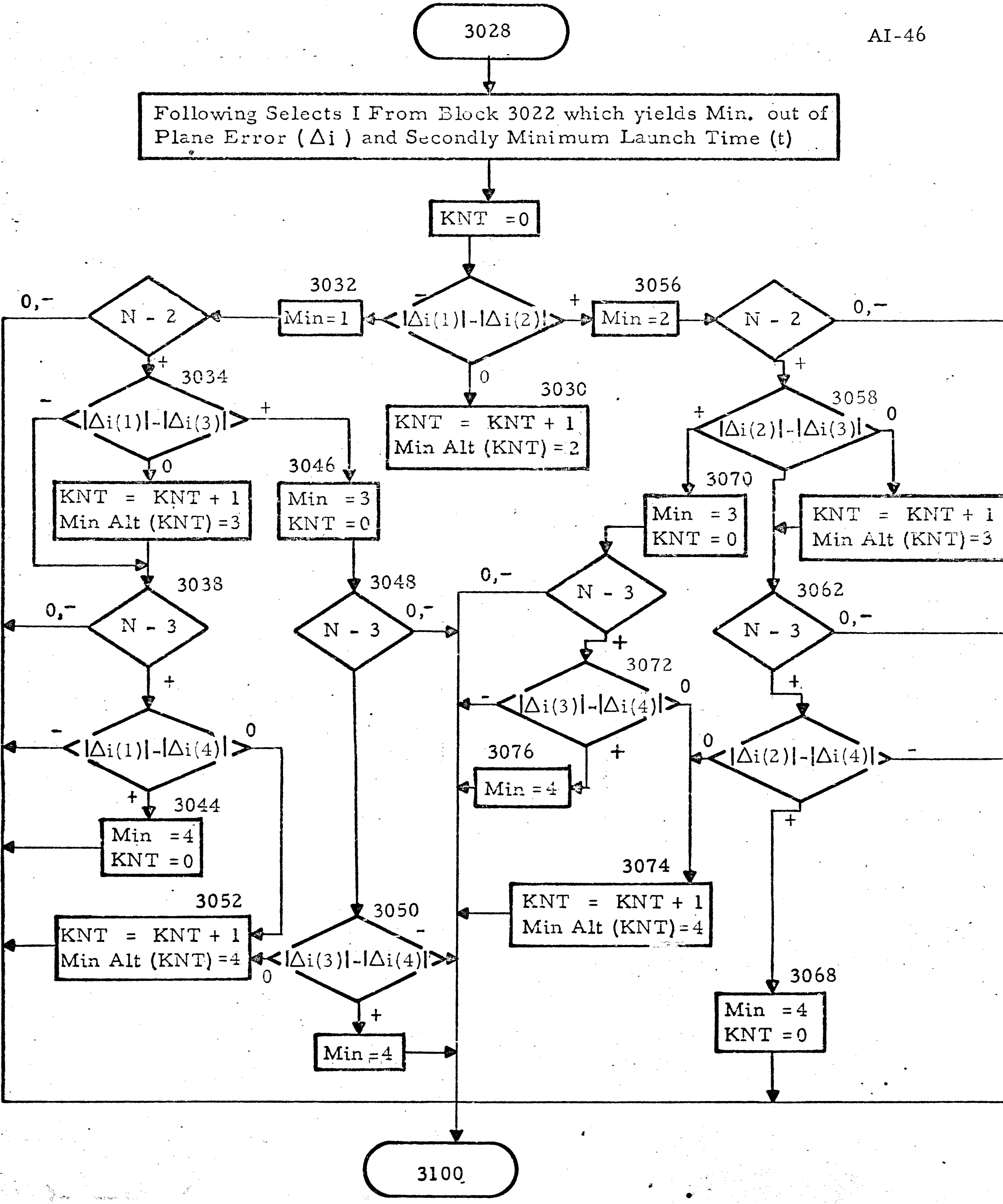


Figure I-2 Main Program
(Sheet 4 of 12)

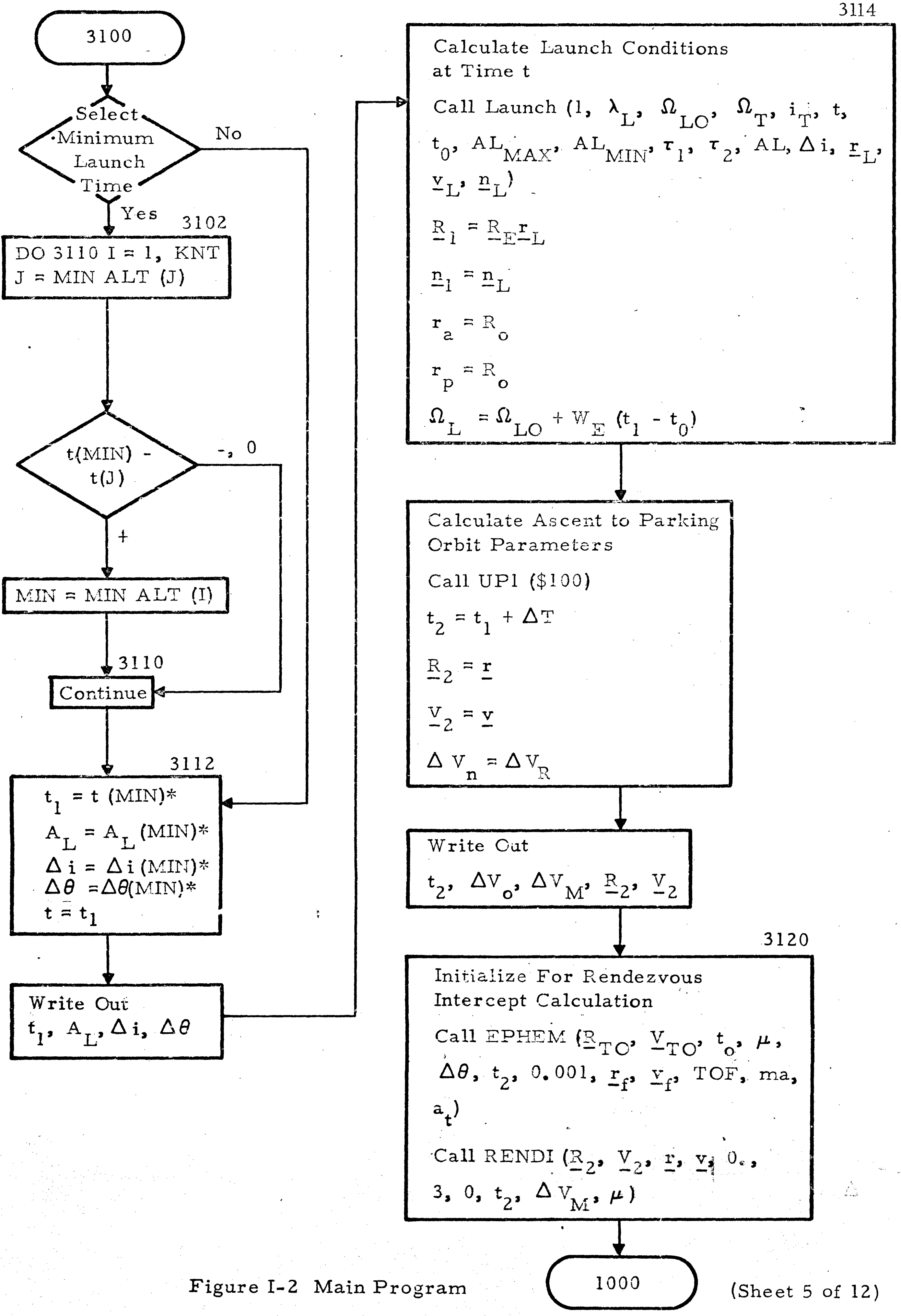

3100
Select Minimum Launch Time
No
Yes
3102
DO 3110 I = 1, KNT
J = MIN ALT (J)
t(MIN) - t(J)
-, 0
+
MIN = MIN ALT (I)
3110
Continue
3112
$t_1 = t\ (MIN)*$
$A_L = A_L\ (MIN)*$
$\Delta i = \Delta i\ (MIN)*$
$\Delta\theta = \Delta\theta(MIN)*$
$t = t_1$
Write Out
$t_1, A_L, \Delta i, \Delta\theta$
3114
Calculate Launch Conditions at Time t
Call Launch (1, λ_L, Ω_{LO}, Ω_T, i_T, t, t_0, AL_{MAX}, AL_{MIN}, τ_1, τ_2, AL, Δi, $\underline{r}_L$, $\underline{v}_L$, $\underline{n}_L$)
$\underline{R}_1 = R_E \underline{r}_L$
$\underline{n}_1 = \underline{n}_L$
$r_a = R_o$
$r_p = R_o$
$\Omega_L = \Omega_{LO} + W_E\ (t_1 - t_0)$
Calculate Ascent to Parking Orbit Parameters
Call UP1 ($100)
$t_2 = t_1 + \Delta T$
$\underline{R}_2 = \underline{r}$
$\underline{V}_2 = \underline{v}$
$\Delta V_n = \Delta V_R$
Write Out
t_2, ΔV_o, ΔV_M, $\underline{R}_2$, $\underline{V}_2$
3120
Initialize For Rendezvous Intercept Calculation
Call EPHEM ($\underline{R}_{TO}$, $\underline{V}_{TO}$, t_o, μ, $\Delta\theta$, t_2, 0.001, $\underline{r}_f$, $\underline{v}_f$, TOF, ma, a_t)
Call RENDI ($\underline{R}_2$, $\underline{V}_2$, $\underline{r}$, $\underline{v}$, 0., 3, 0, t_2, ΔV_M, μ)
1000

Figure I-2 Main Program (Sheet 5 of 12)

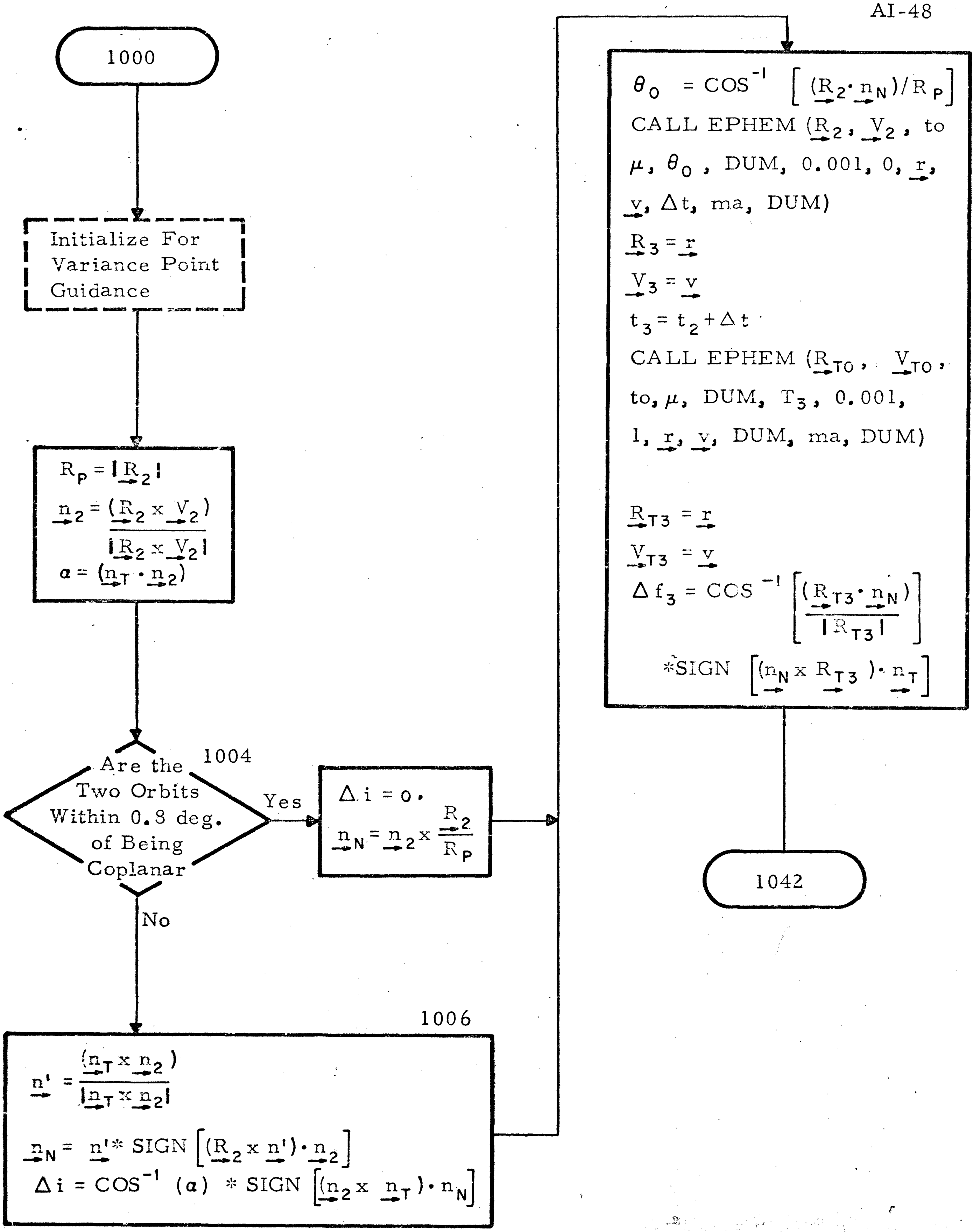

1000
Initialize For Variance Point Guidance
$R_P = |\underrightarrow{R}_2|$
$\underrightarrow{n}_2 = \frac{(\underrightarrow{R}_2 \times \underrightarrow{V}_2)}{|\underrightarrow{R}_2 \times \underrightarrow{V}_2|}$
$\alpha = (\underrightarrow{n}_T \cdot \underrightarrow{n}_2)$
1004
Are the Two Orbits Within 0.8 deg. of Being Coplanar
Yes
$\Delta i = 0.$
$\underrightarrow{n}_N = \underrightarrow{n}_2 \times \frac{\underrightarrow{R}_2}{R_P}$
No
1006
$\underrightarrow{n}' = \frac{(\underrightarrow{n}_T \times \underrightarrow{n}_2)}{|\underrightarrow{n}_T \times \underrightarrow{n}_2|}$
$\underrightarrow{n}_N = \underrightarrow{n}' * \text{SIGN}\left[(\underrightarrow{R}_2 \times \underrightarrow{n}') \cdot \underrightarrow{n}_2\right]$
$\Delta i = \text{COS}^{-1}\ (\alpha)\ *\ \text{SIGN}\left[(\underrightarrow{n}_2 \times \underrightarrow{n}_T) \cdot n_N\right]$
$\theta_0 = \text{COS}^{-1}\left[(\underrightarrow{R}_2 \cdot \underrightarrow{n}_N)/R_P\right]$
CALL EPHEM ($\underrightarrow{R}_2$, $\underrightarrow{V}_2$, to μ, θ_0, DUM, 0.001, 0, $\underrightarrow{r}$, $\underrightarrow{v}$, Δt, ma, DUM)
$\underrightarrow{R}_3 = \underrightarrow{r}$
$\underrightarrow{V}_3 = \underrightarrow{v}$
$t_3 = t_2 + \Delta t$
CALL EPHEM ($\underrightarrow{R}_{TO}$, $\underrightarrow{V}_{TO}$, to, μ, DUM, T_3, 0.001, 1, $\underrightarrow{r}$, $\underrightarrow{v}$, DUM, ma, DUM)
$\underrightarrow{R}_{T3} = \underrightarrow{r}$
$\underrightarrow{V}_{T3} = \underrightarrow{v}$
$\Delta f_3 = \text{COS}^{-1}\left[\frac{(\underrightarrow{R}_{T3} \cdot \underrightarrow{n}_N)}{|R_{T3}|}\right]$
$*\text{SIGN}\left[(\underrightarrow{n}_N \times \underrightarrow{R}_{T3}) \cdot \underrightarrow{n}_T\right]$
1042

Figure I-2 Main Program
(Sheet 6 of 12)

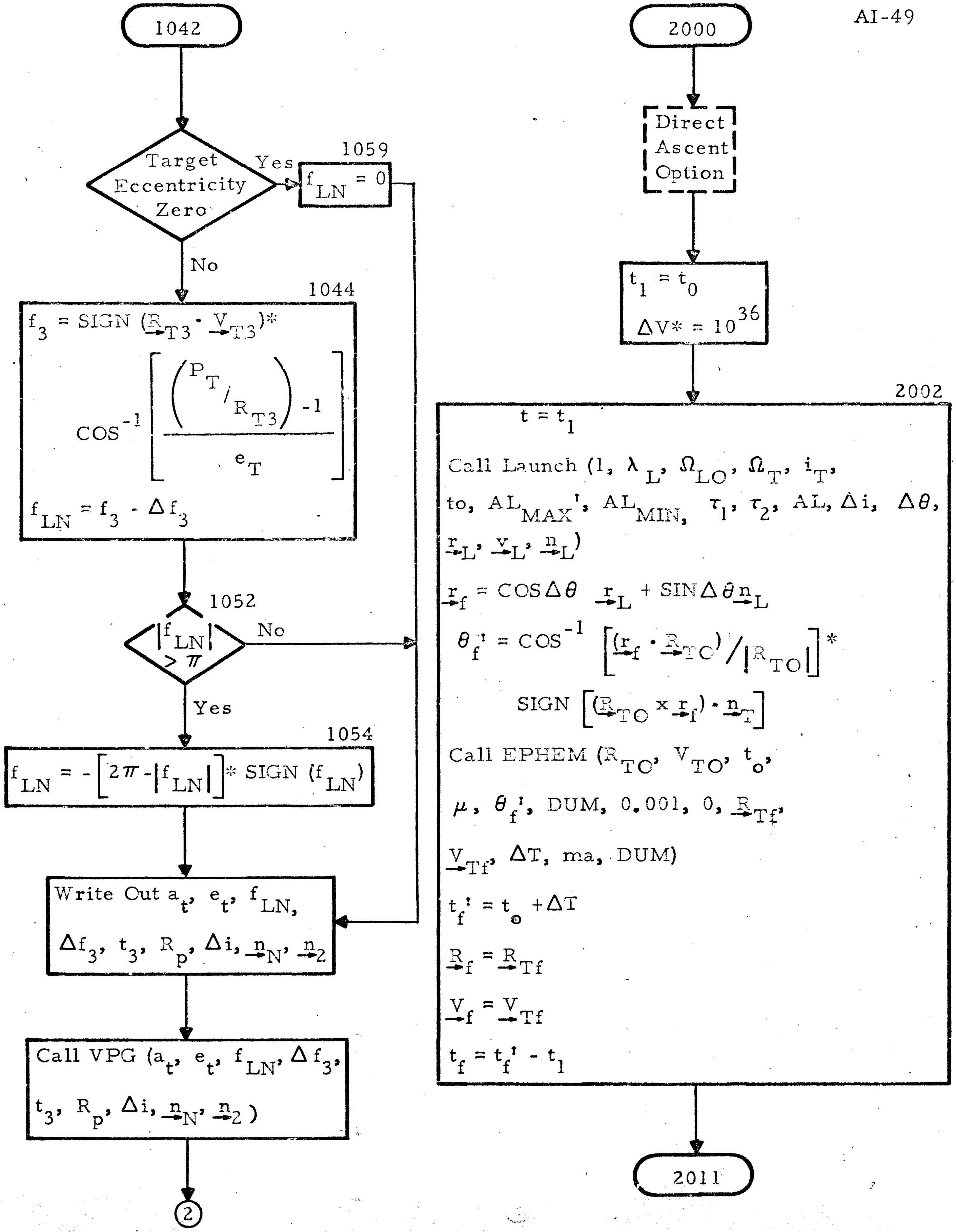

1042
Target Eccentricity Zero
Yes
No
1059
$f_{LN} = 0$
1044
$f_3 = \text{SIGN}\,(\underset{\rightarrow}{R}_{T3} \cdot \underset{\rightarrow}{V}_{T3})*$
$\cos^{-1}\left[\frac{\left(P_T/R_{T3}\right)-1}{e_T}\right]$
$f_{LN} = f_3 - \Delta f_3$
1052
$|f_{LN}| > \pi$
No
Yes
1054
$f_{LN} = -\left[2\pi - |f_{LN}|\right] * \text{SIGN}\,(f_{LN})$
Write Out a_t, e_t, f_{LN}, Δf_3, t_3, R_p, Δi, $\underset{\rightarrow}{n}_N$, $\underset{\rightarrow}{n}_2$
Call VPG (a_t, e_t, f_{LN}, Δf_3, t_3, R_p, Δi, $\underset{\rightarrow}{n}_N$, $\underset{\rightarrow}{n}_2$)
2
2000
Direct Ascent Option
$t_1 = t_0$
$\Delta V* = 10^{36}$
2002
$t = t_1$
Call Launch (1, λ_L, Ω_{LO}, Ω_T, i_T, to, AL_{MAX}', AL_{MIN}, τ_1, τ_2, AL, Δi, $\Delta\theta$, $\underset{\rightarrow}{r}_L$, $\underset{\rightarrow}{v}_L$, $\underset{\rightarrow}{n}_L$)
$\underset{\rightarrow}{r}_f = \cos\Delta\theta \ \underset{\rightarrow}{r}_L + \sin\Delta\theta\,\underset{\rightarrow}{n}_L$
$\theta_f' = \cos^{-1}\left[(\underset{\rightarrow}{r}_f \cdot \underset{\rightarrow}{R}_{TO})/|R_{TO}|\right]*$
$\text{SIGN}\left[(\underset{\rightarrow}{R}_{TO} \times \underset{\rightarrow}{r}_f) \cdot \underset{\rightarrow}{n}_T\right]$
Call EPHEM (R_{TO}, V_{TO}, t_o, μ, θ_f', DUM, 0.001, 0, $\underset{\rightarrow}{R}_{Tf}$, $\underset{\rightarrow}{V}_{Tf}$, ΔT, ma, DUM)
$t_f' = t_o + \Delta T$
$\underset{\rightarrow}{R}_f = \underset{\rightarrow}{R}_{Tf}$
$\underset{\rightarrow}{V}_f = \underset{\rightarrow}{V}_{Tf}$
$t_f = t_f' - t_1$
2011

Figure I-2 Main Program
(Sheet 7 of 12)

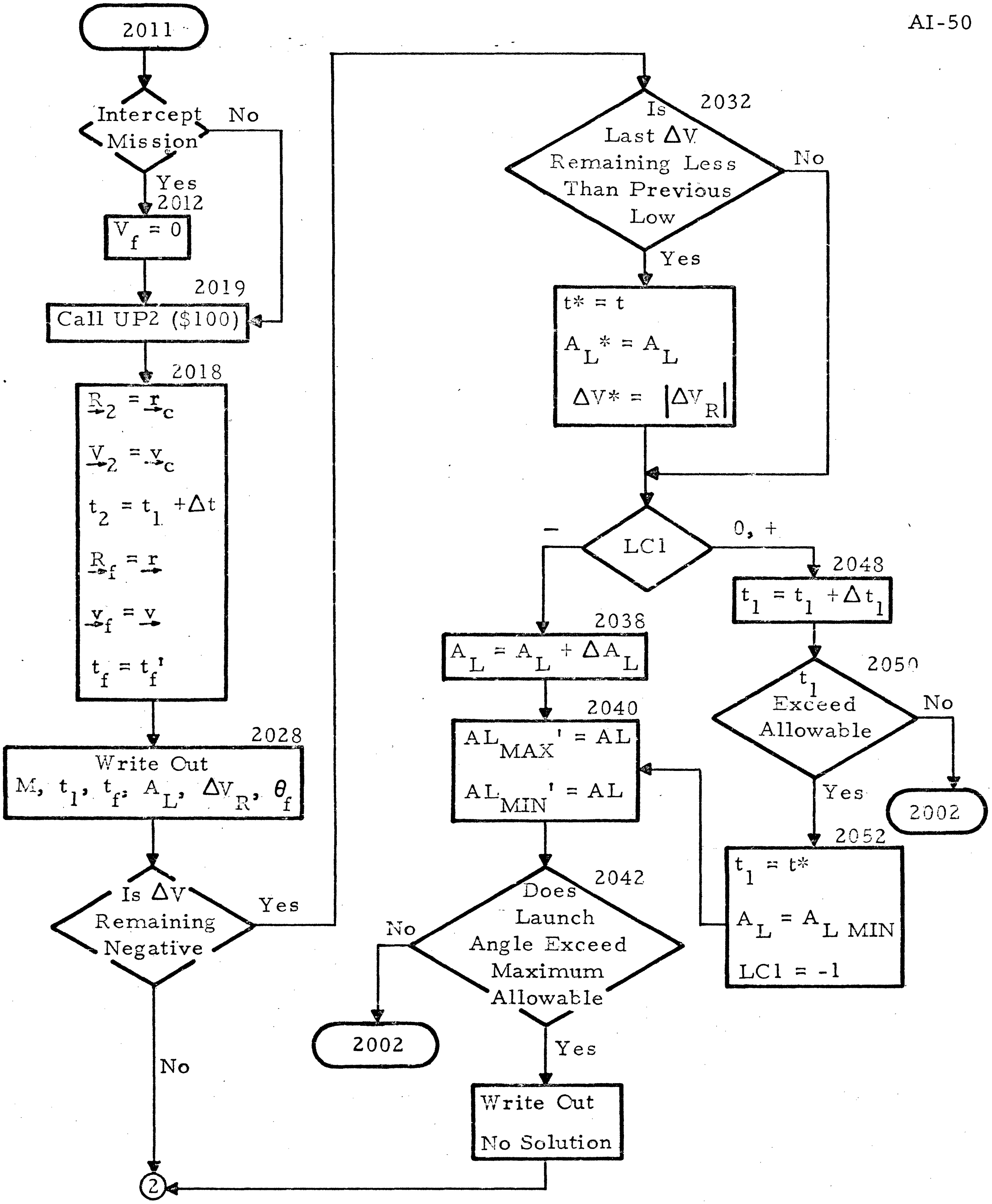

2011
Intercept Mission
No
Yes
2012
$V_f = 0$
2019
Call UP2 ($100)
2018
$\underset{\rightarrow}{R}_2 = \underset{\rightarrow}{r}_c$
$\underset{\rightarrow}{V}_2 = \underset{\rightarrow}{v}_c$
$t_2 = t_1 + \Delta t$
$\underset{\rightarrow}{R}_f = \underset{\rightarrow}{r}$
$\underset{\rightarrow}{v}_f = \underset{\rightarrow}{v}$
$t_f = t_f'$
2028
Write Out
$M, t_1, t_f, A_L, \Delta V_R, \theta_f$
Is ΔV Remaining Negative
Yes
No
2
2032
Is Last ΔV Remaining Less Than Previous Low
No
Yes
$t^* = t$
$A_L^* = A_L$
$\Delta V^* = |\Delta V_R|$
LC1
−
0, +
2048
$t_1 = t_1 + \Delta t_1$
2050
t_1 Exceed Allowable
No
2002
Yes
2052
$t_1 = t^*$
$A_L = A_{L\ MIN}$
LC1 = -1
2038
$A_L = A_L + \Delta A_L$
2040
$AL_{MAX}' = AL$
$AL_{MIN}' = AL$
2042
Does Launch Angle Exceed Maximum Allowable
No
2002
Yes
Write Out
No Solution

Figure I-2 Main Program
(Sheet 8 of 12)

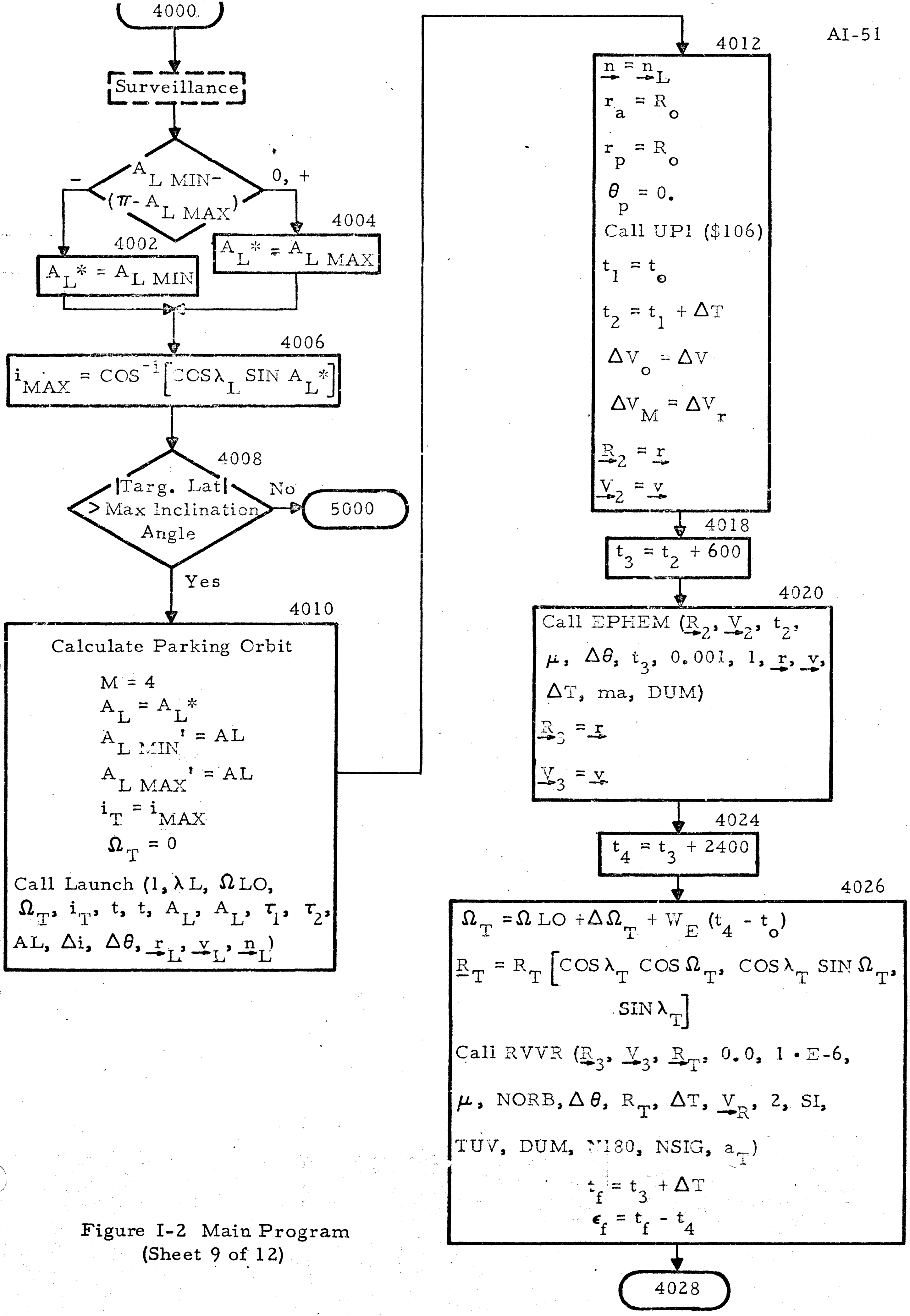

4000
Surveillance
$A_{L\ MIN} - (\pi - A_{L\ MAX})$
-
0, +
4002
$A_L^* = A_{L\ MIN}$
4004
$A_L^* = A_{L\ MAX}$
4006
$i_{MAX} = \cos^{-1}\left[\cos\lambda_L \sin A_L^*\right]$
4008
|Targ. Lat| > Max Inclination Angle
No
5000
Yes
4010
Calculate Parking Orbit
$M = 4$
$A_L = A_L^*$
$A_{L\ MIN}' = AL$
$A_{L\ MAX}' = AL$
$i_T = i_{MAX}$
$\Omega_T = 0$
Call Launch (1, λL, ΩLO, Ω_T, i_T, t, t, A_L, A_L, τ_1, τ_2, AL, Δi, $\Delta\theta$, $\underrightarrow{r}_L$, $\underrightarrow{v}_L$, $\underrightarrow{n}_L$)
4012
$\underrightarrow{n} = \underrightarrow{n}_L$
$r_a = R_o$
$r_p = R_o$
$\theta_p = 0.$
Call UP1 ($106)
$t_1 = t_o$
$t_2 = t_1 + \Delta T$
$\Delta V_o = \Delta V$
$\Delta V_M = \Delta V_r$
$\underrightarrow{R}_2 = \underrightarrow{r}$
$\underrightarrow{V}_2 = \underrightarrow{v}$
4018
$t_3 = t_2 + 600$
4020
Call EPHEM ($\underrightarrow{R}_2$, $\underrightarrow{V}_2$, t_2, μ, $\Delta\theta$, t_3, 0.001, 1, $\underrightarrow{r}$, $\underrightarrow{v}$, ΔT, ma, DUM)
$\underrightarrow{R}_3 = \underrightarrow{r}$
$\underrightarrow{V}_3 = \underrightarrow{v}$
4024
$t_4 = t_3 + 2400$
4026
$\Omega_T = \Omega LO + \Delta\Omega_T + W_E (t_4 - t_o)$
$\underrightarrow{R}_T = R_T \left[\cos\lambda_T \cos\Omega_T, \cos\lambda_T \sin\Omega_T, \sin\lambda_T\right]$
Call RVVR ($\underrightarrow{R}_3$, $\underrightarrow{V}_3$, $\underrightarrow{R}_T$, 0.0, 1 • E-6, μ, NORB, $\Delta\theta$, R_T, ΔT, $\underrightarrow{V}_R$, 2, SI, TUV, DUM, N180, NSIG, a_T)
$t_f = t_3 + \Delta T$
$\epsilon_f = t_f - t_4$
4028

Figure I-2 Main Program
(Sheet 9 of 12)

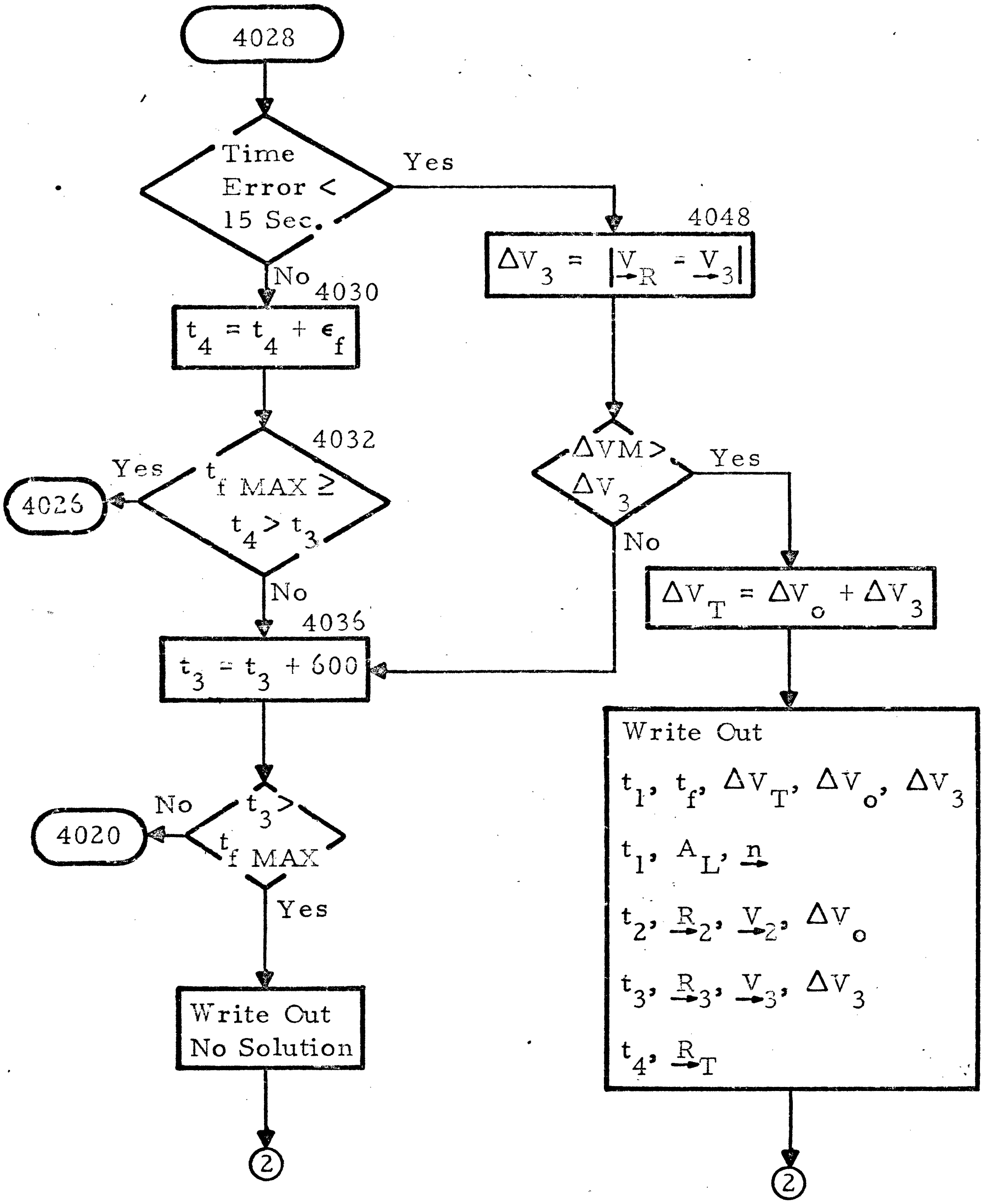

4028
Time Error < 15 Sec.
Yes
No
4048
$\Delta V_3 = |\underset{\rightarrow}{V}_R - \underset{\rightarrow}{V}_3|$
4030
$t_4 = t_4 + \epsilon_f$
4032
Yes
t_f MAX ≥ $t_4 > t_3$
4026
No
$\Delta VM > \Delta V_3$
Yes
No
$\Delta V_T = \Delta V_o + \Delta V_3$
4036
$t_3 = t_3 + 600$
No
4020
$t_3 > t_f$ MAX
Yes
Write Out No Solution
2
Write Out
$t_1, t_f, \Delta V_T, \Delta V_o, \Delta V_3$
$t_1, A_L, \underset{\rightarrow}{n}$
$t_2, \underset{\rightarrow}{R}_2, \underset{\rightarrow}{V}_2, \Delta V_o$
$t_3, \underset{\rightarrow}{R}_3, \underset{\rightarrow}{V}_3, \Delta V_3$
$t_4, \underset{\rightarrow}{R}_T$
2

Figure I-2 Main Program
(Sheet 10 of 12)

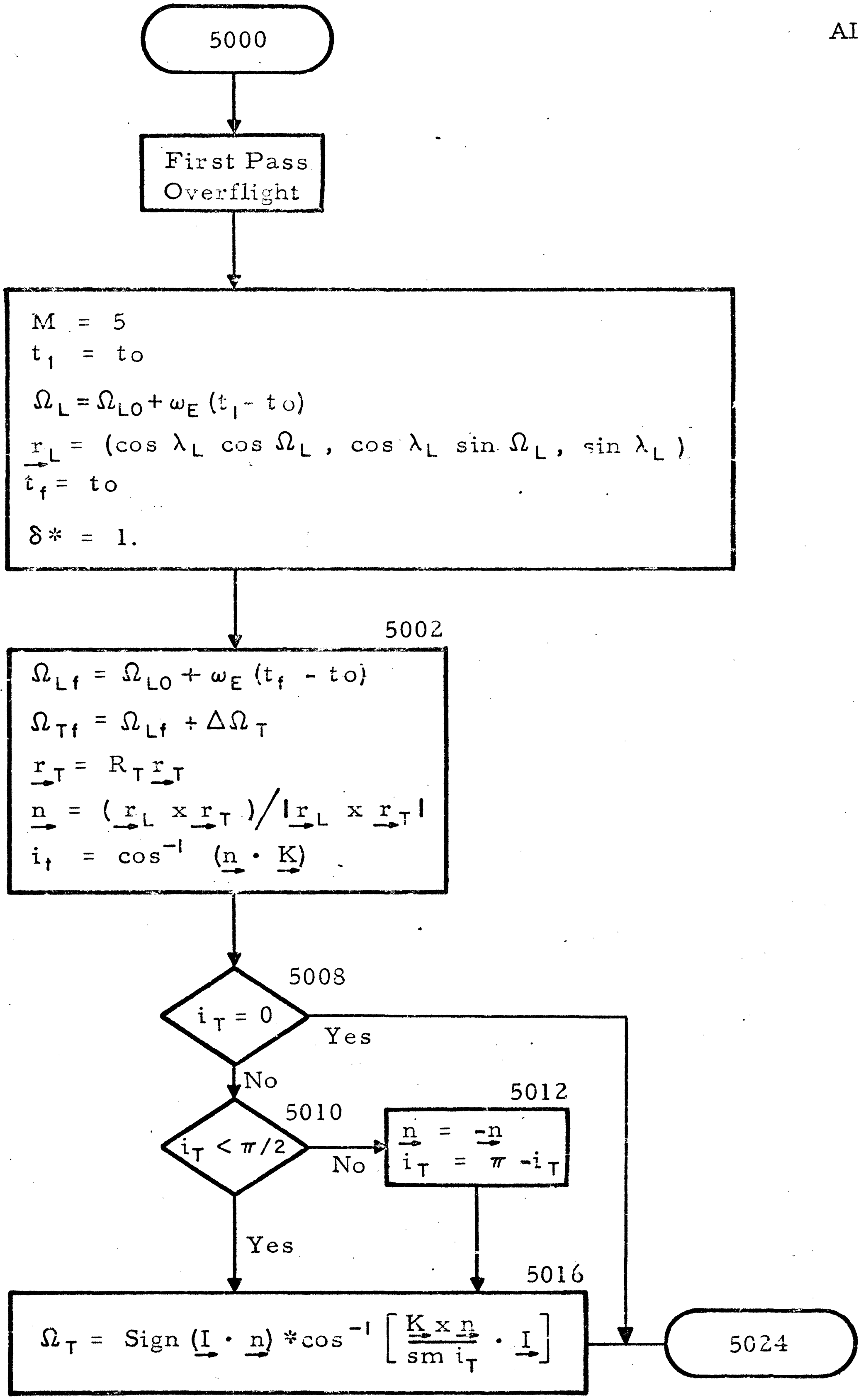

5000
First Pass Overflight
$M = 5$
$t_1 = to$
$\Omega_L = \Omega_{LO} + \omega_E (t_1 - to)$
$\underrightarrow{r}_L = (\cos \lambda_L \cos \Omega_L , \cos \lambda_L \sin \Omega_L , \sin \lambda_L)$
$t_f = to$
$\delta * = 1.$
5002
$\Omega_{Lf} = \Omega_{LO} + \omega_E (t_f - to)$
$\Omega_{Tf} = \Omega_{Lf} + \Delta\Omega_T$
$\underrightarrow{r}_T = R_T \underrightarrow{r}_T$
$\underrightarrow{n} = (\underrightarrow{r}_L \times \underrightarrow{r}_T) / |\underrightarrow{r}_L \times \underrightarrow{r}_T|$
$i_t = \cos^{-1} (\underrightarrow{n} \cdot \underrightarrow{K})$
5008
$i_T = 0$
Yes
No
5010
$i_T < \pi/2$
No
5012
$\underrightarrow{n} = -\underrightarrow{n}$
$i_T = \pi - i_T$
Yes
5016
$\Omega_T = \text{Sign} (\underrightarrow{I} \cdot \underrightarrow{n}) * \cos^{-1} \left[\frac{\underrightarrow{K} \times \underrightarrow{n}}{\text{sm } i_T} \cdot \underrightarrow{I} \right]$
5024

Figure I-2 Main Program
(Sheet 11 of 12)

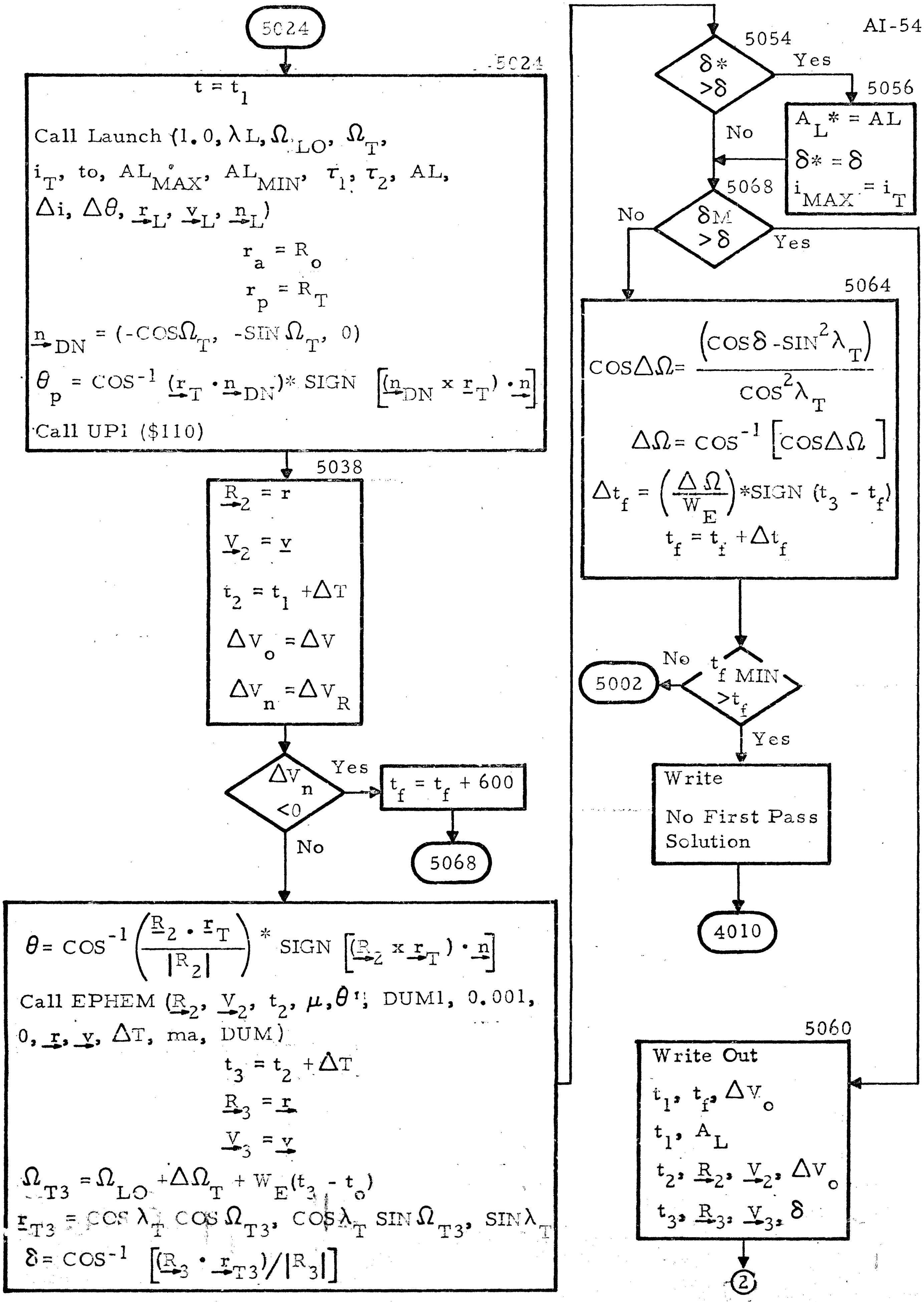

5024
5024
$t = t_1$
Call Launch $(1.0, \lambda L, \Omega_{LO}, \Omega_T, i_T, to, AL_{MAX}, AL_{MIN}, \tau_1, \tau_2, AL, \Delta i, \Delta\theta, \underset{\rightarrow}{r}_L, \underset{\rightarrow}{v}_L, \underset{\rightarrow}{n}_L)$
$r_a = R_o$
$r_p = R_T$
$\underset{\rightarrow}{n}_{DN} = (-COS\Omega_T, -SIN\,\Omega_T, 0)$
$\theta_p = COS^{-1}(\underset{\rightarrow}{r}_T \cdot \underset{\rightarrow}{n}_{DN})* SIGN\ [(\underset{\rightarrow}{n}_{DN} \times \underset{\rightarrow}{r}_T) \cdot \underset{\rightarrow}{n}]$
Call UP1 ($110)
5038
$\underset{\rightarrow}{R}_2 = r$
$\underset{\rightarrow}{V}_2 = \underset{\rightarrow}{v}$
$t_2 = t_1 + \Delta T$
$\Delta V_o = \Delta V$
$\Delta V_n = \Delta V_R$
$\Delta V_n < 0$
Yes
$t_f = t_f + 600$
5068
No
$\theta = COS^{-1}\left(\frac{R_2 \cdot r_T}{|R_2|}\right) * SIGN\ [(\underset{\rightarrow}{R}_2 \times \underset{\rightarrow}{r}_T) \cdot \underset{\rightarrow}{n}]$
Call EPHEM $(\underset{\rightarrow}{R}_2, \underset{\rightarrow}{V}_2, t_2, \mu, \theta'$, DUM1, 0.001, 0, $\underset{\rightarrow}{r}, \underset{\rightarrow}{v}, \Delta T$, ma, DUM)
$t_3 = t_2 + \Delta T$
$\underset{\rightarrow}{R}_3 = \underset{\rightarrow}{r}$
$\underset{\rightarrow}{V}_3 = \underset{\rightarrow}{v}$
$\Omega_{T3} = \Omega_{LO} + \Delta\Omega_T + W_E(t_3 - t_o)$
$\underset{\rightarrow}{r}_{T3} = COS\,\lambda_T\ COS\,\Omega_{T3}, COS\lambda_T\ SIN\,\Omega_{T3}, SIN\lambda_T$
$\delta = COS^{-1}\ [(\underset{\rightarrow}{R}_3 \cdot \underset{\rightarrow}{r}_{T3})/|R_3|]$
5054
$\delta* > \delta$
Yes
5056
$A_L* = AL$
$\delta* = \delta$
$i_{MAX} = i_T$
No
5068
$\delta_M > \delta$
No
Yes
5064
$COS\Delta\Omega = \frac{(COS\delta - SIN^2\lambda_T)}{COS^2\lambda_T}$
$\Delta\Omega = COS^{-1}\ [COS\Delta\Omega]$
$\Delta t_f = \left(\frac{\Delta\Omega}{W_E}\right) * SIGN\ (t_3 - t_f)$
$t_f = t_f + \Delta t_f$
$t_{f\ MIN} > t_f$
No
5002
Yes
Write
No First Pass Solution
4010
5060
Write Out
$t_1, t_f, \Delta V_o$
t_1, A_L
$t_2, \underset{\rightarrow}{R}_2, \underset{\rightarrow}{V}_2, \Delta V_o$
$t_3, \underset{\rightarrow}{R}_3, \underset{\rightarrow}{V}_3, \delta$
2

Figure I-2 Main Program
(Sheet 12 of 12)

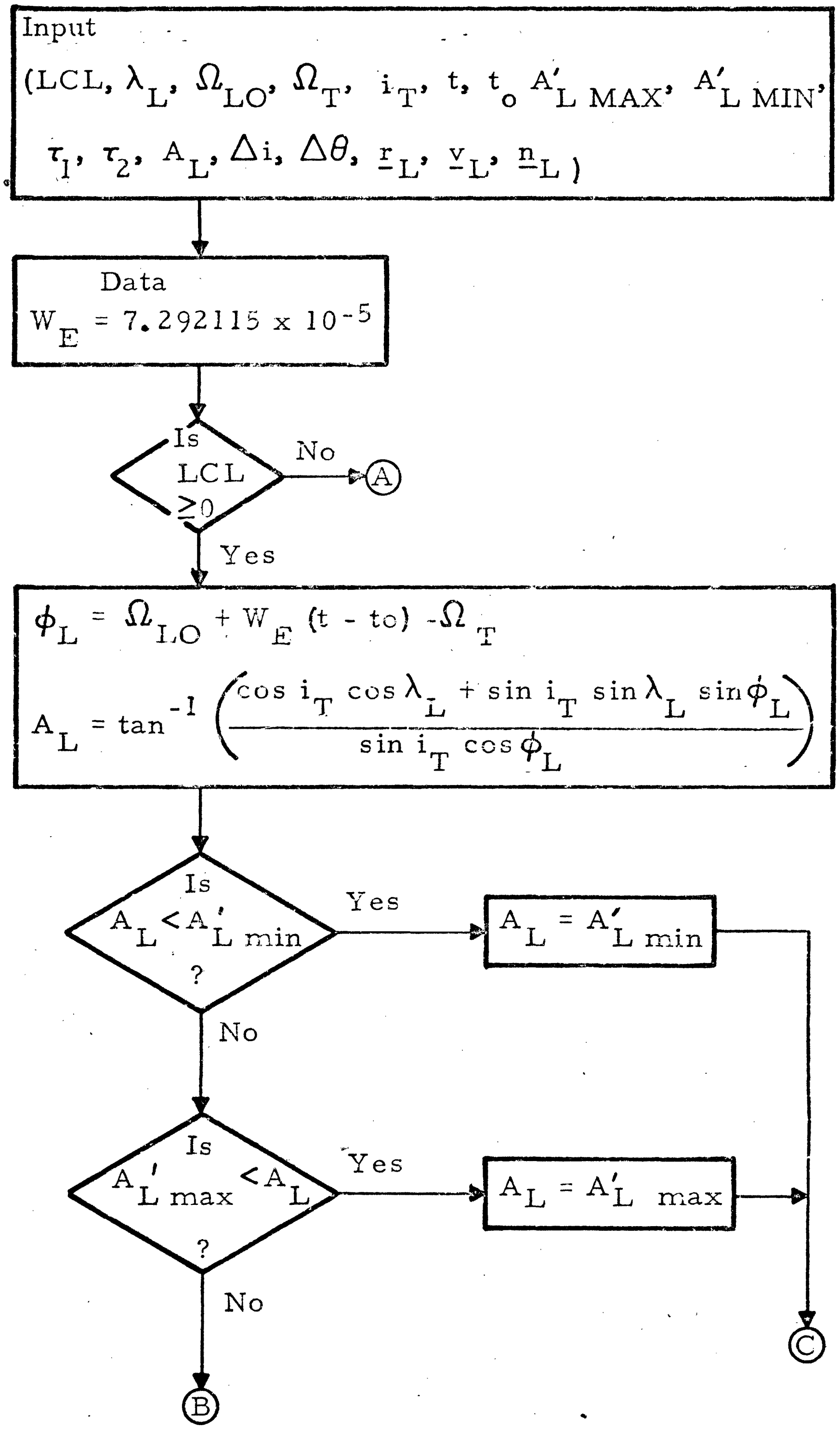
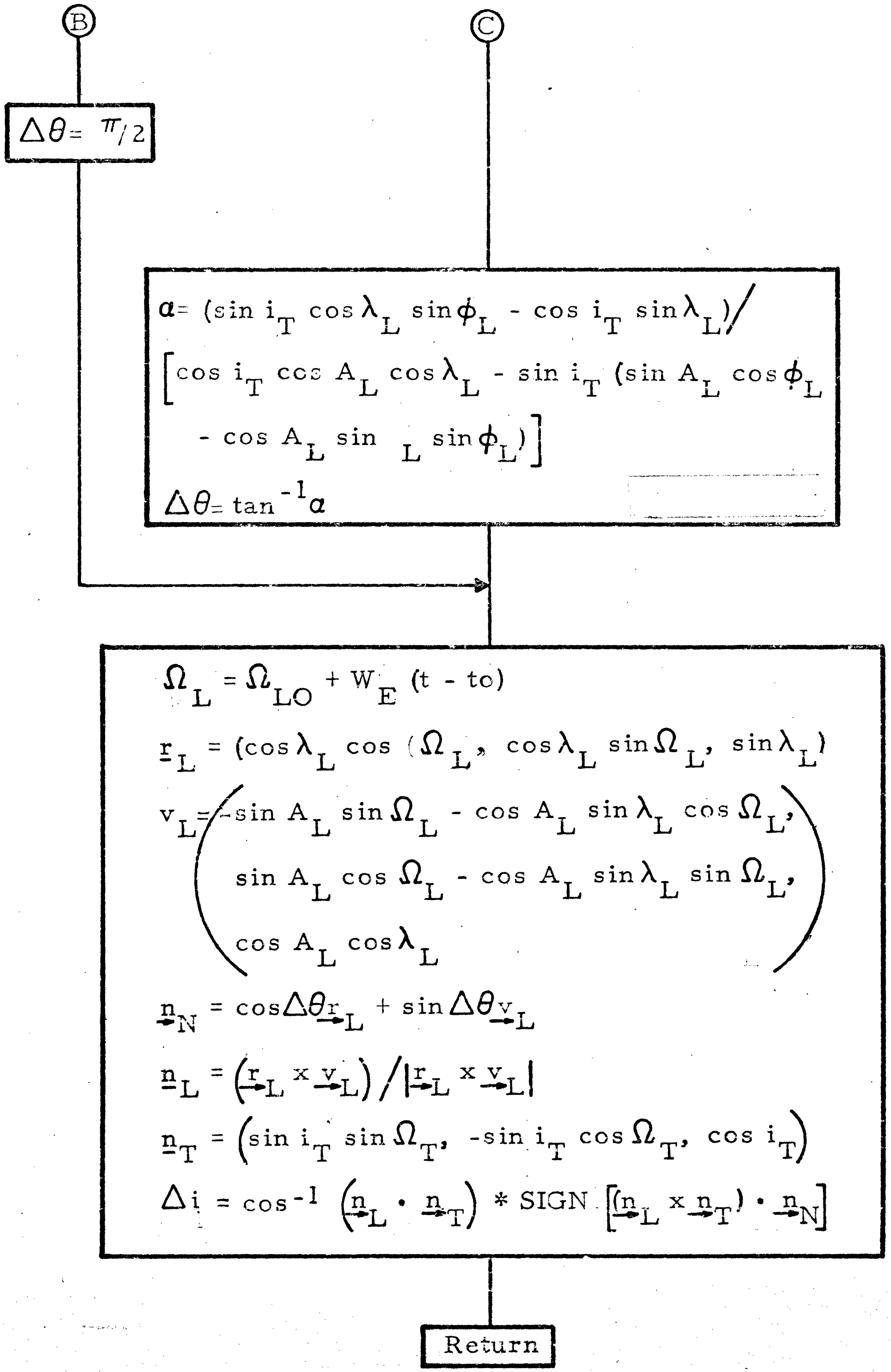

Input
$(LCL, \lambda_L, \Omega_{LO}, \Omega_T, i_T, t, t_o\, A'_{L\,MAX}, A'_{L\,MIN}, \tau_1, \tau_2, A_L, \Delta i, \Delta\theta, \underline{r}_L, \underline{v}_L, \underline{n}_L)$
Data
$W_E = 7.292115 \times 10^{-5}$
Is LCL ≥ 0
No
A
Yes
$\phi_L = \Omega_{LO} + W_E (t - to) - \Omega_T$
$A_L = \tan^{-1} \left(\frac{\cos i_T \cos \lambda_L + \sin i_T \sin \lambda_L \sin \phi_L}{\sin i_T \cos \phi_L} \right)$
Is $A_L < A'_{L\,min}$?
Yes
$A_L = A'_{L\,min}$
No
Is $A'_{L\,max} < A_L$?
Yes
$A_L = A'_L\ max$
No
B
C

Figure I-3 Launch Subroutine
(Sheet 1 of 3)



$\Delta\theta = \pi/2$

$$\alpha = (\sin i_T \cos\lambda_L \sin\phi_L - \cos i_T \sin\lambda_L) / \left[\cos i_T \cos A_L \cos\lambda_L - \sin i_T (\sin A_L \cos\phi_L - \cos A_L \sin {}_L \sin\phi_L)\right]$$

$$\Delta\theta = \tan^{-1}\alpha$$

$$\Omega_L = \Omega_{LO} + W_E (t - to)$$

$$\underline{r}_L = (\cos\lambda_L \cos (\Omega_L, \cos\lambda_L \sin\Omega_L, \sin\lambda_L)$$

$$v_L = \begin{pmatrix} -\sin A_L \sin\Omega_L - \cos A_L \sin\lambda_L \cos\Omega_L, \\ \sin A_L \cos\Omega_L - \cos A_L \sin\lambda_L \sin\Omega_L, \\ \cos A_L \cos\lambda_L \end{pmatrix}$$

$$\vec{n}_N = \cos\Delta\theta \vec{r}_L + \sin\Delta\theta \vec{v}_L$$

$$\underline{n}_L = (\vec{r}_L \times \vec{v}_L) / |\vec{r}_L \times \vec{v}_L|$$

$$\underline{n}_T = (\sin i_T \sin\Omega_T, -\sin i_T \cos\Omega_T, \cos i_T)$$

$$\Delta i = \cos^{-1} (\vec{n}_L \cdot \vec{n}_T) * \mathrm{SIGN} \left[(\vec{n}_L \times \vec{n}_T) \cdot \vec{n}_N\right]$$

Figure I-3 Launch Subroutine
(Sheet 2 of 3)

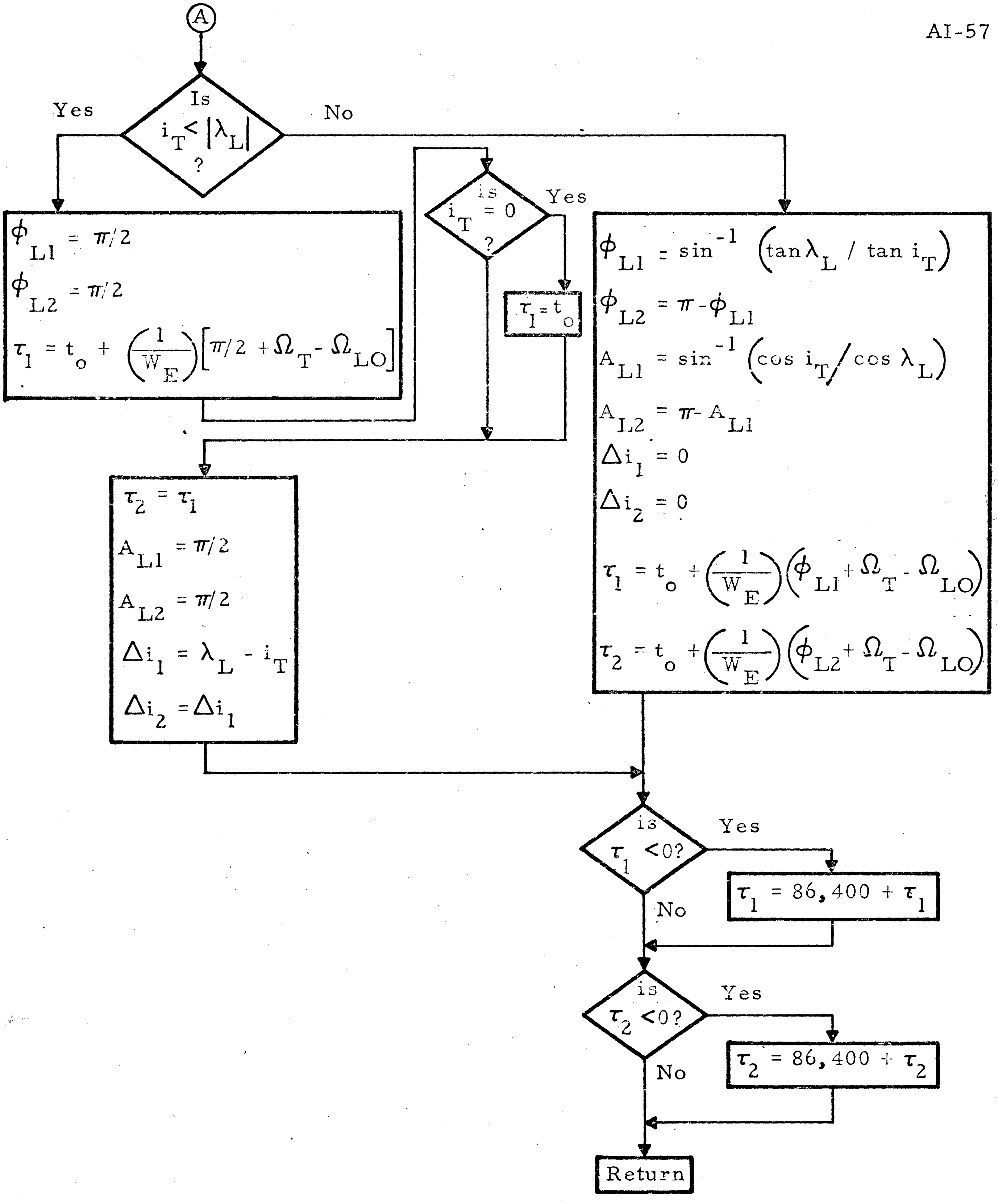

A
Is $i_T < |\lambda_L|$?
Yes
No
$\phi_{L1} = \pi/2$
$\phi_{L2} = \pi/2$
$\tau_1 = t_o + \left(\frac{1}{W_E}\right)\left[\pi/2 + \Omega_T - \Omega_{LO}\right]$
is $i_T = 0$?
Yes
$\tau_1 = t_o$
$\phi_{L1} = \sin^{-1}\left(\tan\lambda_L / \tan i_T\right)$
$\phi_{L2} = \pi - \phi_{L1}$
$A_{L1} = \sin^{-1}\left(\cos i_T / \cos\lambda_L\right)$
$A_{L2} = \pi - A_{L1}$
$\Delta i_1 = 0$
$\Delta i_2 = 0$
$\tau_1 = t_o + \left(\frac{1}{W_E}\right)\left(\phi_{L1} + \Omega_T - \Omega_{LO}\right)$
$\tau_2 = t_o + \left(\frac{1}{W_E}\right)\left(\phi_{L2} + \Omega_T - \Omega_{LO}\right)$
$\tau_2 = \tau_1$
$A_{L1} = \pi/2$
$A_{L2} = \pi/2$
$\Delta i_1 = \lambda_L - i_T$
$\Delta i_2 = \Delta i_1$
is $\tau_1 < 0$?
Yes
No
$\tau_1 = 86,400 + \tau_1$
is $\tau_2 < 0$?
Yes
No
$\tau_2 = 86,400 + \tau_2$
Return

Figure I-3 Launch Subroutine
(Sheet 3 of 3)

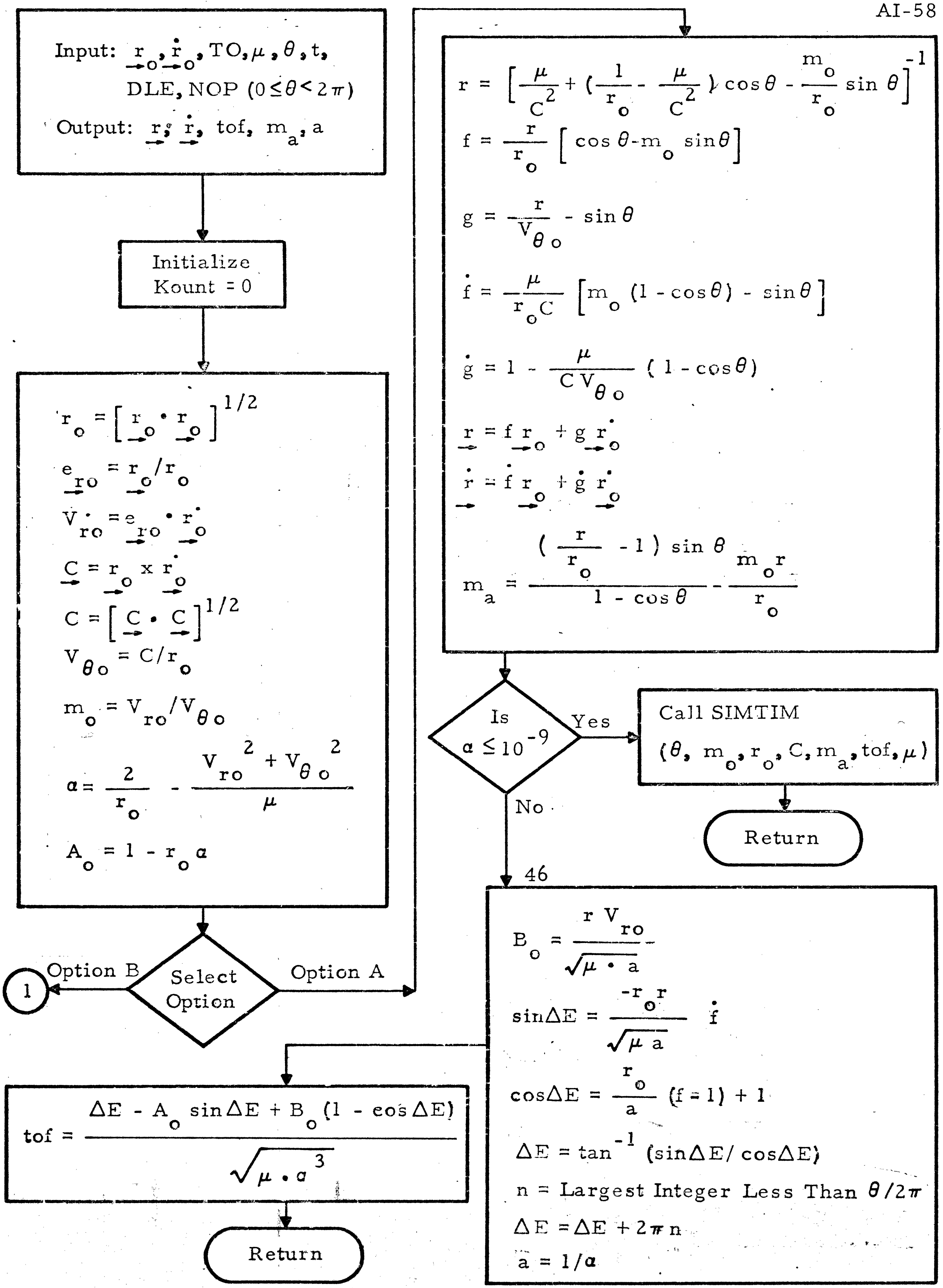

Input: $\vec{r}_o, \dot{\vec{r}}_o$, TO, μ, θ, t, DLE, NOP $(0 \leq \theta < 2\pi)$
Output: $\vec{r}, \dot{\vec{r}}$, tof, m_a, a
Initialize Kount = 0
$r_o = \left[\vec{r}_o \cdot \vec{r}_o\right]^{1/2}$
$\vec{e}_{ro} = \vec{r}_o / r_o$
$V_{ro} = \vec{e}_{ro} \cdot \dot{\vec{r}}_o$
$\vec{C} = \vec{r}_o \times \dot{\vec{r}}_o$
$C = \left[\vec{C} \cdot \vec{C}\right]^{1/2}$
$V_{\theta o} = C / r_o$
$m_o = V_{ro} / V_{\theta o}$
$\alpha = \frac{2}{r_o} - \frac{V_{ro}^2 + V_{\theta o}^2}{\mu}$
$A_o = 1 - r_o \alpha$
Select Option
Option B
1
Option A
$r = \left[\frac{\mu}{C^2} + \left(\frac{1}{r_o} - \frac{\mu}{C^2}\right) \cos\theta - \frac{m_o}{r_o} \sin\theta\right]^{-1}$
$f = \frac{r}{r_o}\left[\cos\theta - m_o \sin\theta\right]$
$g = \frac{r}{V_{\theta o}} - \sin\theta$
$\dot{f} = \frac{\mu}{r_o C}\left[m_o (1 - \cos\theta) - \sin\theta\right]$
$\dot{g} = 1 - \frac{\mu}{C V_{\theta o}} (1 - \cos\theta)$
$\vec{r} = f \vec{r}_o + g \dot{\vec{r}}_o$
$\dot{\vec{r}} = \dot{f} \vec{r}_o + \dot{g} \dot{\vec{r}}_o$
$m_a = \frac{\left(\frac{r}{r_o} - 1\right) \sin\theta}{1 - \cos\theta} - \frac{m_o r}{r_o}$
Is $\alpha \leq 10^{-9}$
Yes
Call SIMTIM $(\theta, m_o, r_o, C, m_a, \text{tof}, \mu)$
Return
No
46
$B_o = \frac{r V_{ro}}{\sqrt{\mu \cdot a}}$
$\sin\Delta E = \frac{-r_o r}{\sqrt{\mu a}} \dot{f}$
$\cos\Delta E = \frac{r_o}{a} (f = 1) + 1$
$\Delta E = \tan^{-1} (\sin\Delta E / \cos\Delta E)$
n = Largest Integer Less Than $\theta / 2\pi$
$\Delta E = \Delta E + 2\pi n$
$a = 1/\alpha$
$\text{tof} = \frac{\Delta E - A_o \sin\Delta E + B_o (1 - \text{eos}\,\Delta E)}{\sqrt{\mu \cdot \alpha^3}}$
Return

Figure I-4 EPHEM Subroutine
(Sheet 1 of 3)

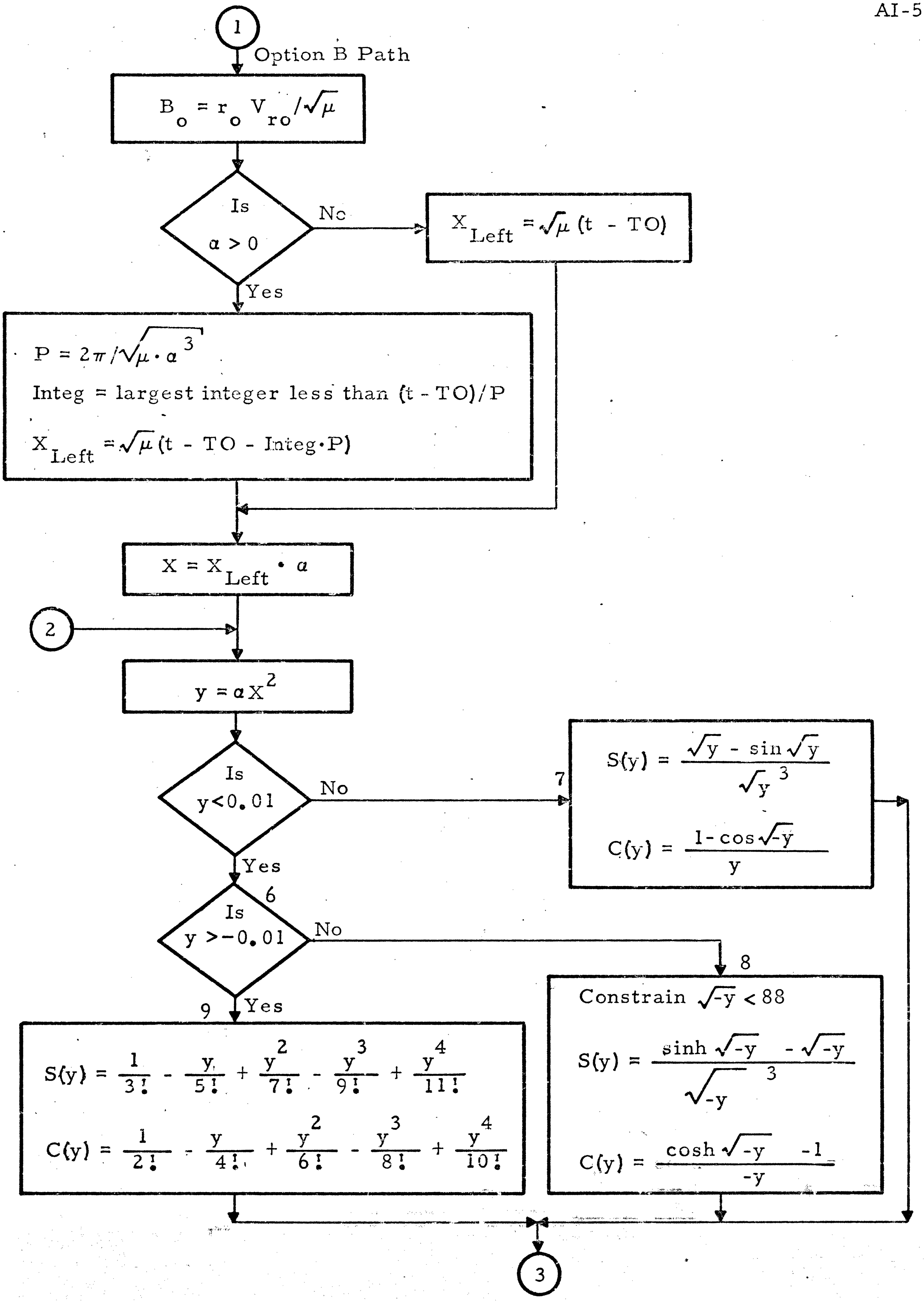

1
Option B Path
$B_o = r_o V_{ro}/\sqrt{\mu}$
Is $a > 0$
No
$X_{Left} = \sqrt{\mu}\,(t - TO)$
Yes
$P = 2\pi/\sqrt{\mu \cdot a^3}$
Integ = largest integer less than $(t - TO)/P$
$X_{Left} = \sqrt{\mu}\,(t - TO - Integ \cdot P)$
$X = X_{Left} \cdot a$
2
$y = aX^2$
Is $y < 0.01$
No
7
$S(y) = \dfrac{\sqrt{y} - \sin\sqrt{y}}{\sqrt{y}^3}$
$C(y) = \dfrac{1 - \cos\sqrt{-y}}{y}$
Yes
6
Is $y > -0.01$
No
8
Constrain $\sqrt{-y} < 88$
$S(y) = \dfrac{\sinh\sqrt{-y} - \sqrt{-y}}{\sqrt{-y}^3}$
$C(y) = \dfrac{\cosh\sqrt{-y} - 1}{-y}$
9
Yes
$S(y) = \dfrac{1}{3!} - \dfrac{y}{5!} + \dfrac{y^2}{7!} - \dfrac{y^3}{9!} + \dfrac{y^4}{11!}$
$C(y) = \dfrac{1}{2!} - \dfrac{y}{4!} + \dfrac{y^2}{6!} - \dfrac{y^3}{8!} + \dfrac{y^4}{10!}$
3

Figure I-4 EPHEM Subroutin
(Sheet 2 of 3)

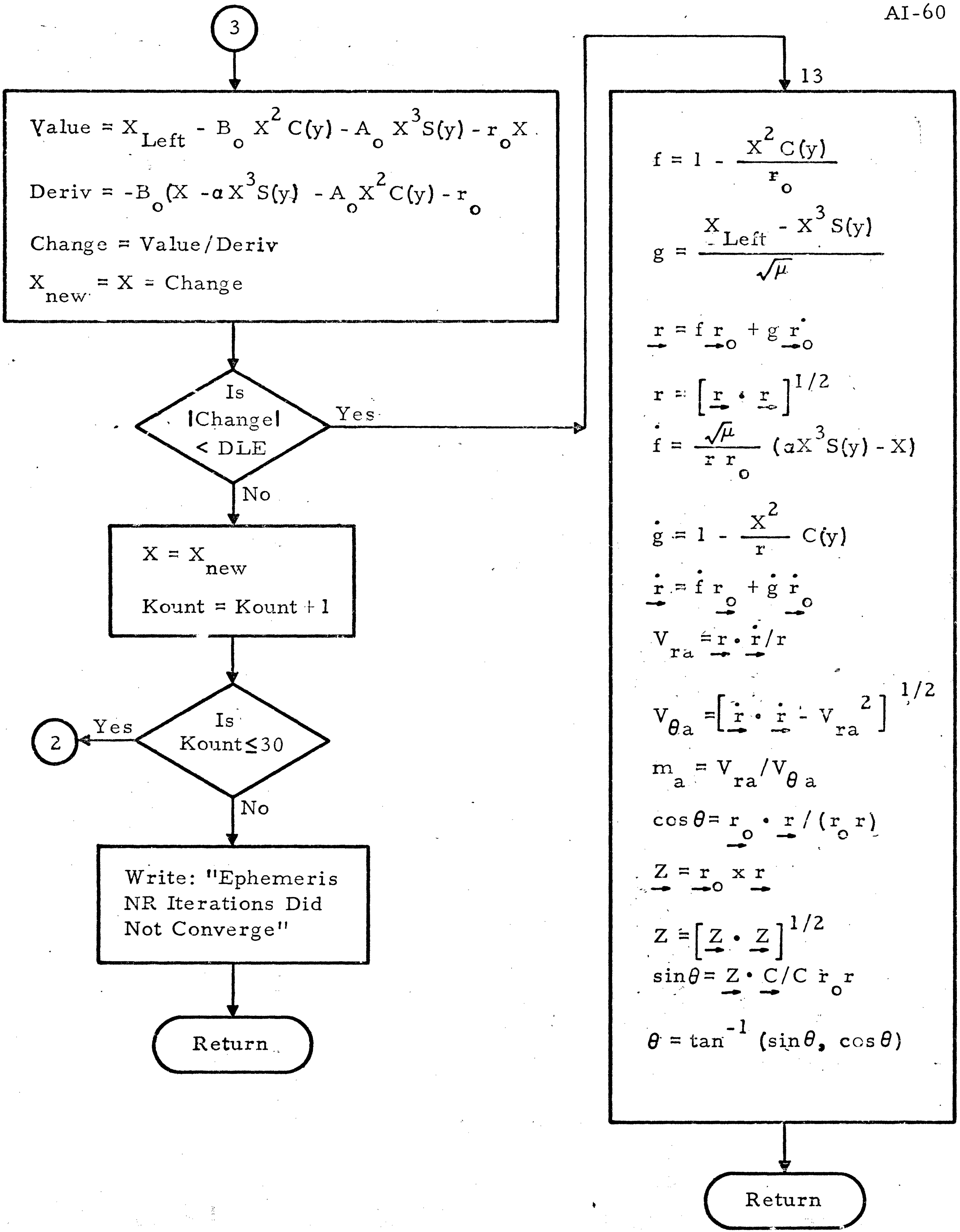

3
$\text{Value} = X_{Left} - B_o X^2 C(y) - A_o X^3 S(y) - r_o X$
$\text{Deriv} = -B_o(X - \alpha X^3 S(y) - A_o X^2 C(y) - r_o$
$\text{Change} = \text{Value}/\text{Deriv}$
$X_{new} = X = \text{Change}$
Is |Change| < DLE
Yes
No
$X = X_{new}$
Kount = Kount + 1
Is Kount ≤ 30
Yes
2
No
Write: "Ephemeris NR Iterations Did Not Converge"
Return
13
$f = 1 - \frac{X^2 C(y)}{r_o}$
$g = \frac{X_{Left} - X^3 S(y)}{\sqrt{\mu}}$
$\underset{\rightarrow}{r} = f \underset{\rightarrow}{r}_o + g \underset{\rightarrow}{\dot{r}}_o$
$r = [\underset{\rightarrow}{r} \cdot \underset{\rightarrow}{r}]^{1/2}$
$\dot{f} = \frac{\sqrt{\mu}}{r r_o} (\alpha X^3 S(y) - X)$
$\dot{g} = 1 - \frac{X^2}{r} C(y)$
$\underset{\rightarrow}{\dot{r}} = \dot{f} \underset{\rightarrow}{r}_o + \dot{g} \underset{\rightarrow}{\dot{r}}_o$
$V_{ra} = \underset{\rightarrow}{r} \cdot \underset{\rightarrow}{\dot{r}}/r$
$V_{\theta a} = [\underset{\rightarrow}{\dot{r}} \cdot \underset{\rightarrow}{\dot{r}} - V_{ra}{}^2]^{1/2}$
$m_a = V_{ra}/V_{\theta a}$
$\cos\theta = \underset{\rightarrow}{r}_o \cdot \underset{\rightarrow}{r}/(r_o r)$
$\underset{\rightarrow}{Z} = \underset{\rightarrow}{r}_o \times \underset{\rightarrow}{r}$
$Z = [\underset{\rightarrow}{Z} \cdot \underset{\rightarrow}{Z}]^{1/2}$
$\sin\theta = \underset{\rightarrow}{Z} \cdot \underset{\rightarrow}{C}/C\, r_o r$
$\theta = \tan^{-1}(\sin\theta, \cos\theta)$
Return

Figure I-4 EPHEM Subroutin
(Sheet 3 of 3)

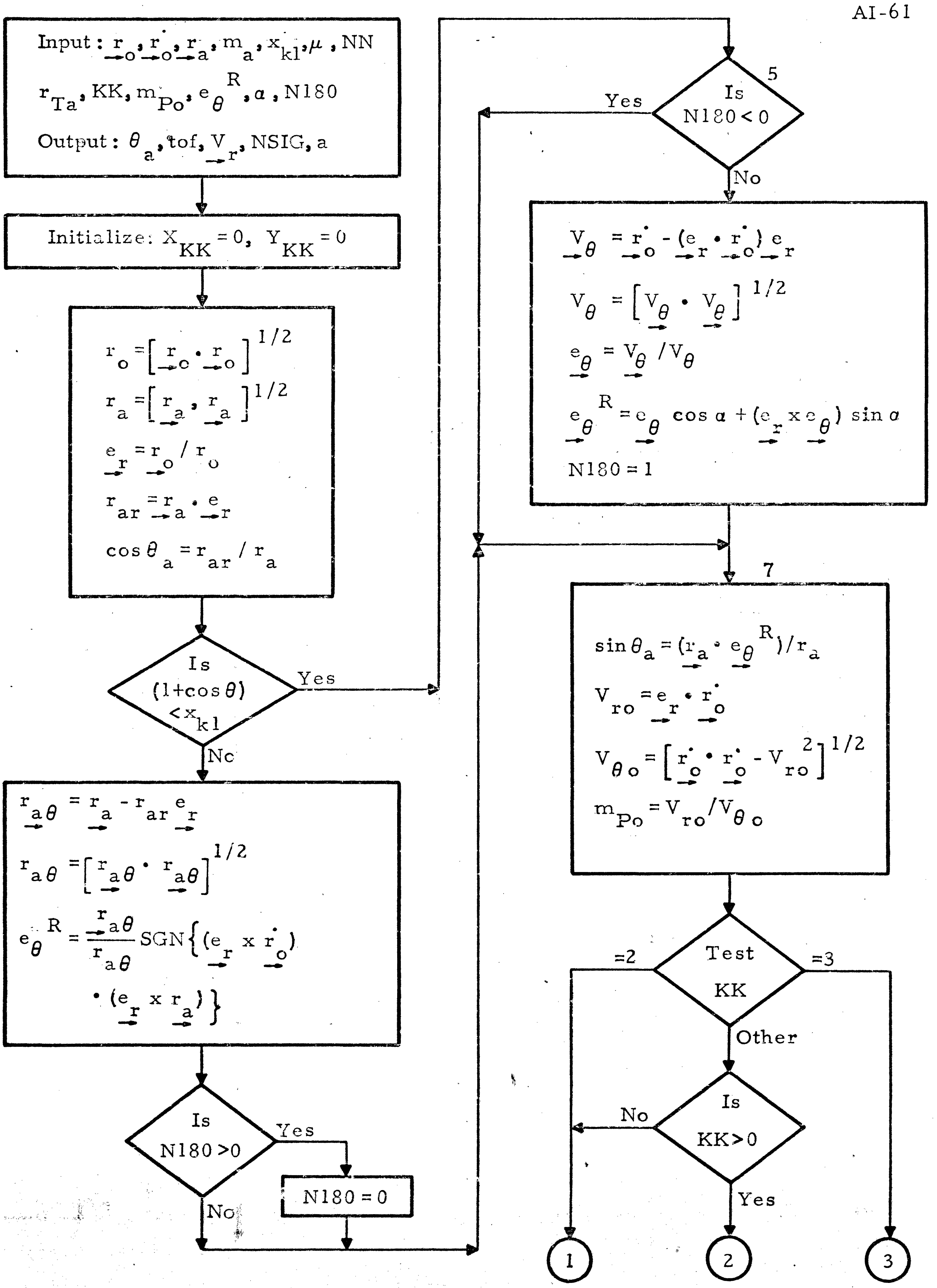

Input : $\underrightarrow{r}_o$, $\dot{\underrightarrow{r}}_o$, $\underrightarrow{r}_a$, m_a, x_{kl}, μ, NN
r_{Ta}, KK, m_{Po}, e_θ^R, a, N180
Output : θ_a, tof, $\underrightarrow{V}_r$, NSIG, a
Initialize: $X_{KK} = 0$, $Y_{KK} = 0$
$r_o = \left[\underrightarrow{r}_o \cdot \underrightarrow{r}_o\right]^{1/2}$
$r_a = \left[\underrightarrow{r}_a , \underrightarrow{r}_a\right]^{1/2}$
$\underrightarrow{e}_r = \underrightarrow{r}_o / r_o$
$r_{ar} = \underrightarrow{r}_a \cdot \underrightarrow{e}_r$
$\cos\theta_a = r_{ar} / r_a$
Is $(1+\cos\theta) < x_{kl}$
Yes
No
$\underrightarrow{r}_{a\theta} = \underrightarrow{r}_a - r_{ar}\underrightarrow{e}_r$
$r_{a\theta} = \left[\underrightarrow{r}_{a\theta} \cdot \underrightarrow{r}_{a\theta}\right]^{1/2}$
$e_\theta^R = \frac{\underrightarrow{r}_{a\theta}}{r_{a\theta}} \text{SGN}\left\{(\underrightarrow{e}_r \times \dot{\underrightarrow{r}}_o) \cdot (\underrightarrow{e}_r \times \underrightarrow{r}_a)\right\}$
Is N180 >0
Yes
N180 = 0
No
5
Is N180 < 0
Yes
No
$\underrightarrow{V}_\theta = \dot{\underrightarrow{r}}_o - (\underrightarrow{e}_r \cdot \dot{\underrightarrow{r}}_o)\underrightarrow{e}_r$
$V_\theta = \left[\underrightarrow{V}_\theta \cdot \underrightarrow{V}_\theta\right]^{1/2}$
$\underrightarrow{e}_\theta = \underrightarrow{V}_\theta / V_\theta$
$\underrightarrow{e}_\theta^R = \underrightarrow{e}_\theta \cos a + (\underrightarrow{e}_r \times \underrightarrow{e}_\theta) \sin a$
N180 = 1
7
$\sin\theta_a = (\underrightarrow{r}_a \cdot \underrightarrow{e}_\theta^R)/r_a$
$V_{ro} = \underrightarrow{e}_r \cdot \dot{\underrightarrow{r}}_o$
$V_{\theta o} = \left[\dot{\underrightarrow{r}}_o \cdot \dot{\underrightarrow{r}}_o - V_{ro}^2\right]^{1/2}$
$m_{Po} = V_{ro}/V_{\theta o}$
Test KK
=2
=3
Other
Is KK>0
No
Yes
1
2
3

Figure I-5 RVVR Subroutine (Sheet 1 of 3)

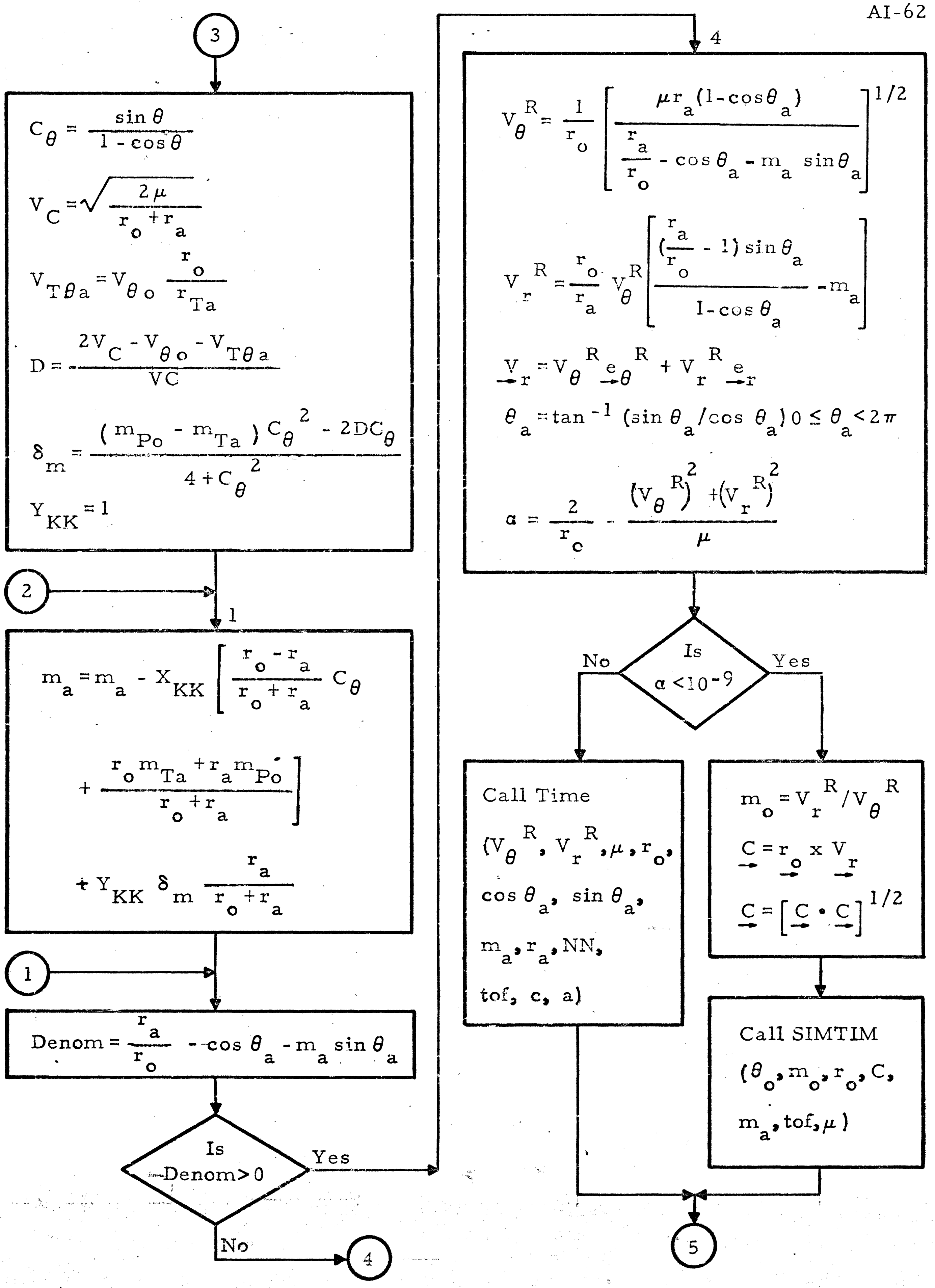

3
$C_\theta = \frac{\sin\theta}{1 - \cos\theta}$
$V_C = \sqrt{\frac{2\mu}{r_o + r_a}}$
$V_{T\theta a} = V_{\theta o} \frac{r_o}{r_{Ta}}$
$D = \frac{2V_C - V_{\theta o} - V_{T\theta a}}{VC}$
$\delta_m = \frac{(m_{Po} - m_{Ta}) C_\theta^2 - 2DC_\theta}{4 + C_\theta^2}$
$Y_{KK} = 1$
2
1
$m_a = m_a - X_{KK}\left[\frac{r_o - r_a}{r_o + r_a} C_\theta + \frac{r_o m_{Ta} + r_a m_{Po}}{r_o + r_a}\right] + Y_{KK}\,\delta_m \frac{r_a}{r_o + r_a}$
1
$Denom = \frac{r_a}{r_o} - -\cos\theta_a - m_a \sin\theta_a$
Is Denom>0
Yes
No
4
4
$V_\theta^R = \frac{1}{r_o}\left[\frac{\mu r_a (1 - \cos\theta_a)}{\frac{r_a}{r_o} - \cos\theta_a - m_a \sin\theta_a}\right]^{1/2}$
$V_r^R = \frac{r_o}{r_a} V_\theta^R \left[\frac{(\frac{r_a}{r_o} - 1)\sin\theta_a}{1 - \cos\theta_a} - m_a\right]$
$\underset{\rightarrow}{V}_r = V_\theta^R\, \underset{\rightarrow}{e}_\theta^R + V_r^R\, \underset{\rightarrow}{e}_r$
$\theta_a = \tan^{-1}(\sin\theta_a / \cos\theta_a)\ 0 \le \theta_a < 2\pi$
$a = \frac{2}{r_o} - \frac{(V_\theta^R)^2 + (V_r^R)^2}{\mu}$
Is $a < 10^{-9}$
No
Yes
Call Time (V_θ^R, V_r^R, μ, r_o, $\cos\theta_a$, $\sin\theta_a$, m_a, r_a, NN, tof, c, a)
$m_o = V_r^R / V_\theta^R$
$\underset{\rightarrow}{C} = \underset{\rightarrow}{r}_o \times \underset{\rightarrow}{V}_r$
$\underset{\rightarrow}{C} = [\underset{\rightarrow}{C} \cdot \underset{\rightarrow}{C}]^{1/2}$
Call SIMTIM (θ_o, m_o, r_o, C, m_a, tof, μ)
5

Figure I-5 RVVR Subrout
(Sheet 2 of 3)

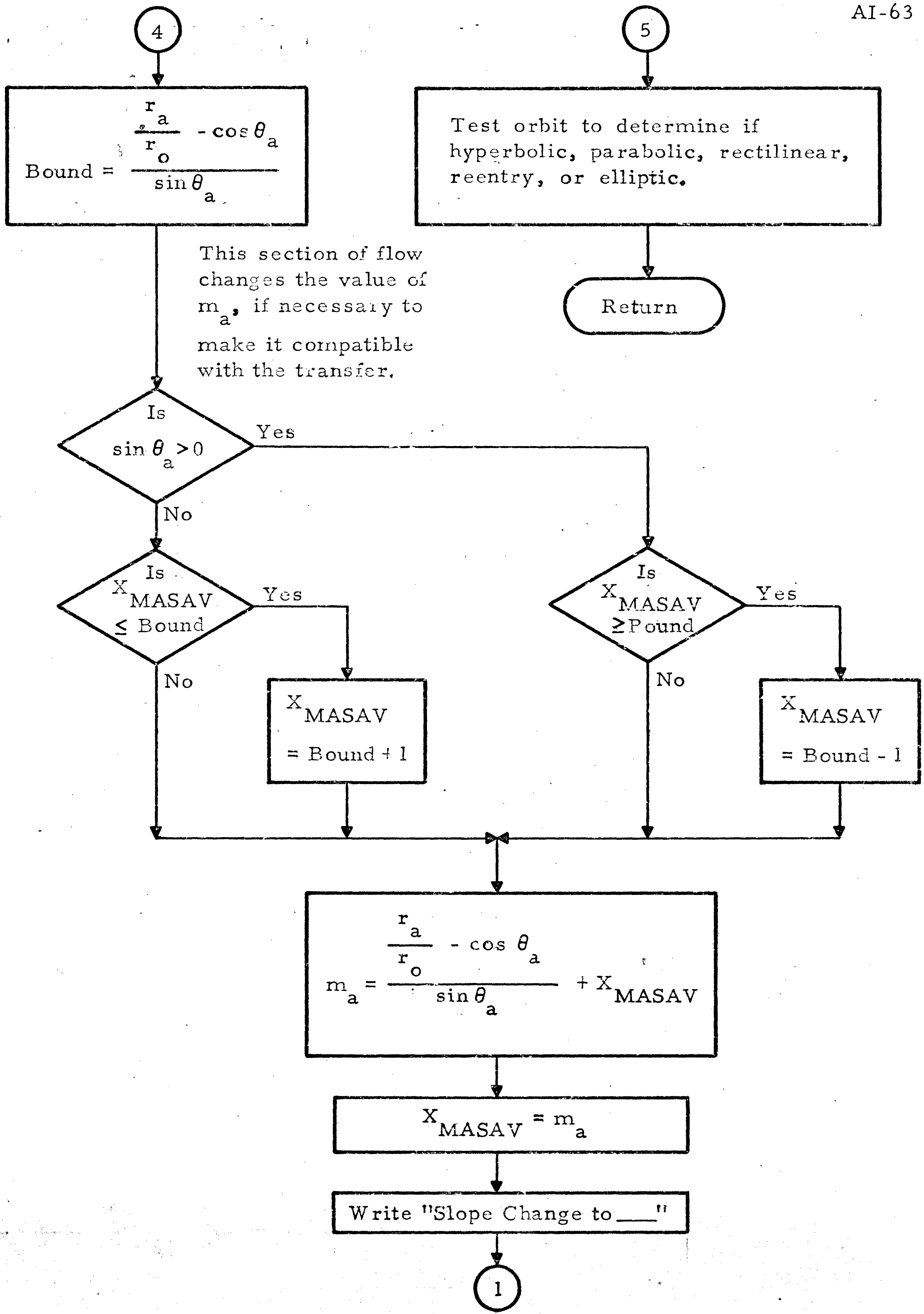
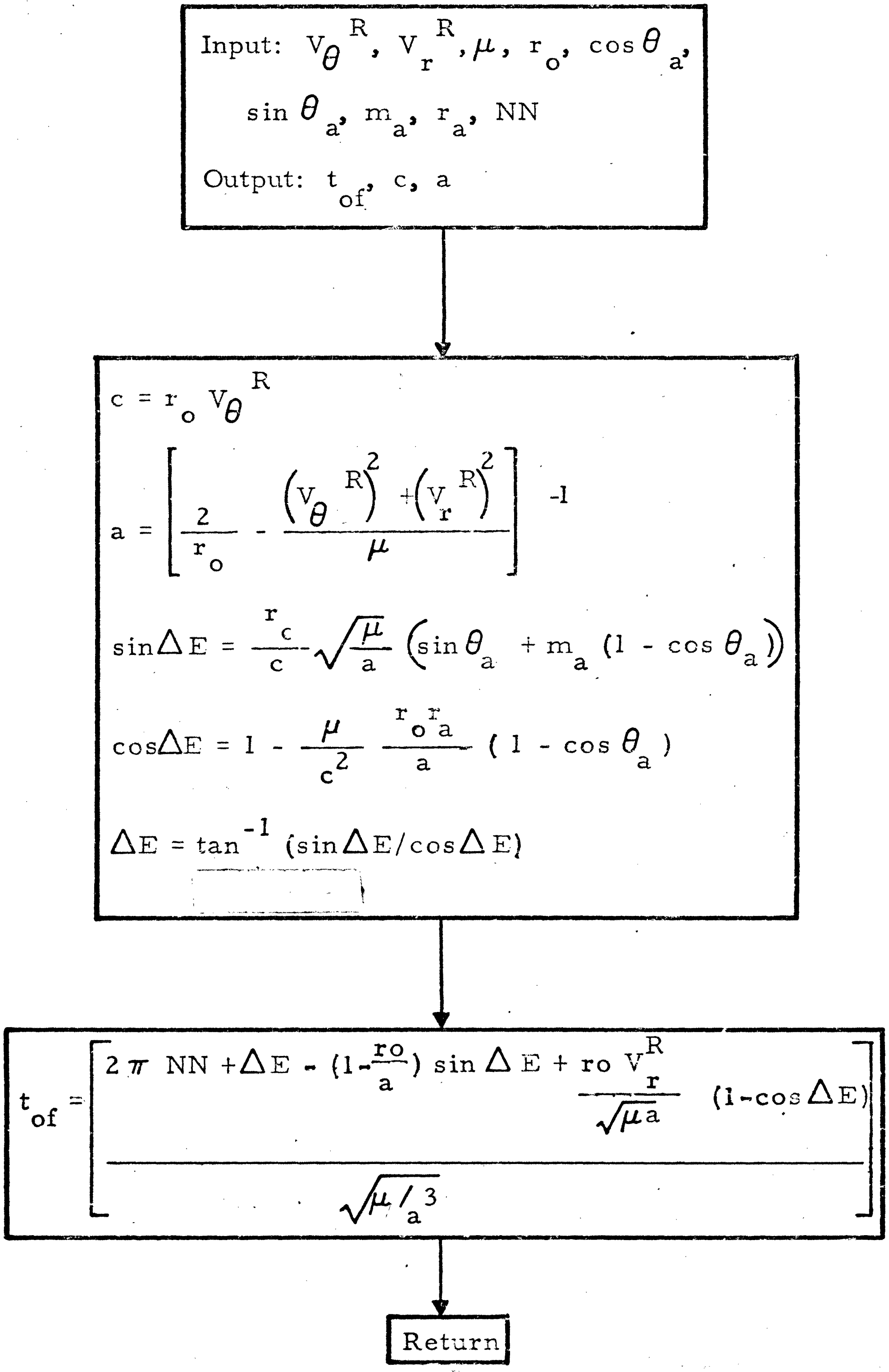

4
Bound = $\frac{\frac{r_a}{r_o} - \cos\theta_a}{\sin\theta_a}$
This section of flow changes the value of m_a, if necessary to make it compatible with the transfer.
Is $\sin\theta_a > 0$
Yes
No
Is $X_{MASAV} \leq$ Bound
Yes
No
X_{MASAV} = Bound + 1
Is $X_{MASAV} \geq$ Bound
Yes
No
X_{MASAV} = Bound - 1
$m_a = \frac{\frac{r_a}{r_o} - \cos\theta_a}{\sin\theta_a} + X_{MASAV}$
$X_{MASAV} = m_a$
Write "Slope Change to ___"
1
5
Test orbit to determine if hyperbolic, parabolic, rectilinear, reentry, or elliptic.
Return

Figure I-5 RVVR Subroutine
(Sheet 3 of 3)



Input: V_θ^R, V_r^R, μ, r_o, $\cos\theta_a$, $\sin\theta_a$, m_a, r_a, NN

Output: t_{of}, c, a

$$c = r_o V_\theta^R$$

$$a = \left[\frac{2}{r_o} - \frac{\left(V_\theta^R\right)^2 + \left(V_r^R\right)^2}{\mu}\right]^{-1}$$

$$\sin\Delta E = \frac{r_c}{c}\sqrt{\frac{\mu}{a}}\left(\sin\theta_a + m_a\,(1 - \cos\theta_a)\right)$$

$$\cos\Delta E = 1 - \frac{\mu}{c^2}\,\frac{r_o r_a}{a}\,(1 - \cos\theta_a)$$

$$\Delta E = \tan^{-1}(\sin\Delta E/\cos\Delta E)$$

$$t_{of} = \left[\frac{2\pi\ NN + \Delta E - (1 - \frac{ro}{a})\sin\Delta E + \frac{ro\ V_r^R}{\sqrt{\mu a}}\,(1 - \cos\Delta E)}{\sqrt{\mu/a^3}}\right]$$

Return

Figure I-6 Time Subroutine
(For Elliptic Transfers)

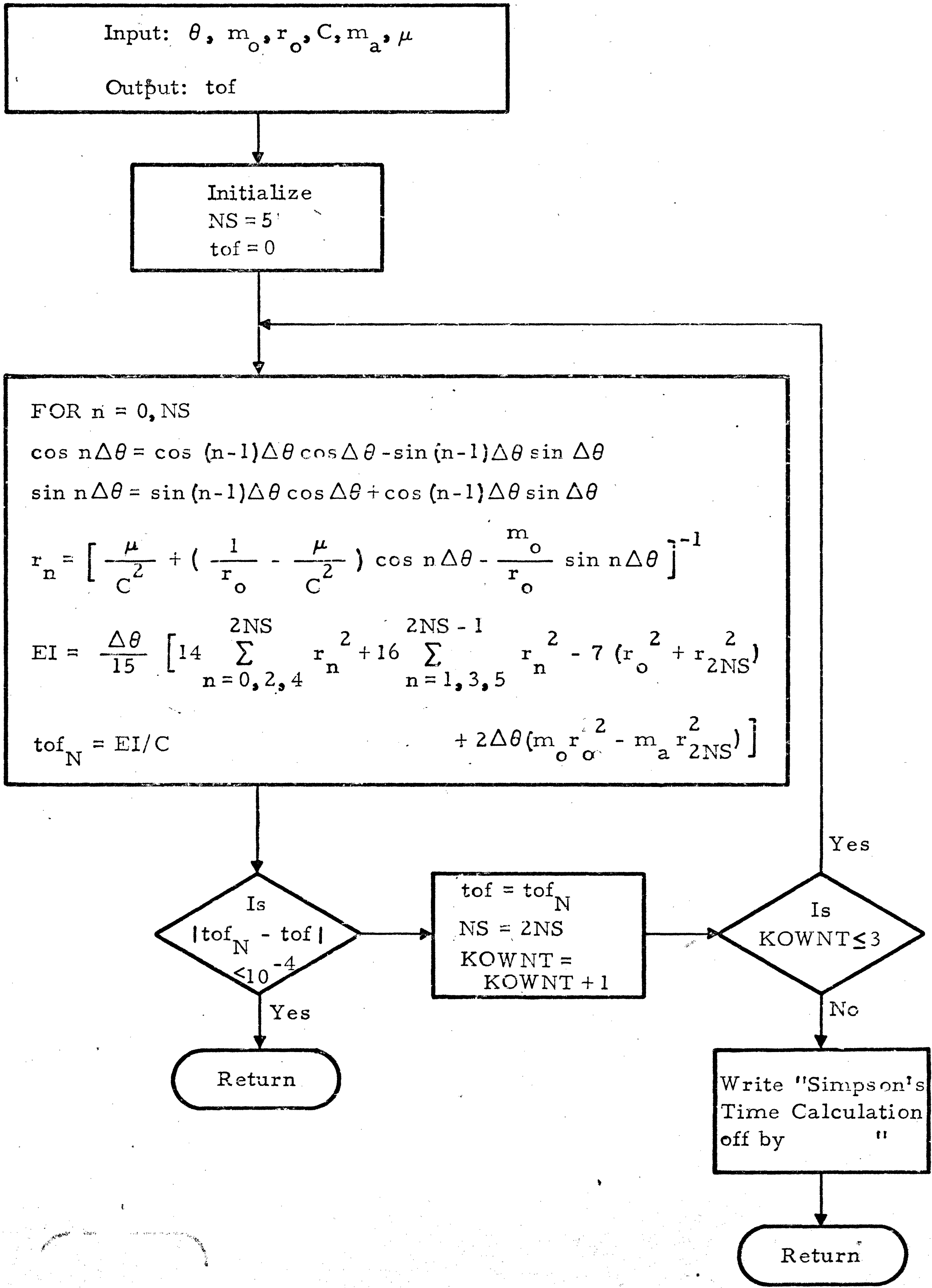

Input: θ, m_o, r_o, C, m_a, μ
Output: tof
Initialize
NS = 5
tof = 0
FOR n = 0, NS
$\cos n\Delta\theta = \cos (n-1)\Delta\theta \cos\Delta\theta - \sin (n-1)\Delta\theta \sin \Delta\theta$
$\sin n\Delta\theta = \sin (n-1)\Delta\theta \cos \Delta\theta + \cos (n-1)\Delta\theta \sin \Delta\theta$
$r_n = \left[\frac{\mu}{C^2} + \left(\frac{1}{r_o} - \frac{\mu}{C^2} \right) \cos n\Delta\theta - \frac{m_o}{r_o} \sin n\Delta\theta \right]^{-1}$
$EI = \frac{\Delta\theta}{15} \left[14 \sum_{n=0,2,4}^{2NS} r_n^2 + 16 \sum_{n=1,3,5}^{2NS-1} r_n^2 - 7 (r_o^2 + r_{2NS}^2) + 2\Delta\theta (m_o r_o^2 - m_a r_{2NS}^2) \right]$
$tof_N = EI/C$
Is $|tof_N - tof| < 10^{-4}$
Yes
Return
$tof = tof_N$
NS = 2NS
KOWNT = KOWNT + 1
Is KOWNT ≤ 3
Yes
No
Write "Simpson's Time Calculation off by "
Return

Figure I-7 SIMTIM Subroutine

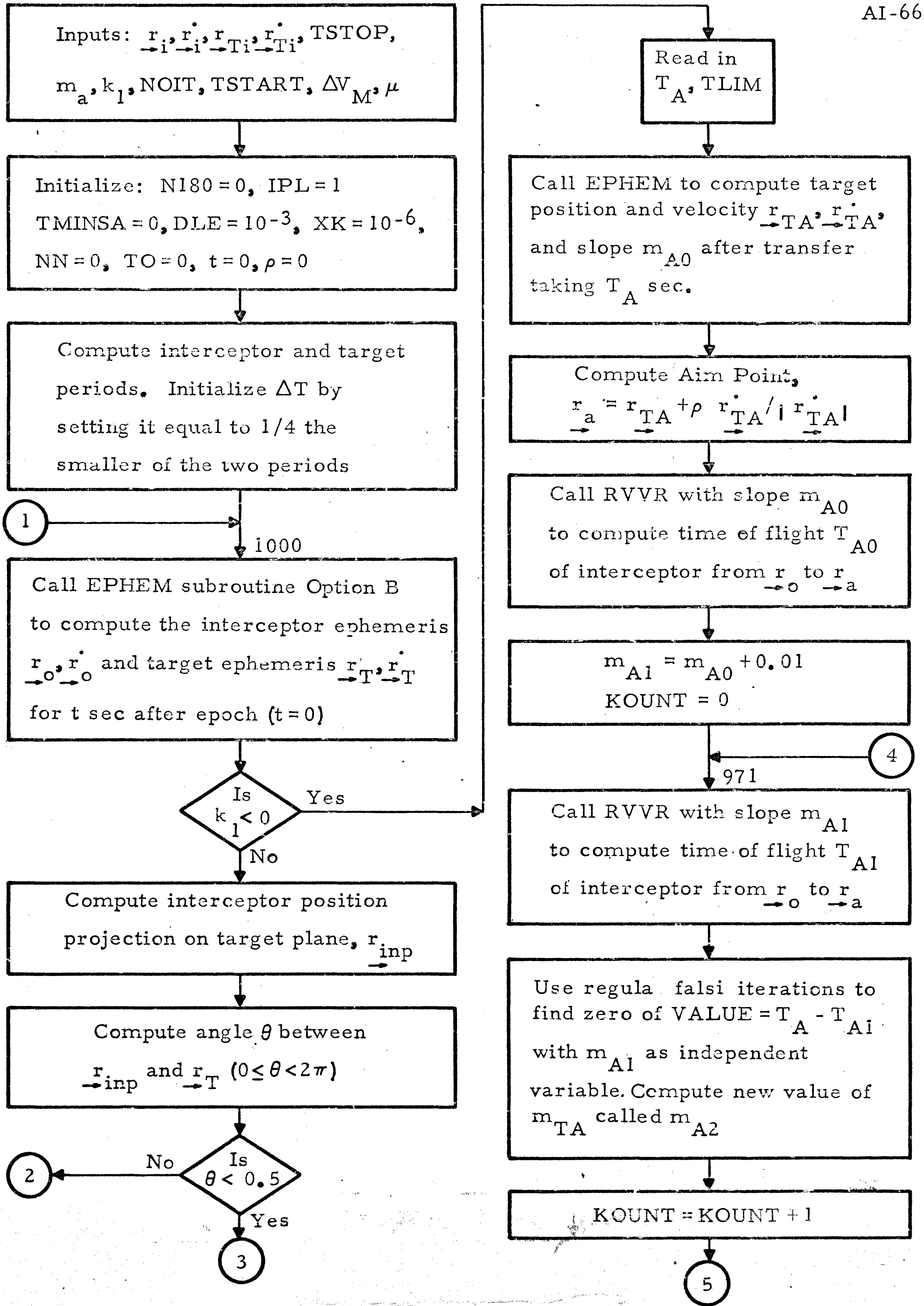

Inputs: $\underset{\rightarrow}{r}_i$, $\dot{\underset{\rightarrow}{r}}_i$, $\underset{\rightarrow}{r}_{Ti}$, $\dot{\underset{\rightarrow}{r}}_{Ti}$, TSTOP, m_a, k_1, NOIT, TSTART, ΔV_M, μ
Initialize: N180 = 0, IPL = 1 TMINSA = 0, DLE = 10^{-3}, XK = 10^{-6}, NN = 0, TO = 0, t = 0, $\rho = 0$
Compute interceptor and target periods. Initialize ΔT by setting it equal to 1/4 the smaller of the two periods
1
1000
Call EPHEM subroutine Option B to compute the interceptor ephemeris $\underset{\rightarrow}{r}_o$, $\dot{\underset{\rightarrow}{r}}_o$ and target ephemeris $\underset{\rightarrow}{r}_T$, $\dot{\underset{\rightarrow}{r}}_T$ for t sec after epoch (t = 0)
Is $k_1 < 0$
Yes
No
Compute interceptor position projection on target plane, $\underset{\rightarrow}{r}_{inp}$
Compute angle θ between $\underset{\rightarrow}{r}_{inp}$ and $\underset{\rightarrow}{r}_T$ $(0 \leq \theta < 2\pi)$
Is $\theta < 0.5$
No
2
Yes
3
Read in T_A, TLIM
Call EPHEM to compute target position and velocity $\underset{\rightarrow}{r}_{TA}$, $\dot{\underset{\rightarrow}{r}}_{TA}$, and slope m_{A0} after transfer taking T_A sec.
Compute Aim Point, $\underset{\rightarrow}{r}_a = \underset{\rightarrow}{r}_{TA} + \rho \; \dot{\underset{\rightarrow}{r}}_{TA} / | \dot{\underset{\rightarrow}{r}}_{TA} |$
Call RVVR with slope m_{A0} to compute time of flight T_{A0} of interceptor from $\underset{\rightarrow}{r}_o$ to $\underset{\rightarrow}{r}_a$
$m_{A1} = m_{A0} + 0.01$ KOUNT = 0
4
971
Call RVVR with slope m_{A1} to compute time of flight T_{A1} of interceptor from $\underset{\rightarrow}{r}_o$ to $\underset{\rightarrow}{r}_a$
Use regula falsi iterations to find zero of VALUE = $T_A - T_{A1}$ with m_{A1} as independent variable. Compute new value of m_{TA} called m_{A2}
KOUNT = KOUNT + 1
5

Figure I-8 RENDI Subroutine
(Sheet 1 of 4)

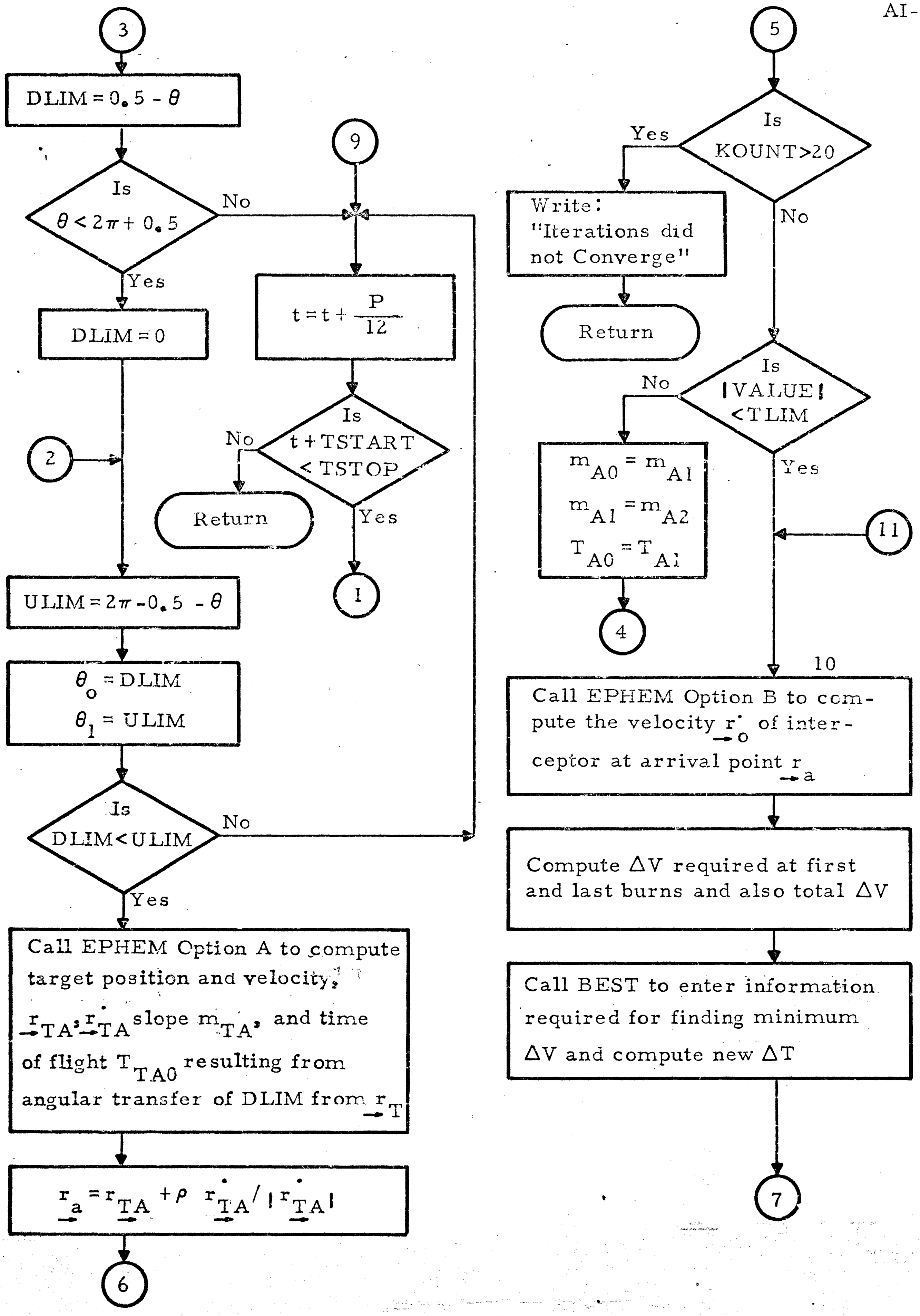

3
$DLIM = 0.5 - \theta$
Is $\theta < 2\pi + 0.5$
No
Yes
$DLIM = 0$
2
$ULIM = 2\pi - 0.5 - \theta$
$\theta_o = DLIM$
$\theta_1 = ULIM$
Is DLIM<ULIM
No
Yes
Call EPHEM Option A to compute target position and velocity, $\underrightarrow{r}_{TA}$, $\dot{\underrightarrow{r}}_{TA}$ slope m_{TA}, and time of flight T_{TA0} resulting from angular transfer of DLIM from $\underrightarrow{r}_T$
$\underrightarrow{r}_a = \underrightarrow{r}_{TA} + \rho \; \dot{\underrightarrow{r}}_{TA} / |\dot{\underrightarrow{r}}_{TA}|$
6
9
$t = t + \frac{P}{12}$
Is t + TSTART < TSTOP
No
Return
Yes
1
5
Is KOUNT>20
Yes
Write: "Iterations did not Converge"
Return
No
Is |VALUE| <TLIM
No
$m_{A0} = m_{A1}$
$m_{A1} = m_{A2}$
$T_{A0} = T_{A1}$
4
Yes
11
10
Call EPHEM Option B to compute the velocity $\dot{\underrightarrow{r}}_o$ of interceptor at arrival point $\underrightarrow{r}_a$
Compute ΔV required at first and last burns and also total ΔV
Call BEST to enter information required for finding minimum ΔV and compute new ΔT
7

Figure I-8 RENDI Subroutine (Sheet 2 of 4)

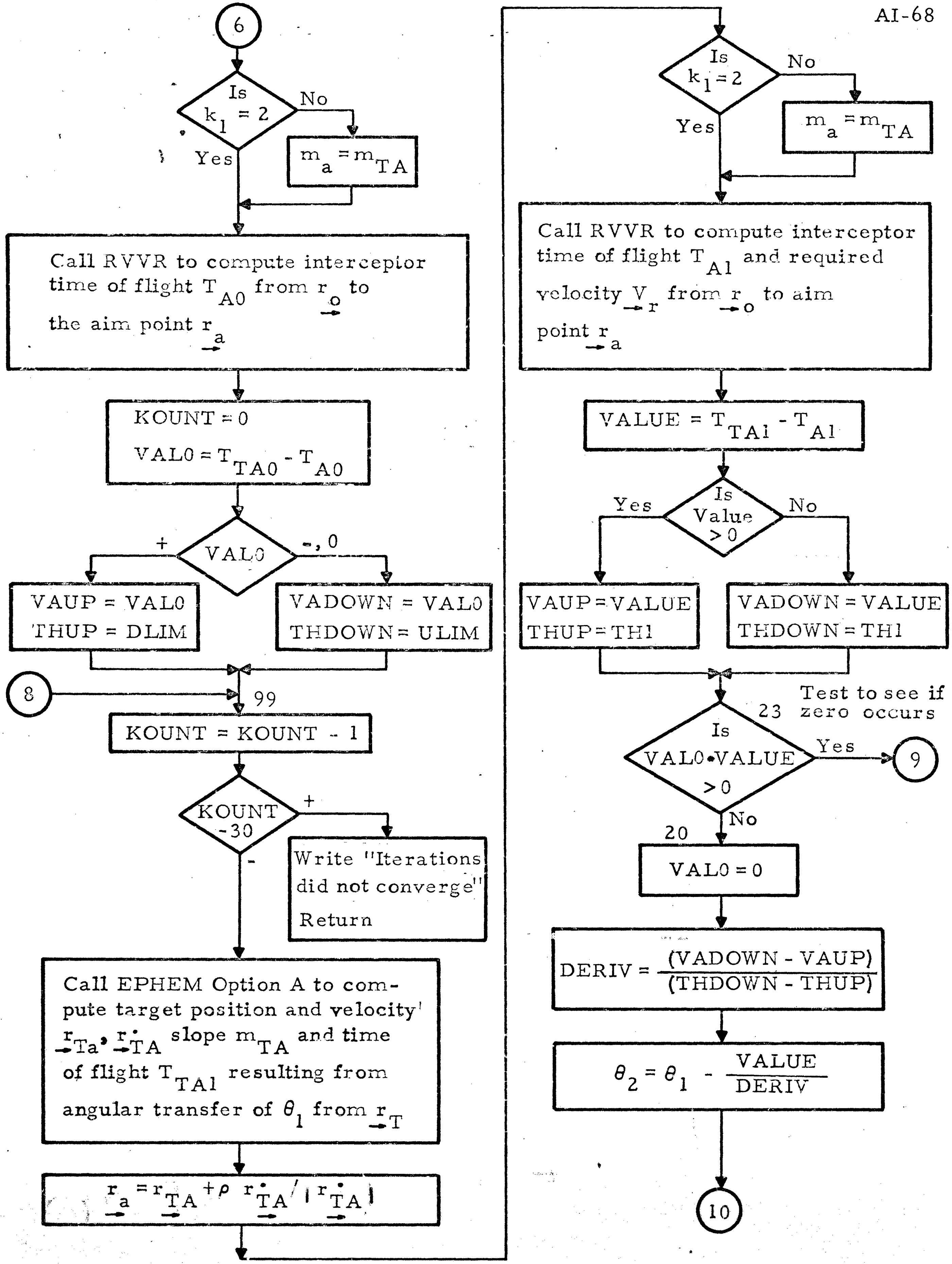

6
Is $k_1 = 2$
No
Yes
$m_a = m_{TA}$
Call RVVR to compute interceptor time of flight T_{A0} from $\underset{\rightarrow}{r}_o$ to the aim point $\underset{\rightarrow}{r}_a$
KOUNT = 0
$VAL0 = T_{TA0} - T_{A0}$
+
VAL0
-, 0
VAUP = VAL0
THUP = DLIM
VADOWN = VAL0
THDOWN = ULIM
8
99
KOUNT = KOUNT - 1
KOUNT -30
+
-
Write "Iterations did not converge"
Return
Call EPHEM Option A to compute target position and velocity $\underset{\rightarrow}{r}_{Ta}$, $\underset{\rightarrow}{\dot{r}}_{TA}$ slope m_{TA} and time of flight T_{TA1} resulting from angular transfer of θ_1 from $\underset{\rightarrow}{r}_T$
$\underset{\rightarrow}{r}_a = \underset{\rightarrow}{r}_{TA} + \rho\ \underset{\rightarrow}{\dot{r}}_{TA} / |\underset{\rightarrow}{\dot{r}}_{TA}|$
Is $k_1 = 2$
No
Yes
$m_a = m_{TA}$
Call RVVR to compute interceptor time of flight T_{A1} and required velocity $\underset{\rightarrow}{V}_r$ from $\underset{\rightarrow}{r}_o$ to aim point $\underset{\rightarrow}{r}_a$
$VALUE = T_{TA1} - T_{A1}$
Is Value > 0
Yes
No
VAUP = VALUE
THUP = TH1
VADOWN = VALUE
THDOWN = TH1
Test to see if zero occurs
23
Is VAL0•VALUE > 0
Yes
9
No
20
VAL0 = 0
$DERIV = \frac{(VADOWN - VAUP)}{(THDOWN - THUP)}$
$\theta_2 = \theta_1 - \frac{VALUE}{DERIV}$
10

Figure I-8 RENDI Subroutine
(Sheet 3 of 4)

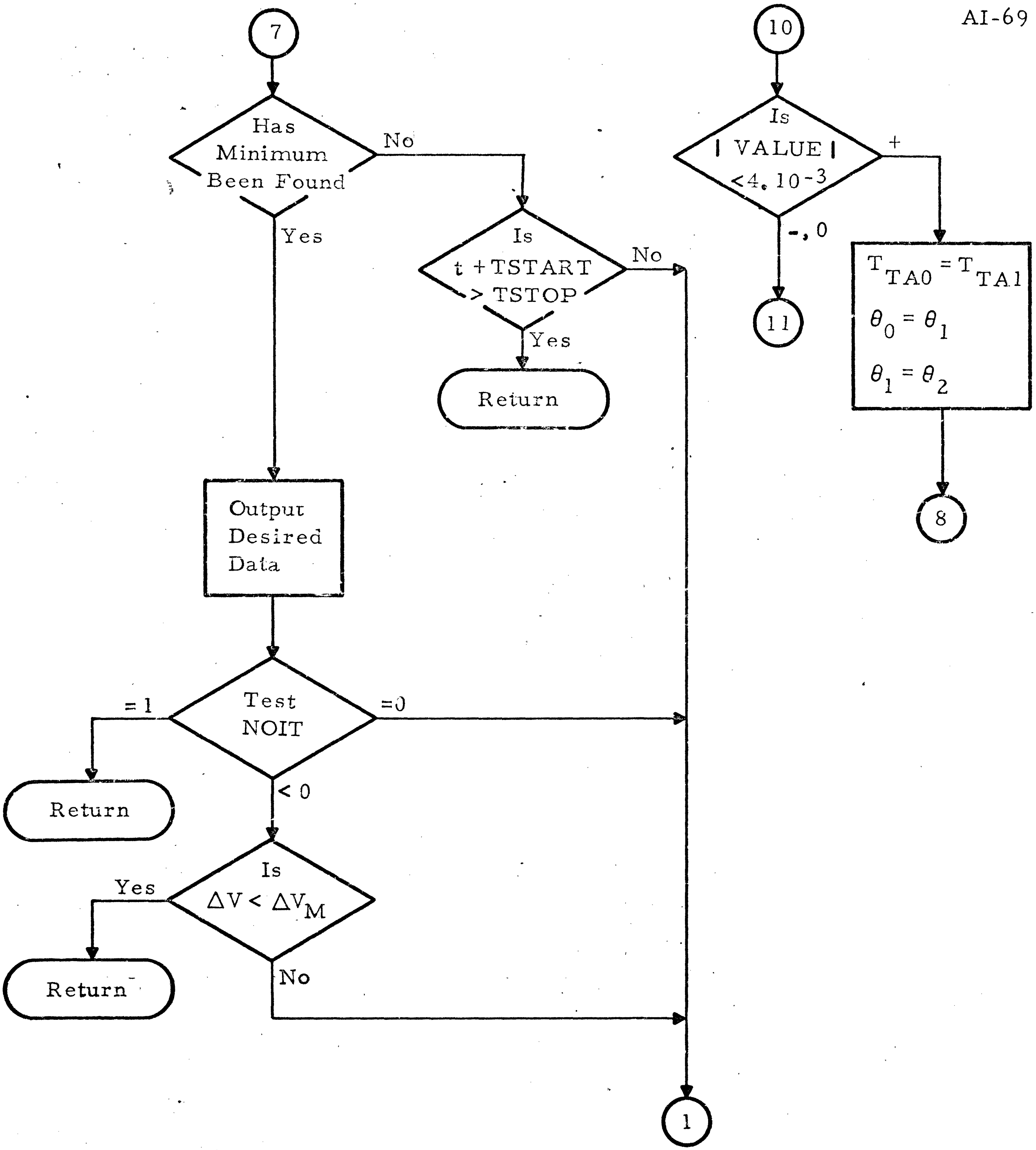


Figure I-8 RENDI Subroutine

(Sheet 4 of 4)

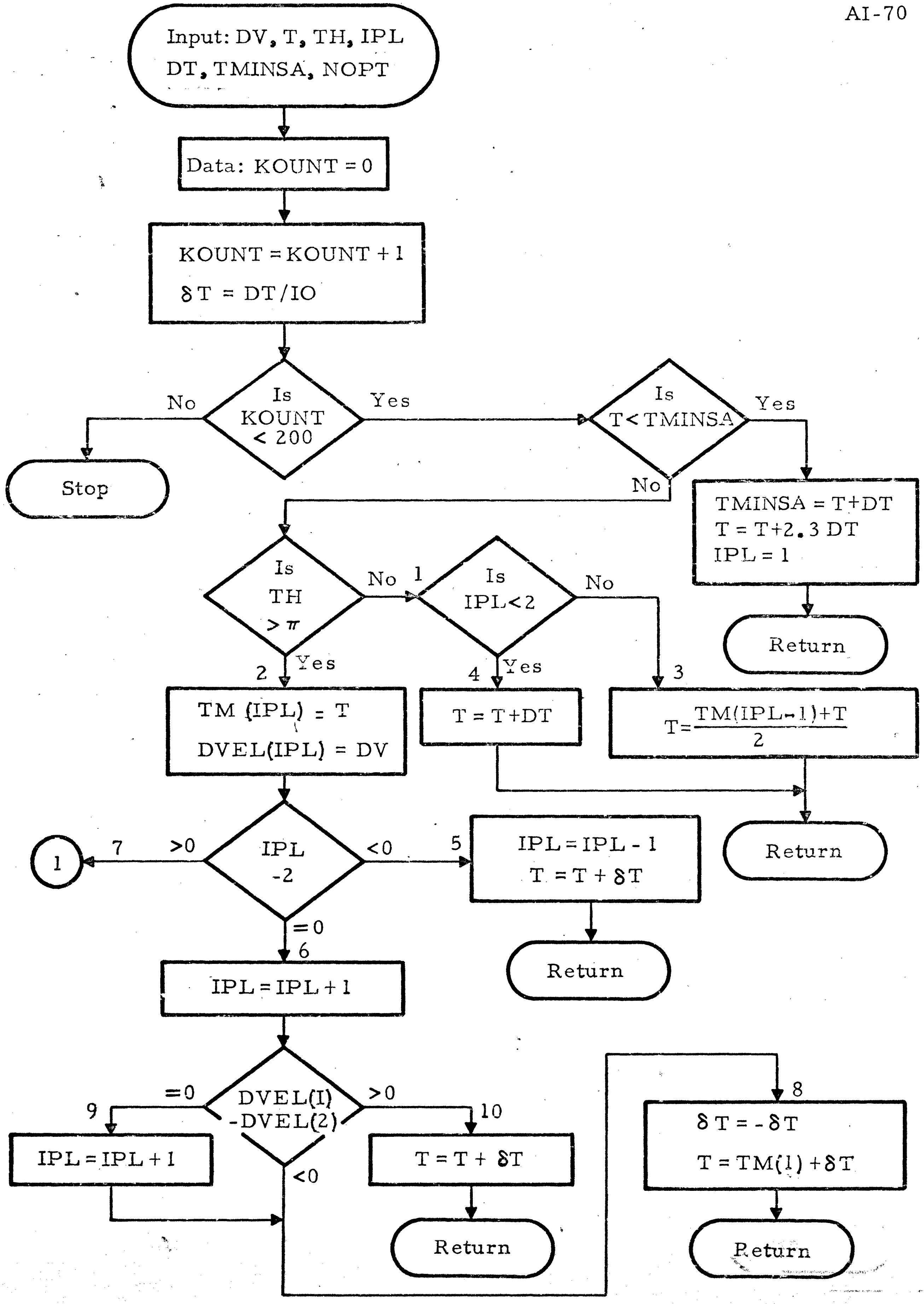


Figure I-9 BEST Subroutine
(Sheet 1 of 3)

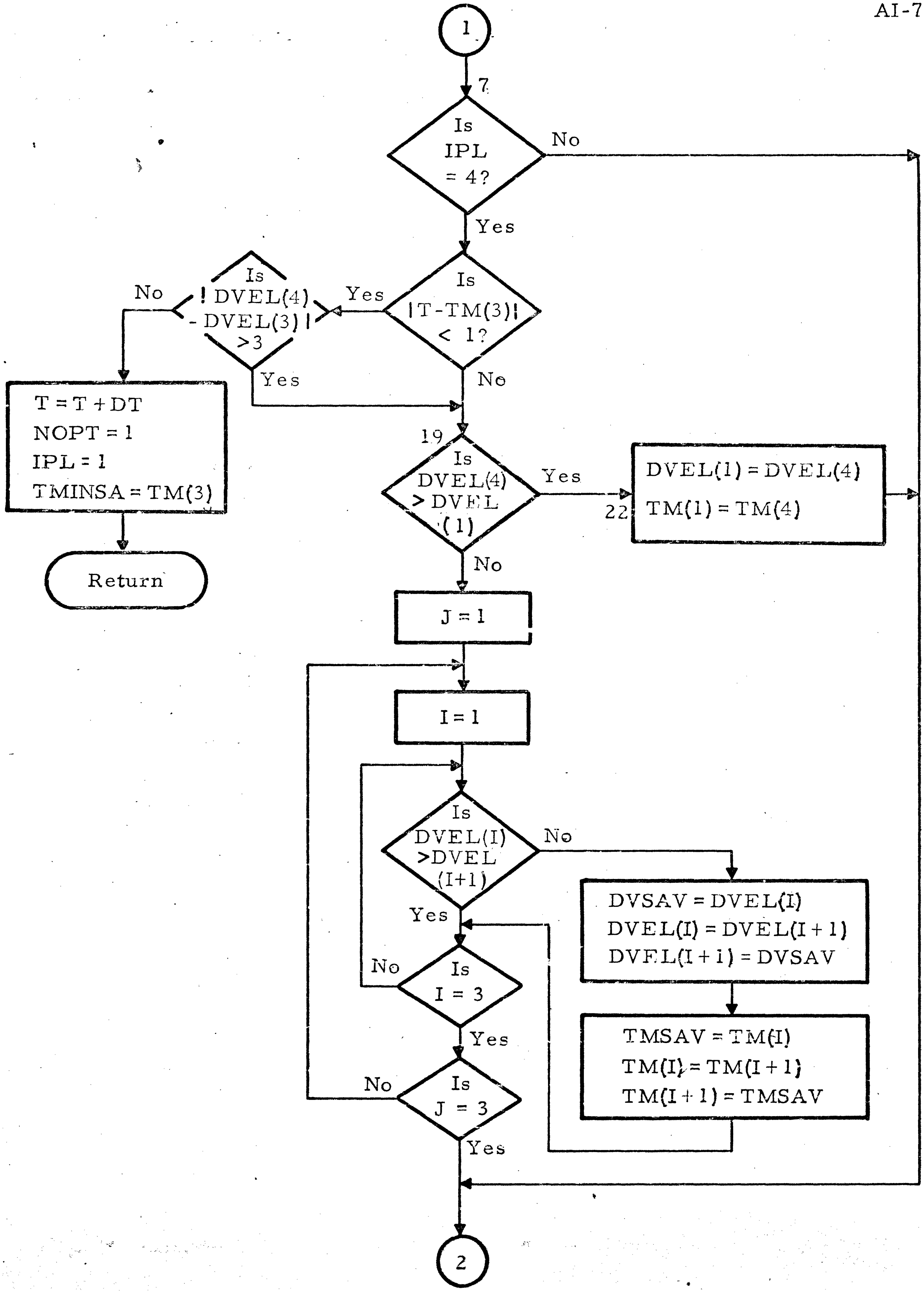


Figure I-9 BEST Subroutine
(Sheet 2 of 3)

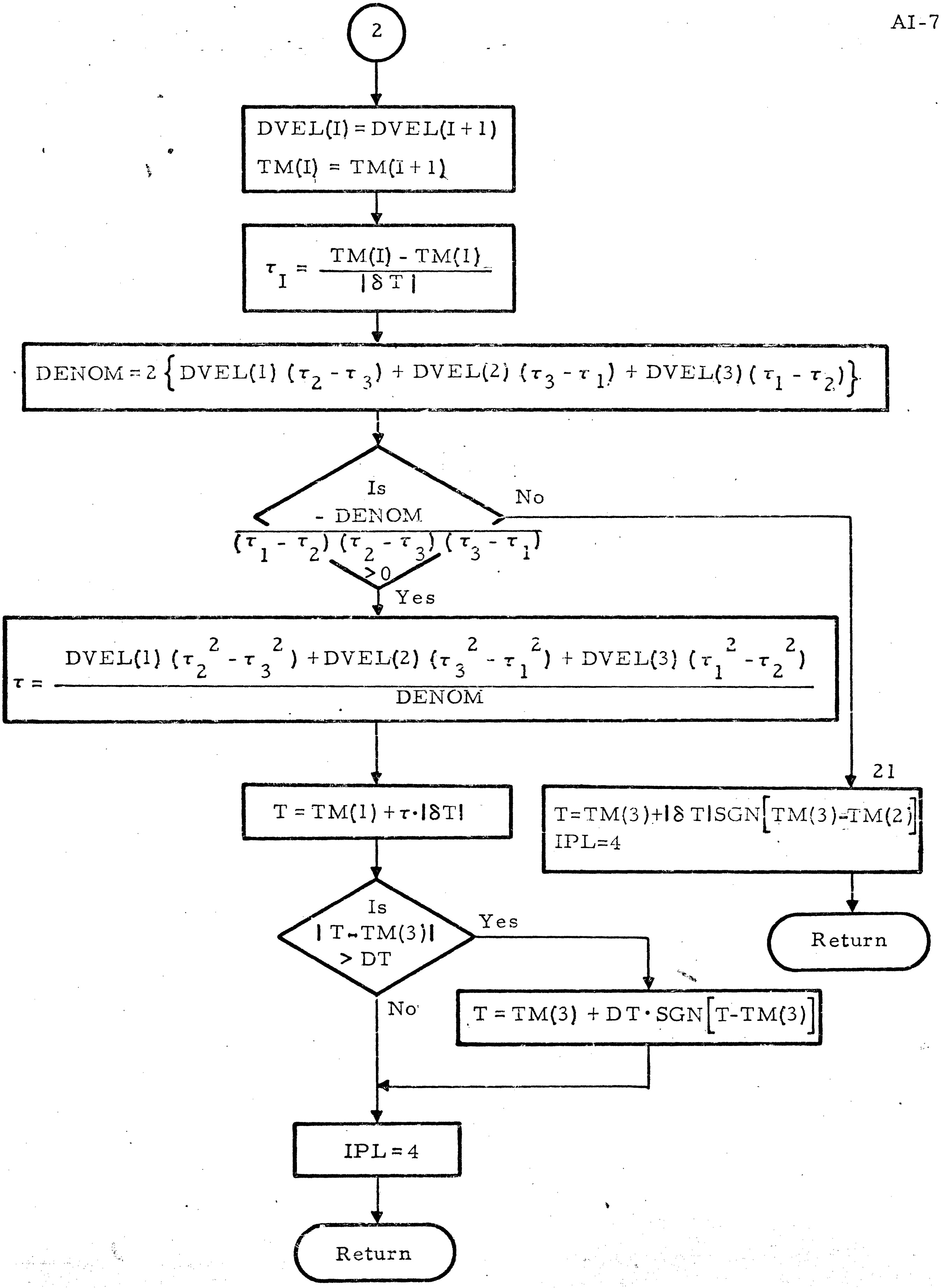
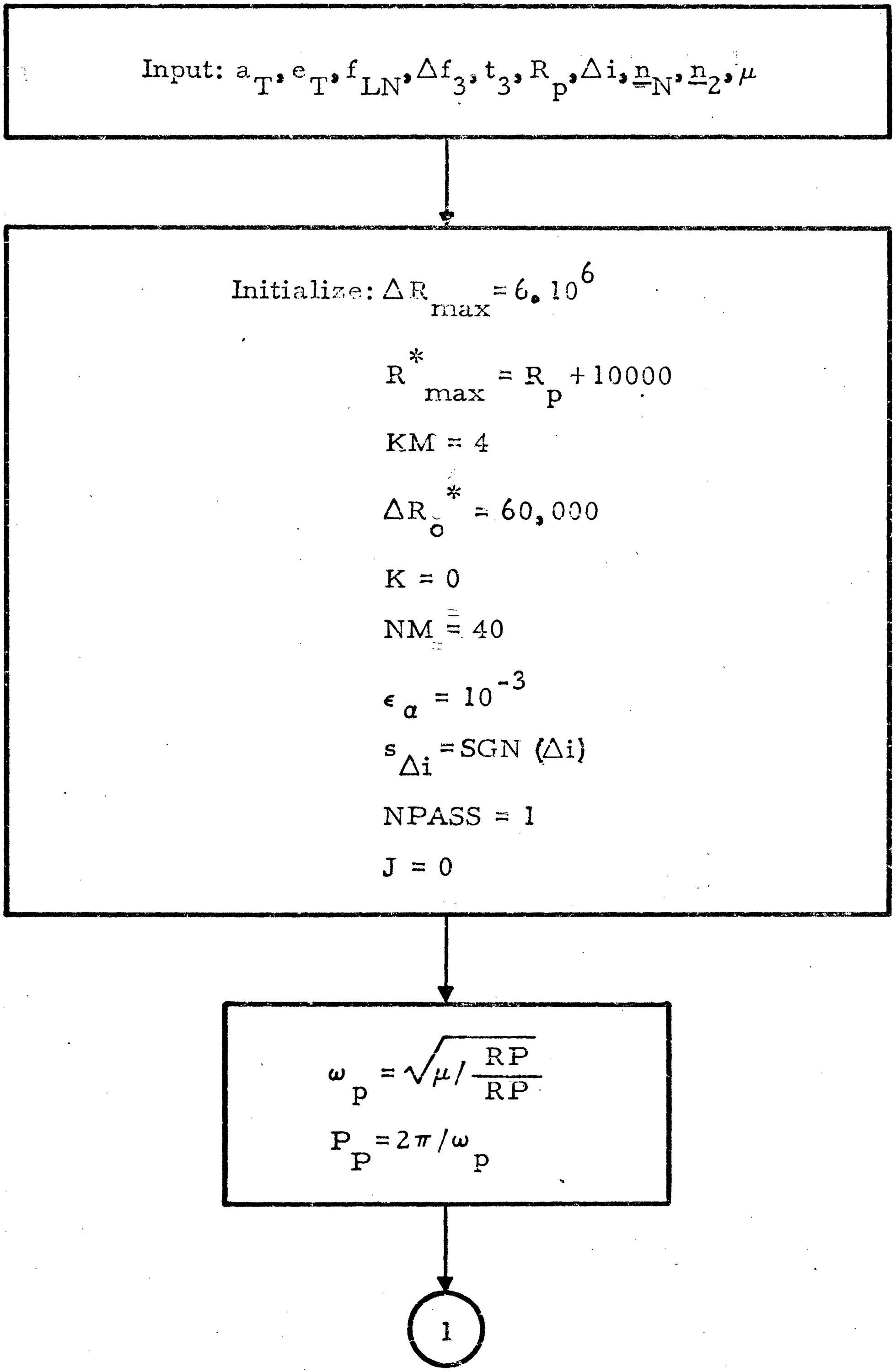

2
DVEL(I) = DVEL(I + 1)
TM(I) = TM(I + 1)
$\tau_I = \frac{TM(I) - TM(1)}{|\delta T|}$
DENOM = 2 $\{DVEL(1)(\tau_2 - \tau_3) + DVEL(2)(\tau_3 - \tau_1) + DVEL(3)(\tau_1 - \tau_2)\}$
Is $\frac{-DENOM}{(\tau_1 - \tau_2)(\tau_2 - \tau_3)(\tau_3 - \tau_1)} > 0$
No
Yes
$\tau = \frac{DVEL(1)(\tau_2^2 - \tau_3^2) + DVEL(2)(\tau_3^2 - \tau_1^2) + DVEL(3)(\tau_1^2 - \tau_2^2)}{DENOM}$
21
T = TM(1) + $\tau \cdot |\delta T|$
T=TM(3)+$|\delta T|$SGN[TM(3)-TM(2)]
IPL=4
Return
Is $|T - TM(3)| > DT$
Yes
No
T = TM(3) + DT · SGN[T-TM(3)]
IPL = 4
Return

Figure I-9 BEST Subroutine

(Sheet 3 of 3)



Input: $a_T, e_T, f_{LN}, \Delta f_3, t_3, R_p, \Delta i, \underline{n}_N, \underline{n}_2, \mu$

Initialize: $\Delta R_{max} = 6.10^6$

$R^*_{max} = R_p + 10000$

$KM = 4$

$\Delta R_o^* = 60,000$

$K = 0$

$NM = 40$

$\epsilon_\alpha = 10^{-3}$

$s_{\Delta i} = SGN\ (\Delta i)$

$NPASS = 1$

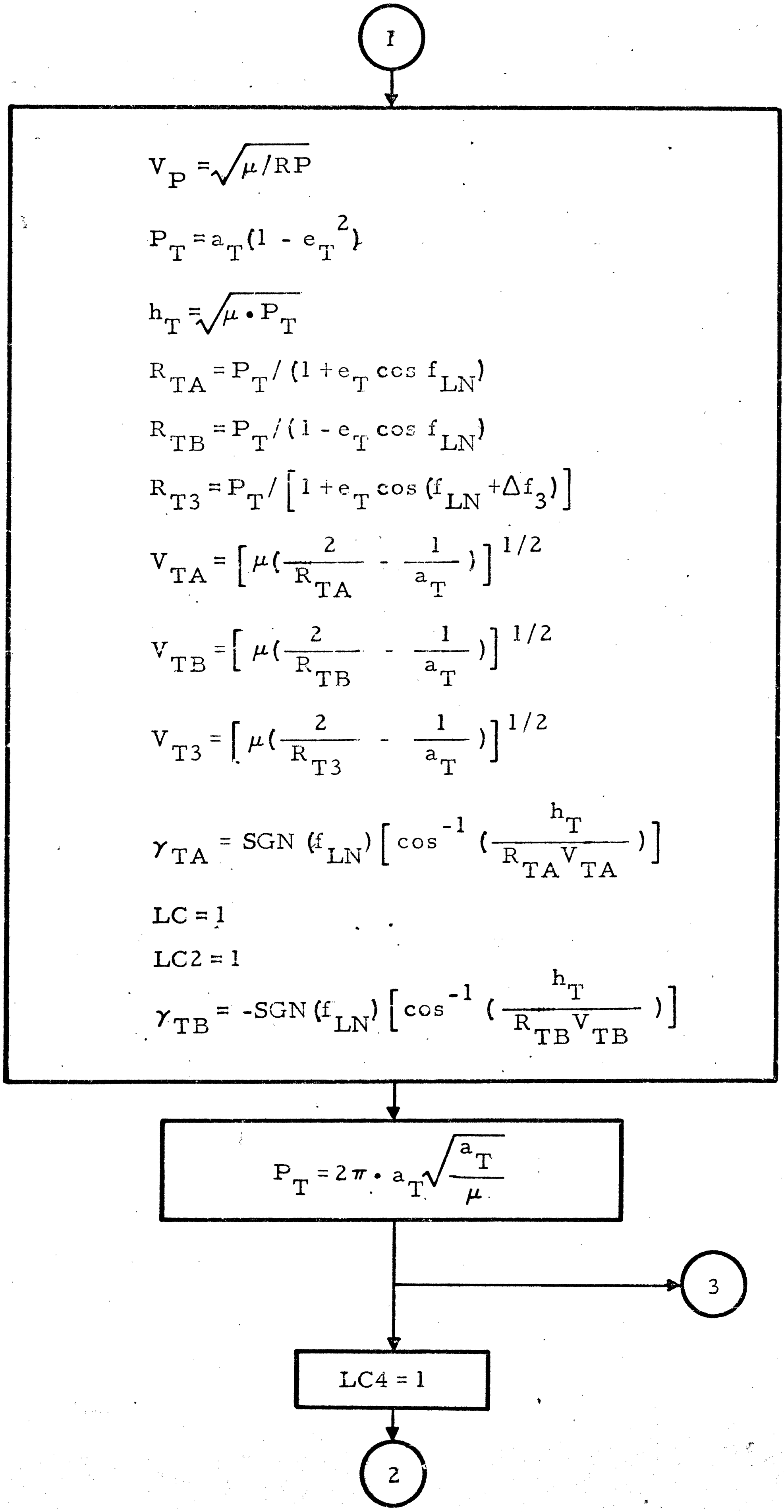
$J = 0$

$\omega_p = \sqrt{\mu / \frac{RP}{RP}}$

$P_P = 2\pi / \omega_p$

Figure I-10 VPG Subroutine

(Sheet 1 of 13)



$$V_P = \sqrt{\mu / RP}$$

$$P_T = a_T(1 - e_T^{2})$$

$$h_T = \sqrt{\mu \cdot P_T}$$

$$R_{TA} = P_T / (1 + e_T \cos f_{LN})$$

$$R_{TB} = P_T / (1 - e_T \cos f_{LN})$$

$$R_{T3} = P_T / \left[1 + e_T \cos (f_{LN} + \Delta f_3) \right]$$

$$V_{TA} = \left[\mu \left(\frac{2}{R_{TA}} - \frac{1}{a_T} \right) \right]^{1/2}$$

$$V_{TB} = \left[\mu \left(\frac{2}{R_{TB}} - \frac{1}{a_T} \right) \right]^{1/2}$$

$$V_{T3} = \left[\mu \left(\frac{2}{R_{T3}} - \frac{1}{a_T} \right) \right]^{1/2}$$

$$\gamma_{TA} = \mathrm{SGN}\,(f_{LN}) \left[\cos^{-1} \left(\frac{h_T}{R_{TA} V_{TA}} \right) \right]$$

$$LC = 1$$

$$LC2 = 1$$

$$\gamma_{TB} = -\mathrm{SGN}\,(f_{LN}) \left[\cos^{-1} \left(\frac{h_T}{R_{TB} V_{TB}} \right) \right]$$

$$P_T = 2\pi \cdot a_T \sqrt{\frac{a_T}{\mu}}$$

$$LC4 = 1$$

Figure I-10 VPG Subroutine

(Sheet 2 of 13)

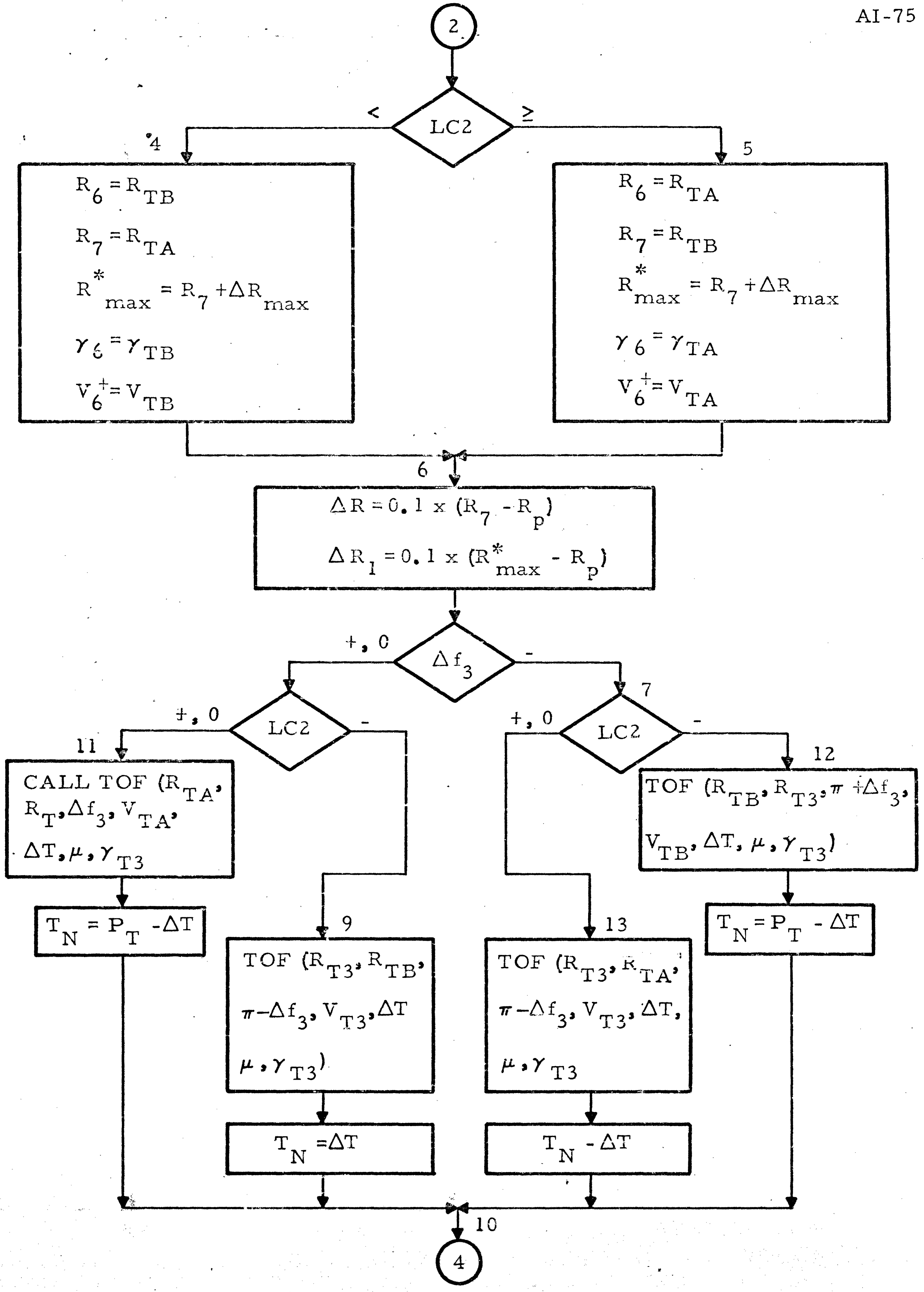

2
LC2
<
≥
4
$R_6 = R_{TB}$
$R_7 = R_{TA}$
$R^*_{max} = R_7 + \Delta R_{max}$
$\gamma_6 = \gamma_{TB}$
$V_6^+ = V_{TB}$
5
$R_6 = R_{TA}$
$R_7 = R_{TB}$
$R^*_{max} = R_7 + \Delta R_{max}$
$\gamma_6 = \gamma_{TA}$
$V_6^+ = V_{TA}$
6
$\Delta R = 0.1 \times (R_7 - R_p)$
$\Delta R_1 = 0.1 \times (R^*_{max} - R_p)$
Δf_3
+, 0
-
7
LC2
+, 0
-
LC2
+, 0
-
11
CALL TOF (R_{TA}, R_T, Δf_3, V_{TA}, ΔT, μ, γ_{T3}
$T_N = P_T - \Delta T$
9
TOF (R_{T3}, R_{TB}, $\pi - \Delta f_3$, V_{T3}, ΔT μ, γ_{T3})
$T_N = \Delta T$
13
TOF (R_{T3}, R_{TA}, $\pi - \Delta f_3$, V_{T3}, ΔT, μ, γ_{T3}
$T_N - \Delta T$
12
TOF (R_{TB}, R_{T3}, $\pi + \Delta f_3$, V_{TB}, ΔT, μ, γ_{T3})
$T_N = P_T - \Delta T$
10
4

Figure I-10 VPG Subroutine

(Sheet 3 of 13)

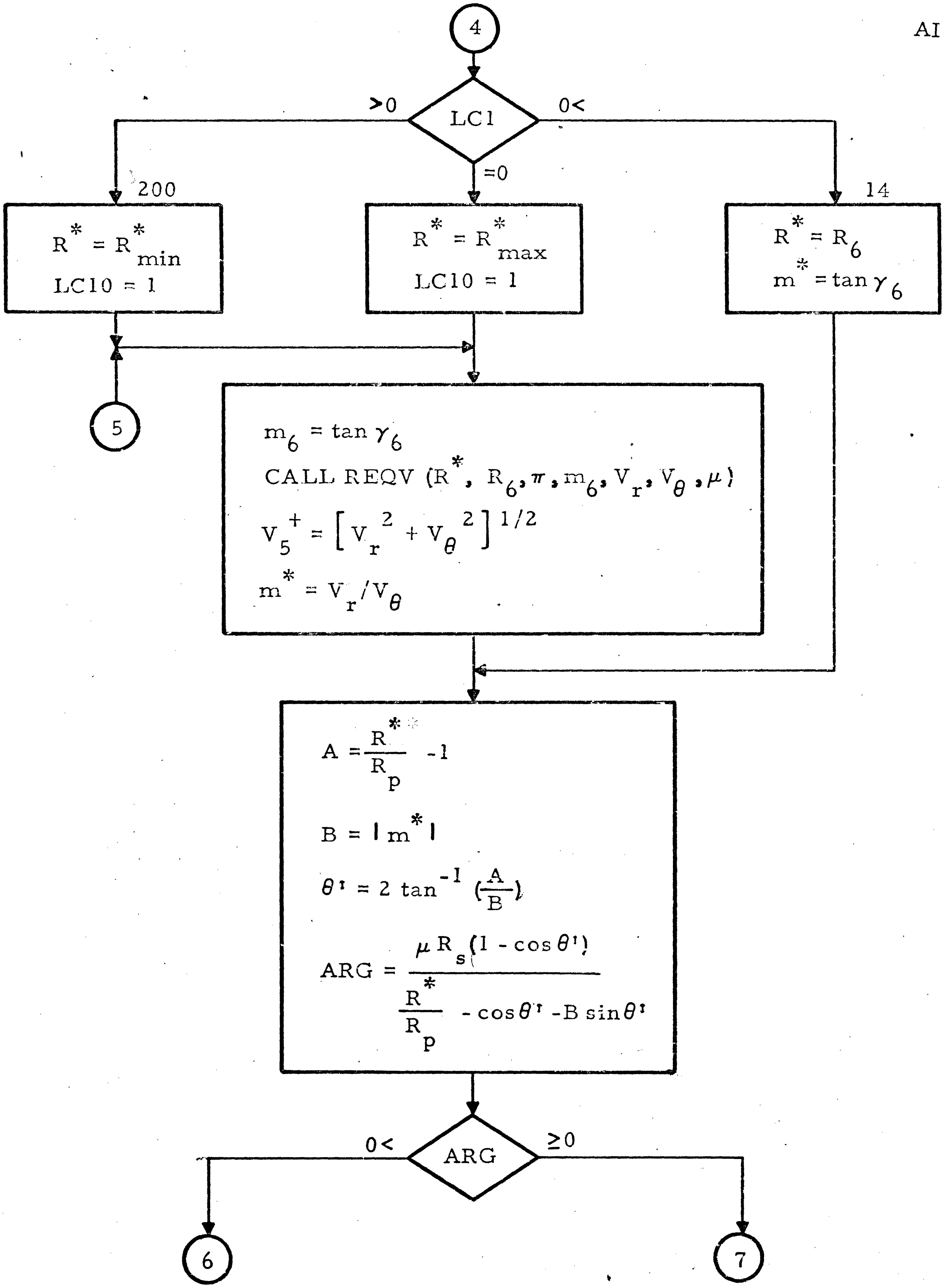

4
LC1
>0
=0
0<
200
$R^* = R^*_{min}$
LC10 = 1
$R^* = R^*_{max}$
LC10 = 1
14
$R^* = R_6$
$m^* = \tan \gamma_6$
5
$m_6 = \tan \gamma_6$
CALL REQV $(R^*, R_6, \pi, m_6, V_r, V_\theta, \mu)$
$V_5^+ = \left[V_r^2 + V_\theta^2 \right]^{1/2}$
$m^* = V_r / V_\theta$
$A = \frac{R^*}{R_p} - 1$
$B = |m^*|$
$\theta' = 2 \tan^{-1} (\frac{A}{B})$
$ARG = \frac{\mu R_s (1 - \cos \theta')}{\frac{R^*}{R_p} - \cos \theta' - B \sin \theta'}$
ARG
0<
≥0
6
7

Figure I-10 VPG Subroutine

(Sheet 4 of 13)

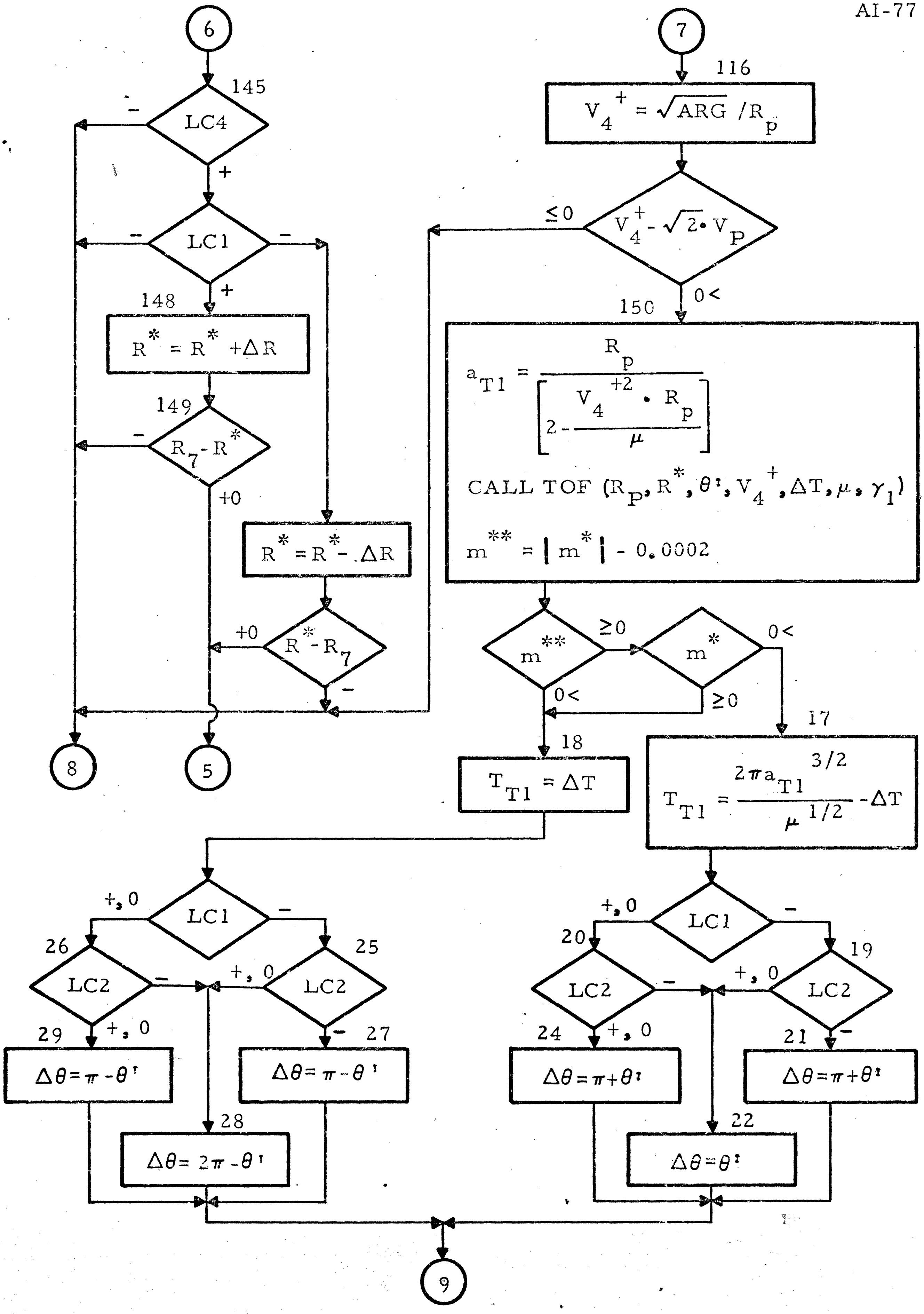

6
145
LC4
-
+
LC1
-
-
+
148
$R^* = R^* + \Delta R$
149
$R_7 - R^*$
-
+0
$R^* = R^* - \Delta R$
$R^* - R_7$
+0
-
8
5
7
116
$V_4^+ = \sqrt{ARG} / R_p$
$V_4^+ - \sqrt{2} \cdot V_P$
≤0
0<
150
$a_{T1} = \frac{R_p}{\left[2 - \frac{V_4^{+2} \cdot R_p}{\mu}\right]}$
CALL TOF (R_P, R^*, θ', V_4^+, ΔT, μ, γ_1)
$m^{**} = |m^*| - 0.0002$
m^{**}
≥0
m^*
0<
0<
≥0
18
$T_{T1} = \Delta T$
17
$T_{T1} = \frac{2\pi a_{T1}^{3/2}}{\mu^{1/2}} - \Delta T$
LC1
+, 0
-
26
LC2
-
+, 0
25
LC2
29
+, 0
-
27
$\Delta\theta = \pi - \theta'$
$\Delta\theta = \pi - \theta'$
28
$\Delta\theta = 2\pi - \theta'$
LC1
+, 0
-
20
LC2
-
+, 0
19
LC2
24
+, 0
21
-
$\Delta\theta = \pi + \theta'$
$\Delta\theta = \pi + \theta'$
22
$\Delta\theta = \theta'$
9

Figure I-10 VPG Subroutine

(Sheet 5 of 13)

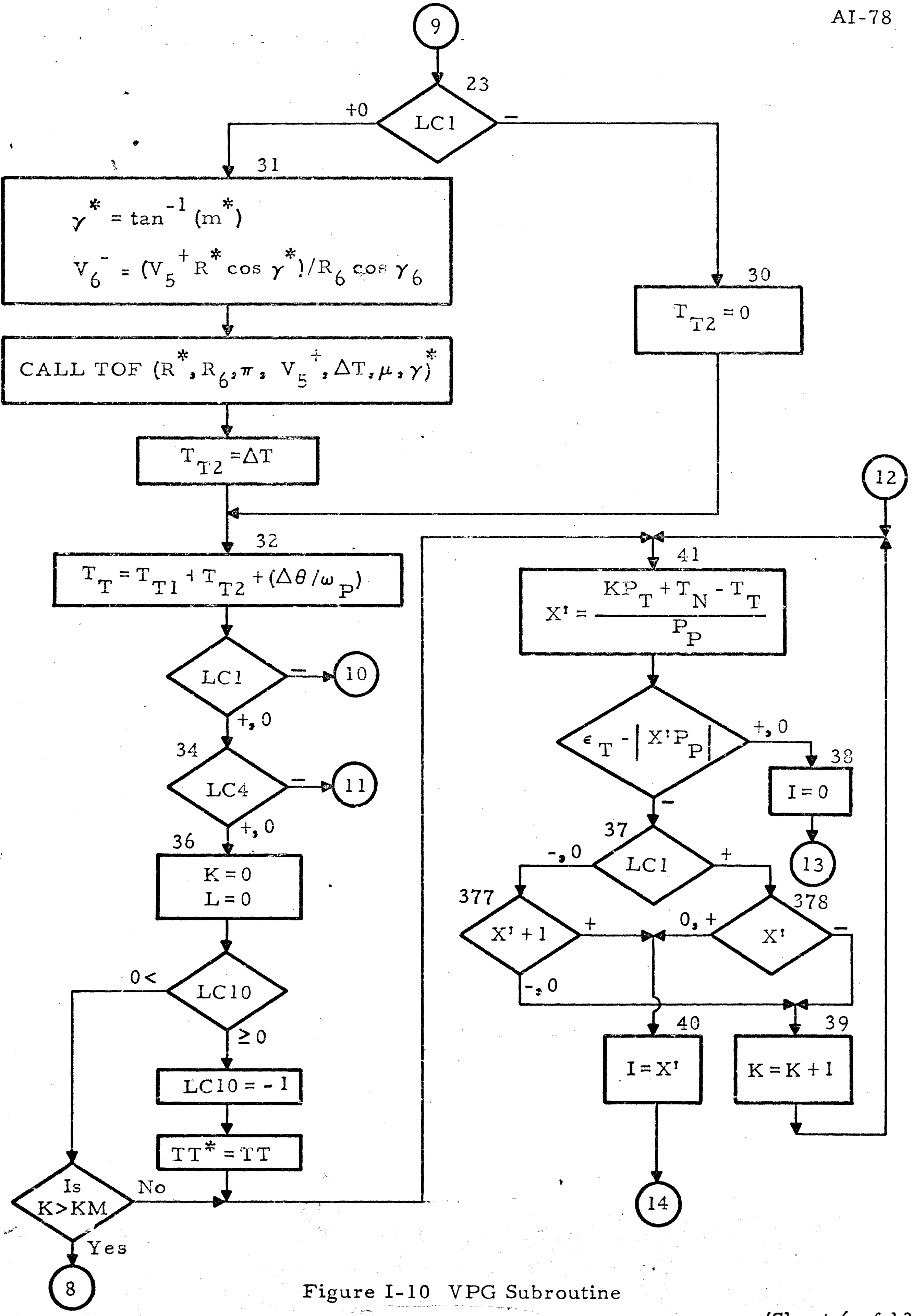


Figure I-10 VPG Subroutine

(Sheet 6 of 13)

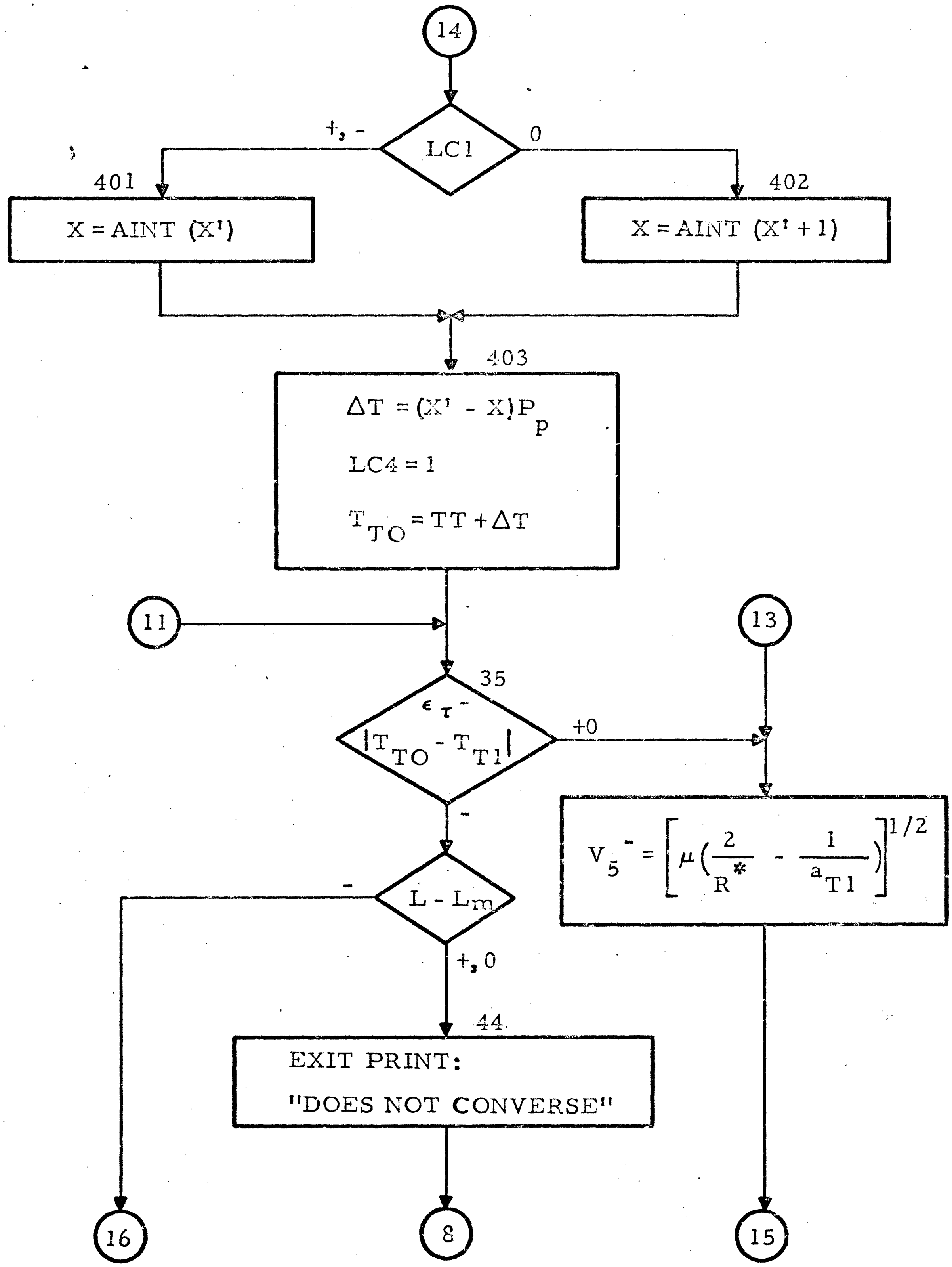


Figure I-10 VPG Subroutine

(Sheet 7 of 13)

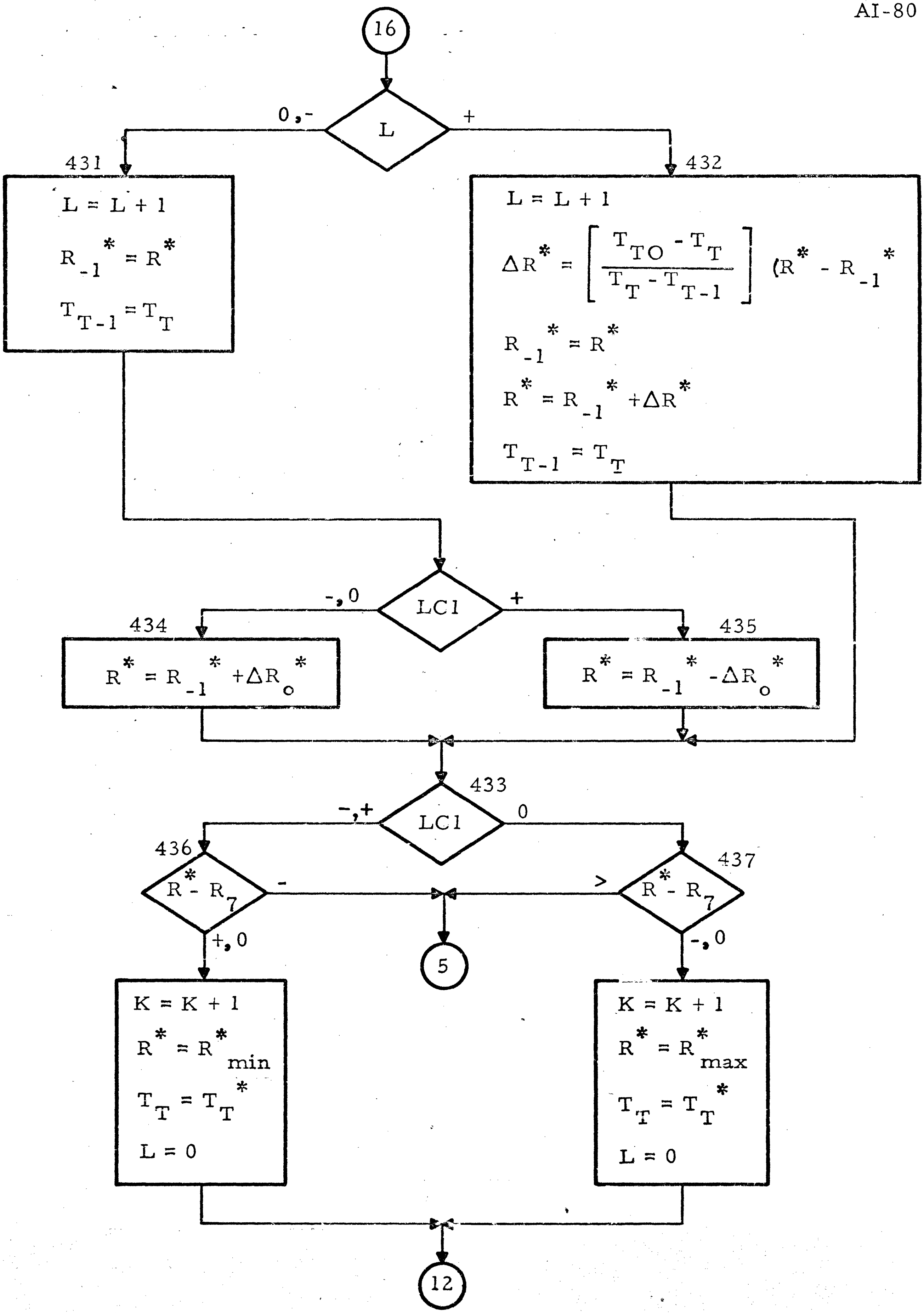


Figure I-10 VPG Subroutine

(Sheet 8 of 13)

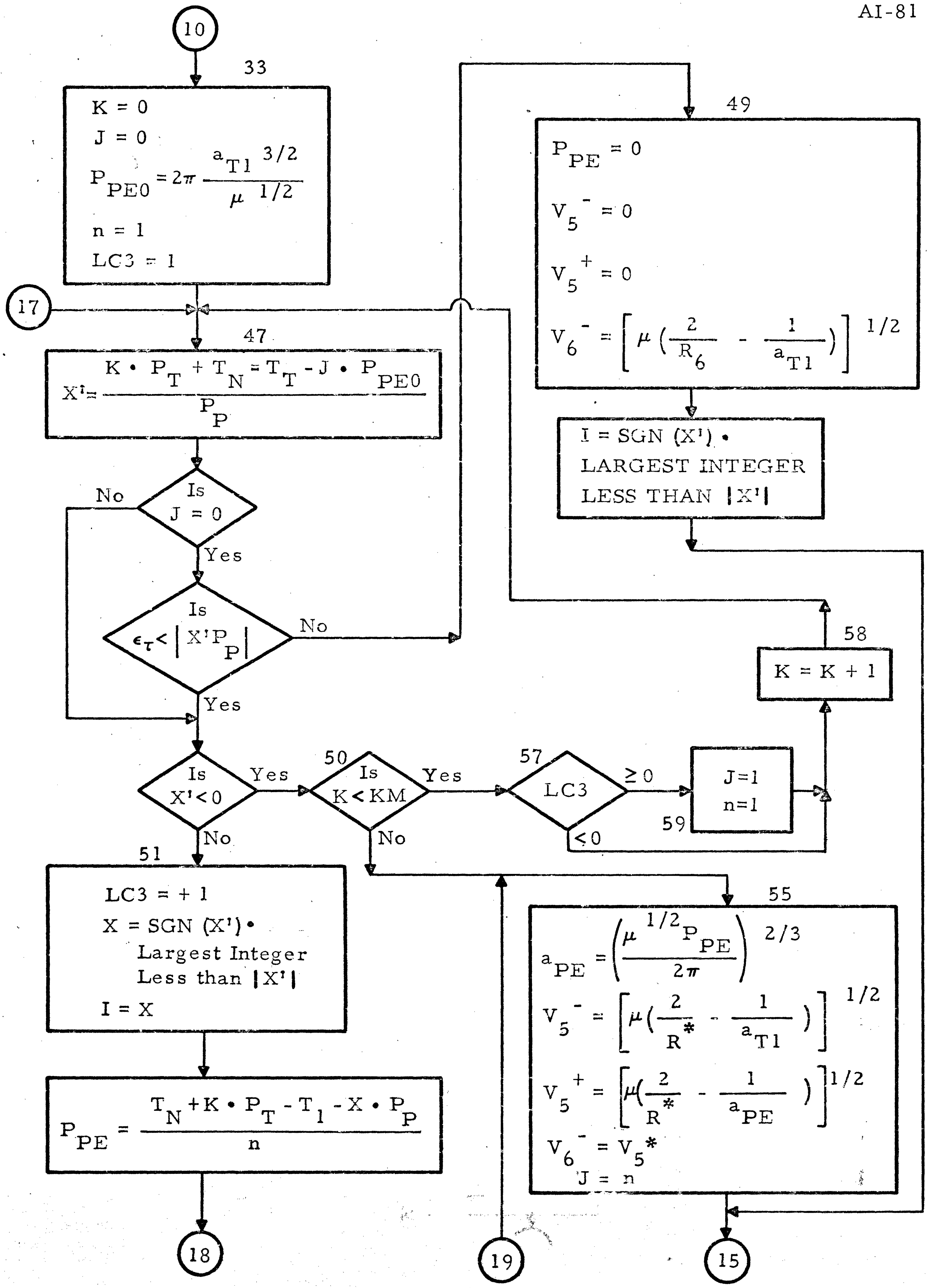

10
33
K = 0
J = 0
$P_{PE0} = 2\pi \frac{a_{T1}^{3/2}}{\mu^{1/2}}$
n = 1
LC3 = 1
17
47
$X' = \frac{K \cdot P_T + T_N = T_T - J \cdot P_{PE0}}{P_P}$
Is J = 0
No
Yes
Is $\epsilon_\tau < |X' P_P|$
No
Yes
Is $X' < 0$
Yes
No
50
Is K < KM
Yes
No
57
LC3
≥ 0
< 0
59
J=1
n=1
58
K = K + 1
49
$P_{PE} = 0$
$V_5^- = 0$
$V_5^+ = 0$
$V_6^- = \left[\mu\left(\frac{2}{R_6} - \frac{1}{a_{T1}}\right)\right]^{1/2}$
I = SGN (X') • LARGEST INTEGER LESS THAN |X'|
51
LC3 = + 1
X = SGN (X') • Largest Integer Less than |X'|
I = X
$P_{PE} = \frac{T_N + K \cdot P_T - T_1 - X \cdot P_P}{n}$
55
$a_{PE} = \left(\frac{\mu^{1/2} P_{PE}}{2\pi}\right)^{2/3}$
$V_5^- = \left[\mu\left(\frac{2}{R^*} - \frac{1}{a_{T1}}\right)\right]^{1/2}$
$V_5^+ = \left[\mu\left(\frac{2}{R^*} - \frac{1}{a_{PE}}\right)\right]^{1/2}$
$V_6^- = V_5^*$
J = n
18
19
15

Figure I-10 VPG Subroutine

(Sheet 9 of 13)

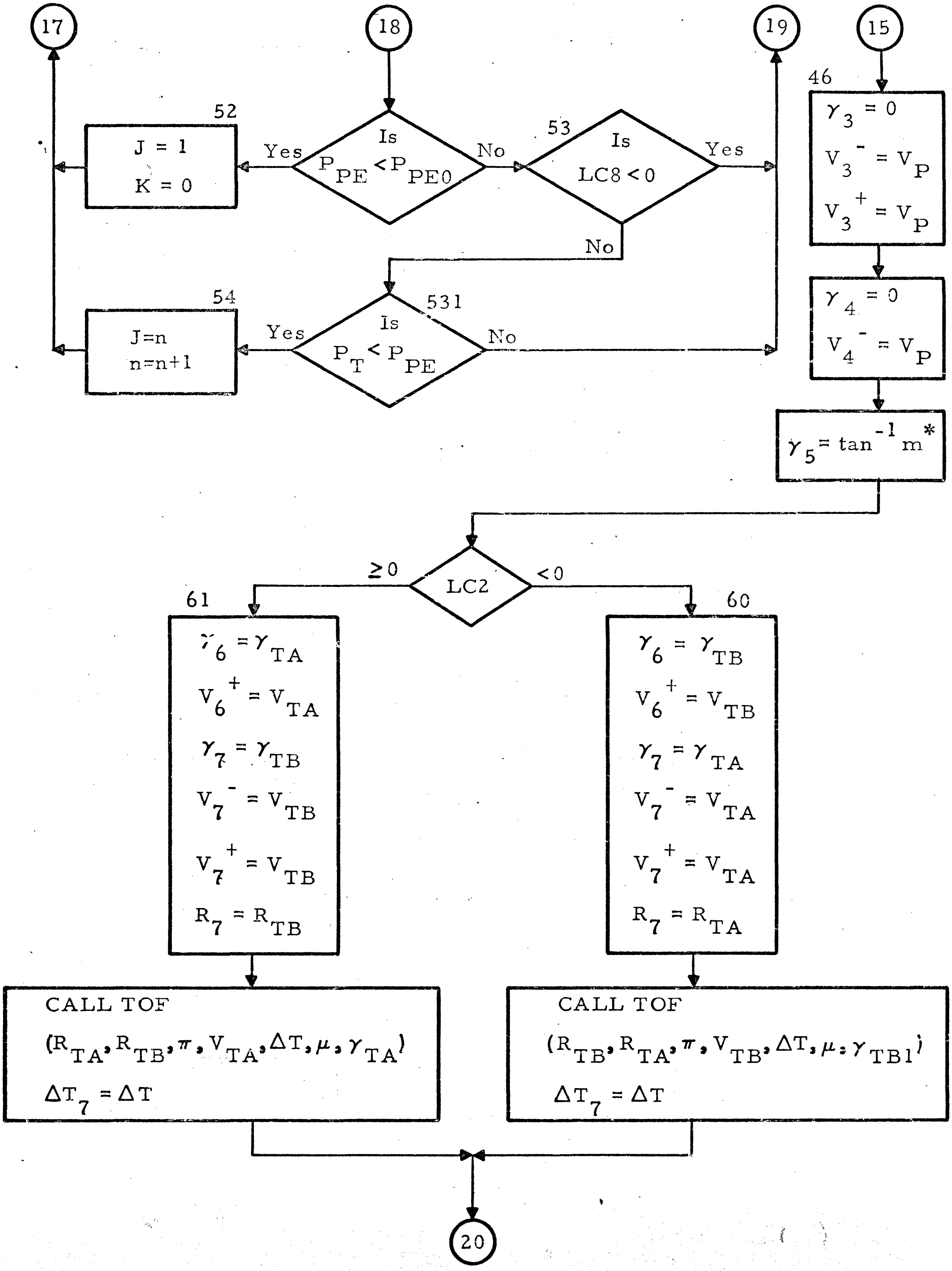


Figure I-10 VPG Subroutine

(Sheet 10 of 13)

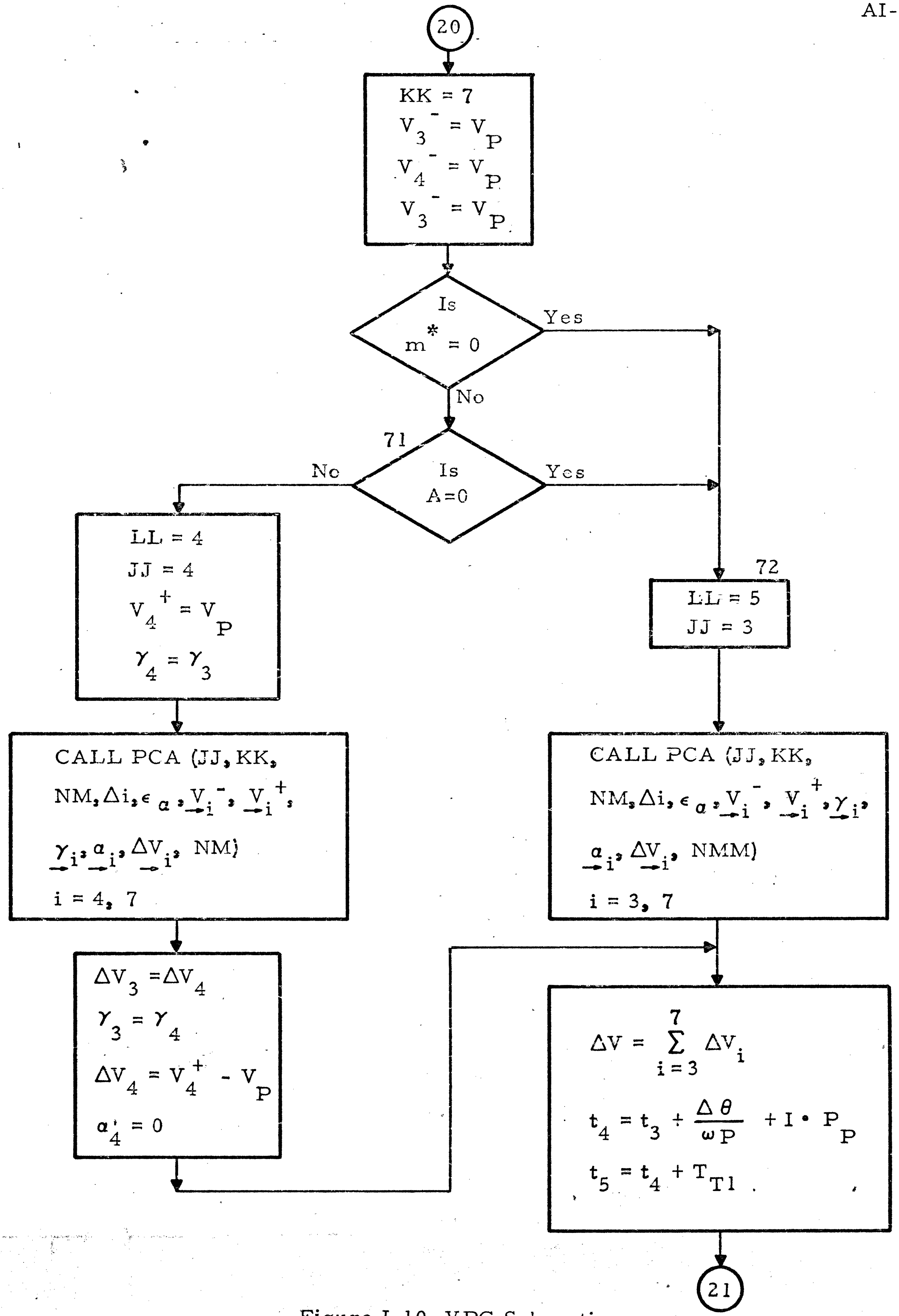

20
KK = 7
$V_3^- = V_P$
$V_4^- = V_P$
$V_3^- = V_P$
Is $m^* = 0$
Yes
No
71
Is A=0
No
Yes
LL = 4
JJ = 4
$V_4^+ = V_P$
$\gamma_4 = \gamma_3$
72
LL = 5
JJ = 3
CALL PCA (JJ, KK, NM, Δi, ϵ_a, $\underset{\rightarrow}{V_i^-}$, $\underset{\rightarrow}{V_i^+}$, $\underset{\rightarrow}{\gamma_i}$, $\underset{\rightarrow}{\alpha_i}$, $\underset{\rightarrow}{\Delta V_i}$, NM)
i = 4, 7
CALL PCA (JJ, KK, NM, Δi, ϵ_a, $\underset{\rightarrow}{V_i^-}$, $\underset{\rightarrow}{V_i^+}$, $\underset{\rightarrow}{\gamma_i}$, $\underset{\rightarrow}{\alpha_i}$, $\underset{\rightarrow}{\Delta V_i}$, NMM)
i = 3, 7
$\Delta V_3 = \Delta V_4$
$\gamma_3 = \gamma_4$
$\Delta V_4 = V_4^+ - V_P$
$\alpha'_4 = 0$
$\Delta V = \sum_{i=3}^{7} \Delta V_i$
$t_4 = t_3 + \frac{\Delta\theta}{\omega P} + I \cdot P_P$
$t_5 = t_4 + T_{T1}$
21

Figure I-10 VPG Subroutine

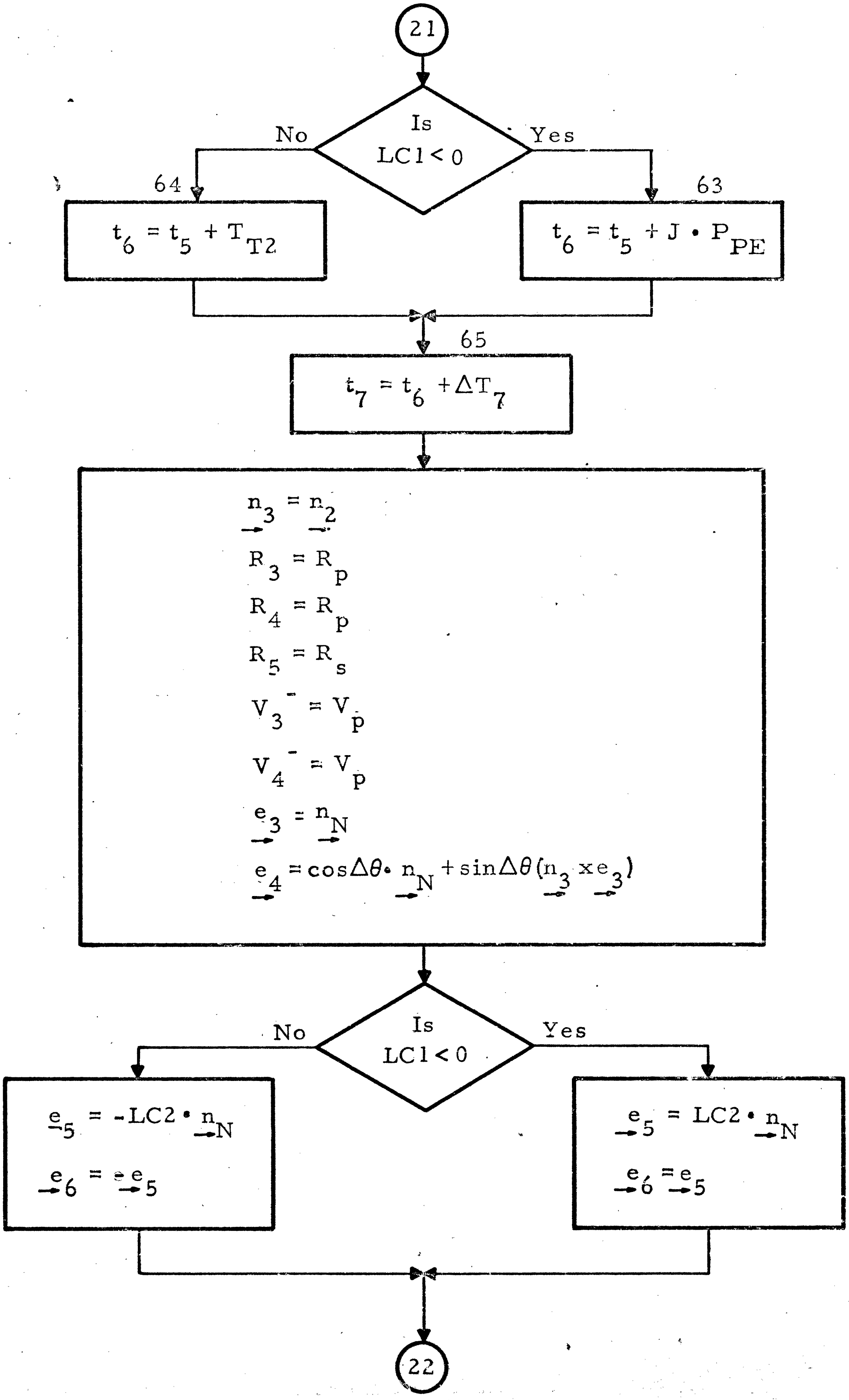

21
Is LC1 < 0
No
Yes
64
$t_6 = t_5 + T_{T2}$
63
$t_6 = t_5 + J \cdot P_{PE}$
65
$t_7 = t_6 + \Delta T_7$
$\underset{\rightarrow}{n}_3 = \underset{\rightarrow}{n}_2$
$R_3 = R_p$
$R_4 = R_p$
$R_5 = R_s$
$V_3^- = V_p$
$V_4^- = V_p$
$\underset{\rightarrow}{e}_3 = \underset{\rightarrow}{n}_N$
$\underset{\rightarrow}{e}_4 = \cos\Delta\theta \cdot \underset{\rightarrow}{n}_N + \sin\Delta\theta (\underset{\rightarrow}{n}_3 \times \underset{\rightarrow}{e}_3)$
Is LC1 < 0
No
Yes
$\underset{\rightarrow}{e}_5 = -LC2 \cdot \underset{\rightarrow}{n}_N$
$\underset{\rightarrow}{e}_6 = -\underset{\rightarrow}{e}_5$
$\underset{\rightarrow}{e}_5 = LC2 \cdot \underset{\rightarrow}{n}_N$
$\underset{\rightarrow}{e}_6 = \underset{\rightarrow}{e}_5$
22

Figure I-10 VPG Subroutine

(Sheet 12 of 13)

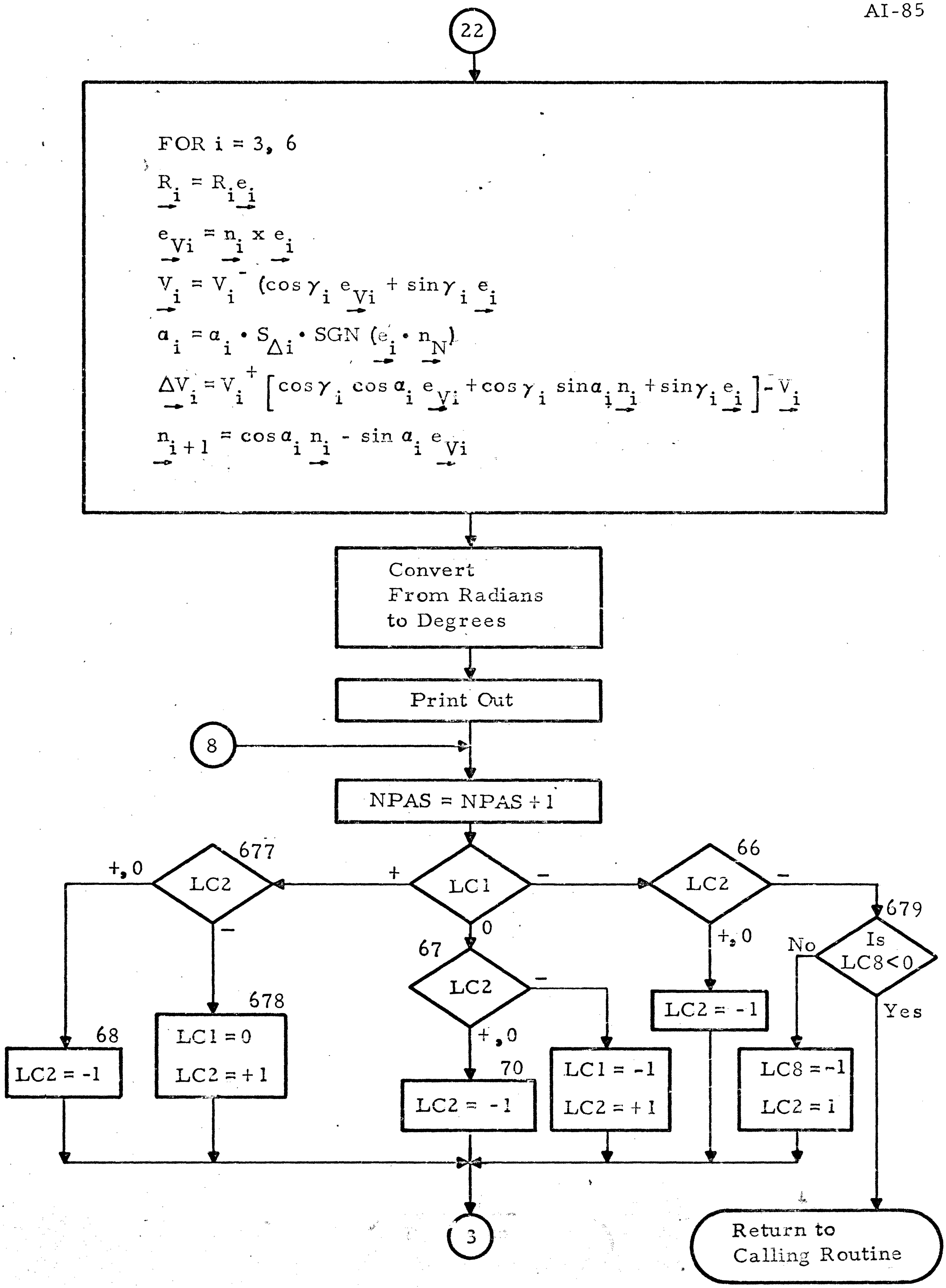

22
FOR i = 3, 6
$\vec{R}_i = R_i \vec{e}_i$
$\vec{e}_{Vi} = \vec{n}_i \times \vec{e}_i$
$\vec{V}_i = V_i^- (\cos\gamma_i \vec{e}_{Vi} + \sin\gamma_i \vec{e}_i$
$\alpha_i = \alpha_i \cdot S_{\Delta i} \cdot SGN (\vec{e}_i \cdot \vec{n}_N)$
$\Delta\vec{V}_i = V_i^+ \left[\cos\gamma_i \cos\alpha_i \vec{e}_{Vi} + \cos\gamma_i \sin\alpha_i \vec{n}_i + \sin\gamma_i \vec{e}_i\right] - \vec{V}_i$
$\vec{n}_{i+1} = \cos\alpha_i \vec{n}_i - \sin\alpha_i \vec{e}_{Vi}$
Convert From Radians to Degrees
Print Out
8
NPAS = NPAS + 1
677
+, 0
LC2
+
LC1
–
66
LC2
–
679
–
0
67
+, 0
No
Is LC8 < 0
Yes
LC2
–
678
68
LC2 = -1
LC1 = 0
LC2 = +1
+, 0
70
LC2 = -1
LC1 = -1
LC2 = +1
LC2 = -1
LC8 = -1
LC2 = 1
3
Return to Calling Routine

Figure I-10 VPG Subroutine

(Sheet 13 of 13)

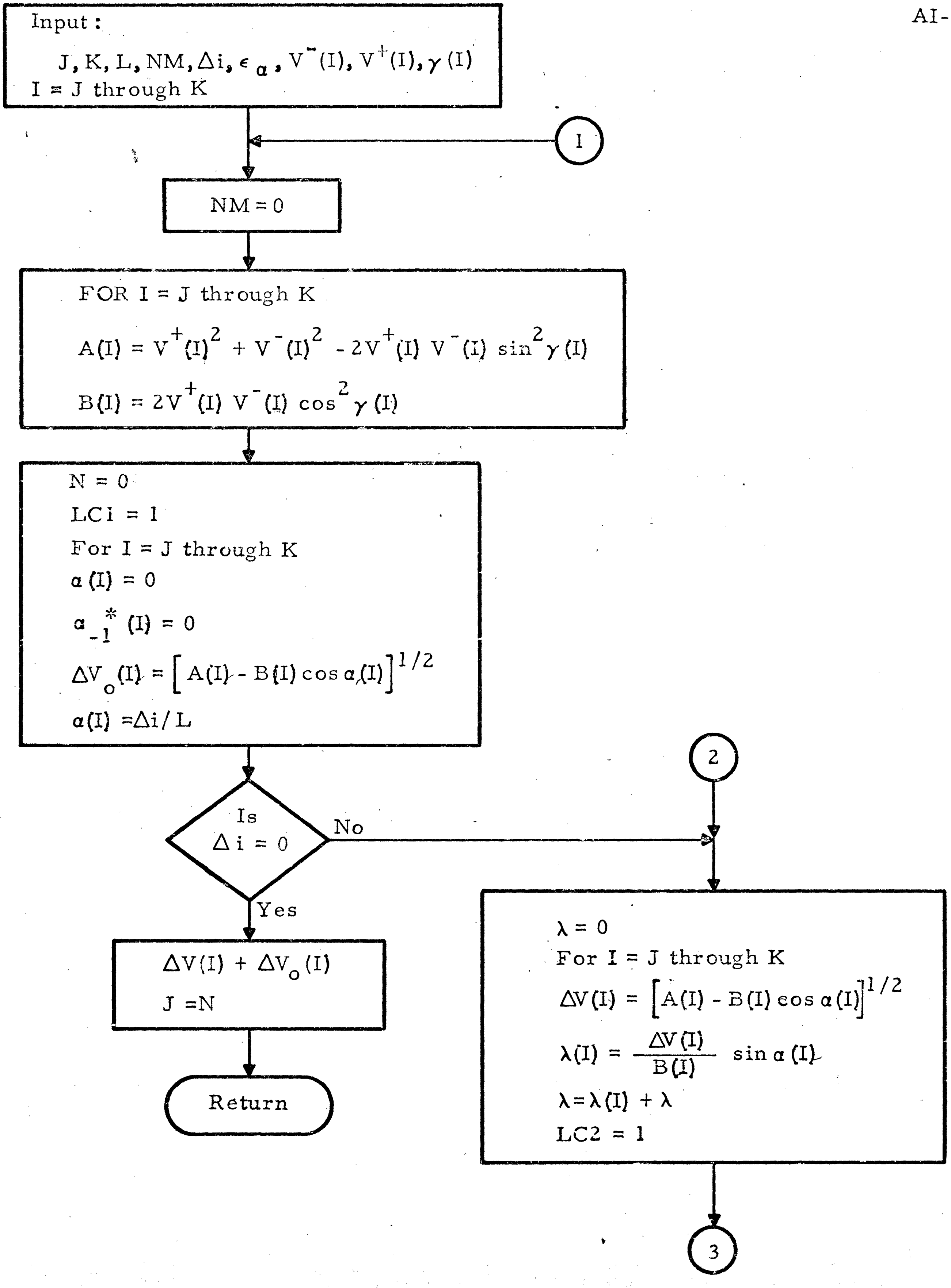

Input:
J, K, L, NM, Δi, ϵ_α, $V^-(I)$, $V^+(I)$, $\gamma(I)$
I = J through K
1
NM = 0
FOR I = J through K
$A(I) = V^+(I)^2 + V^-(I)^2 - 2V^+(I)\ V^-(I)\ \sin^2\gamma(I)$
$B(I) = 2V^+(I)\ V^-(I)\ \cos^2\gamma(I)$
N = 0
LCi = 1
For I = J through K
$\alpha(I) = 0$
$\alpha^*_{-1}(I) = 0$
$\Delta V_o(I) = \left[A(I) - B(I)\cos\alpha(I)\right]^{1/2}$
$\alpha(I) = \Delta i / L$
2
Is
$\Delta i = 0$
No
Yes
$\Delta V(I) + \Delta V_o(I)$
J = N
Return
$\lambda = 0$
For I = J through K
$\Delta V(I) = \left[A(I) - B(I)\cos\alpha(I)\right]^{1/2}$
$\lambda(I) = \frac{\Delta V(I)}{B(I)}\ \sin\alpha(I)$
$\lambda = \lambda(I) + \lambda$
LC2 = 1
3

Figure I-11 PCA Subroutine

(Sheet 1 of 3)

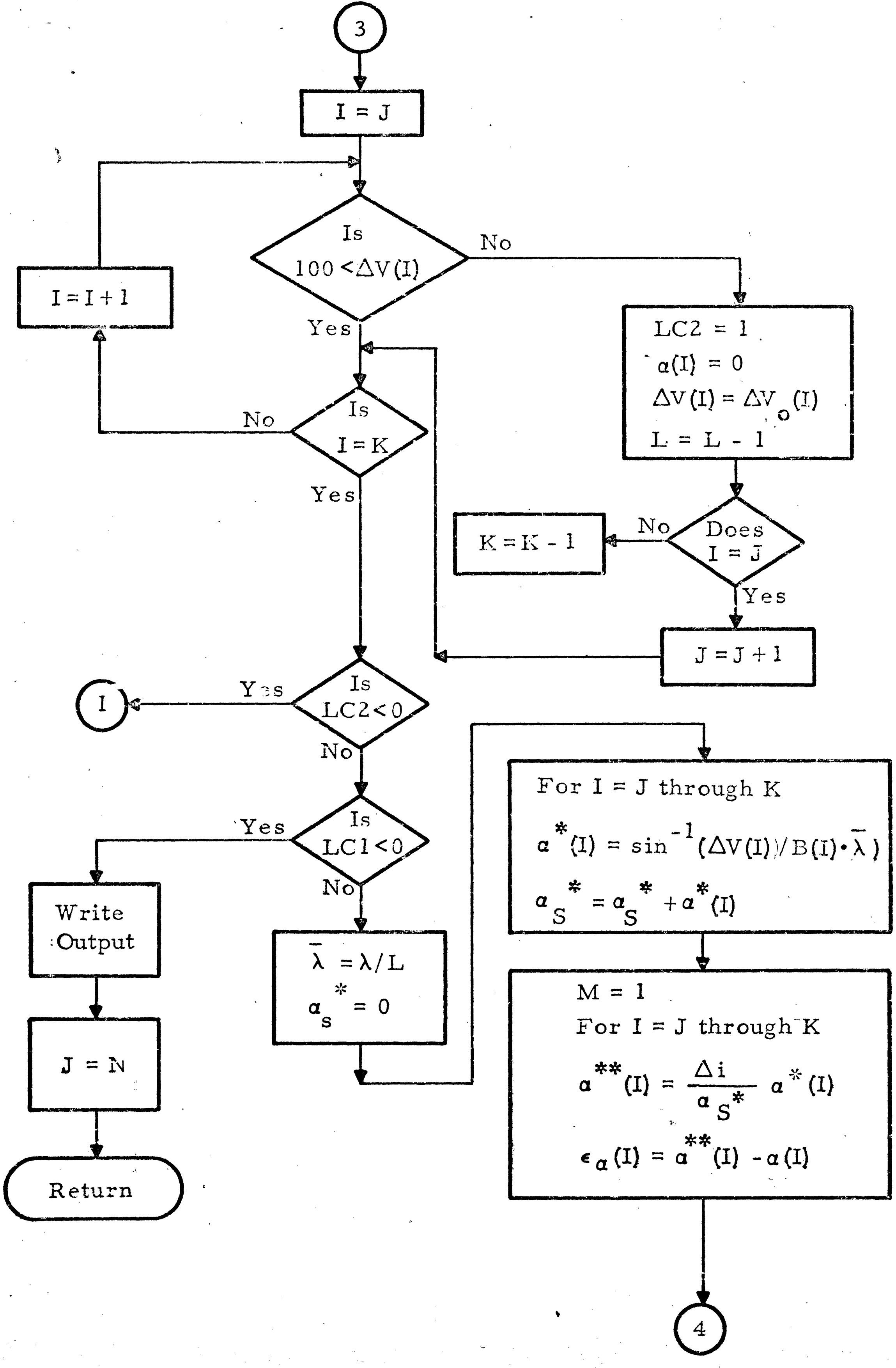

3
$I = J$
Is $100 < \Delta V(I)$
No
Yes
$I = I + 1$
$LC2 = 1$
$\alpha(I) = 0$
$\Delta V(I) = \Delta V_o(I)$
$L = L - 1$
Is $I = K$
No
Yes
Does $I = J$
No
Yes
$K = K - 1$
$J = J + 1$
Is $LC2 < 0$
Yes
No
I
Is $LC1 < 0$
Yes
No
Write Output
$J = N$
Return
$\bar{\lambda} = \lambda / L$
$\alpha_s^* = 0$
For I = J through K
$\alpha^*(I) = \sin^{-1}(\Delta V(I))/B(I) \cdot \bar{\lambda})$
$\alpha_S^* = \alpha_S^* + \alpha^*(I)$
$M = 1$
For I = J through K
$\alpha^{**}(I) = \frac{\Delta i}{\alpha_S^*} \alpha^*(I)$
$\epsilon_\alpha(I) = \alpha^{**}(I) - \alpha(I)$
4

Figure I-11 PCA Subroutine

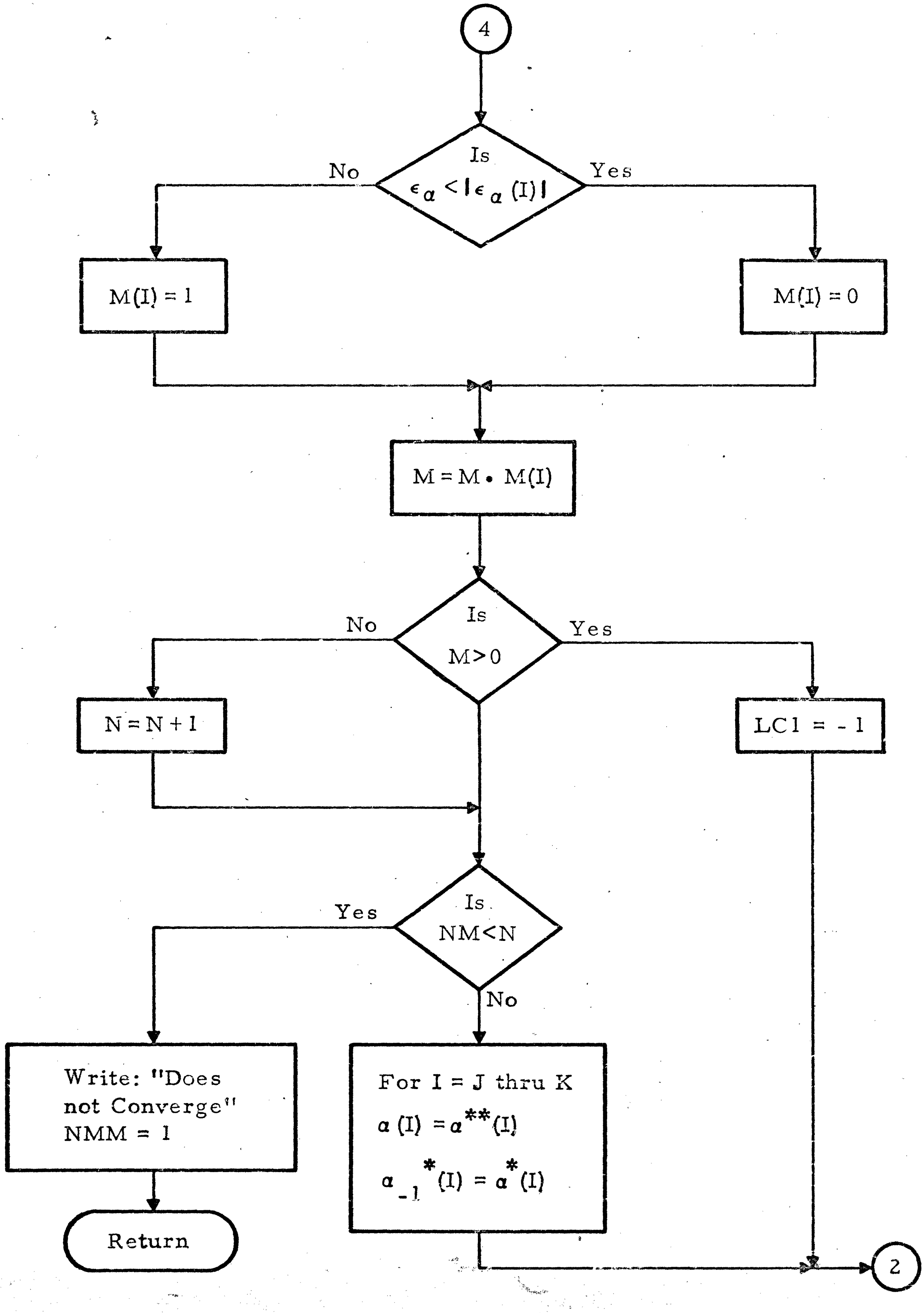
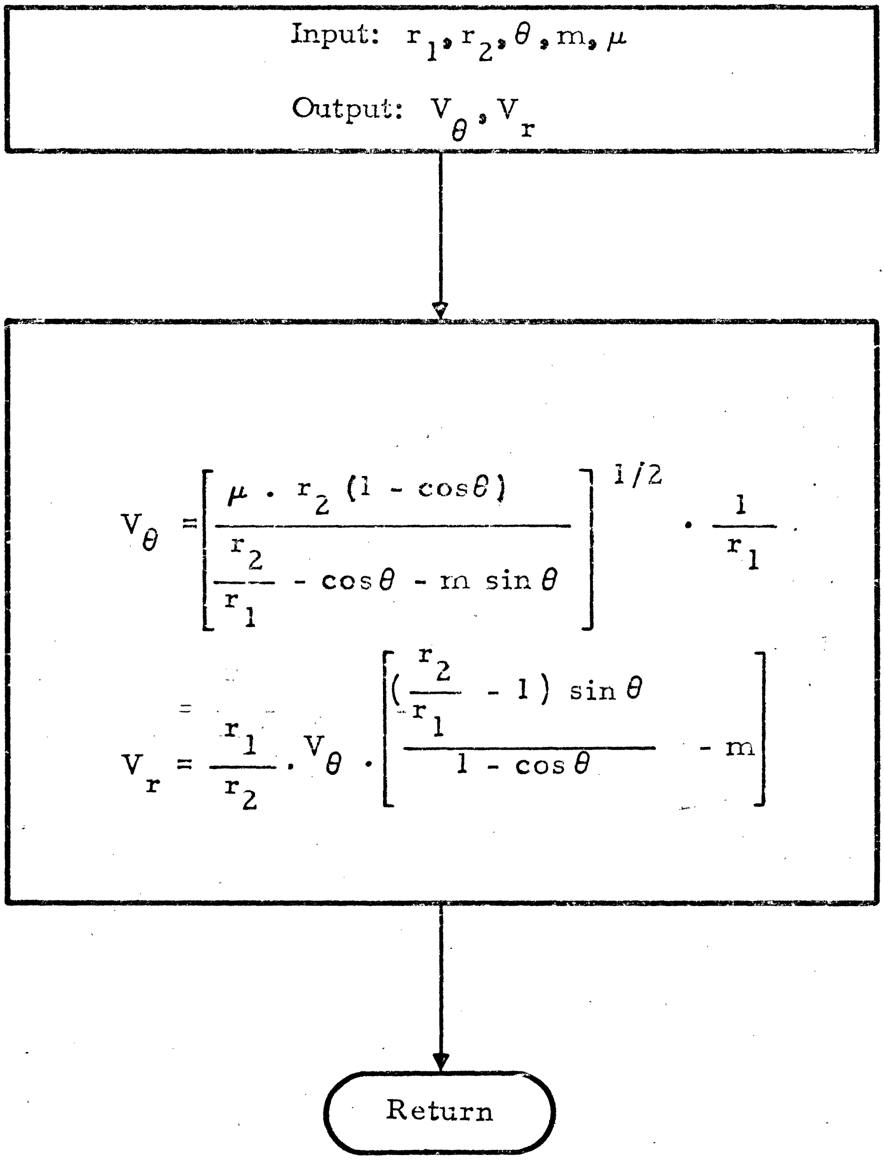

4
Is
$\epsilon_\alpha < |\epsilon_\alpha(I)|$
No
Yes
M(I) = 1
M(I) = 0
M = M • M(I)
Is
M>0
No
Yes
N = N + 1
LC1 = - 1
Is
NM<N
Yes
No
Write: "Does not Converge" NMM = 1
Return
For I = J thru K
$\alpha(I) = \alpha^{**}(I)$
$\alpha_{-1}^{*}(I) = \alpha^{*}(I)$
2

Figure I-11 PCA Subroutine

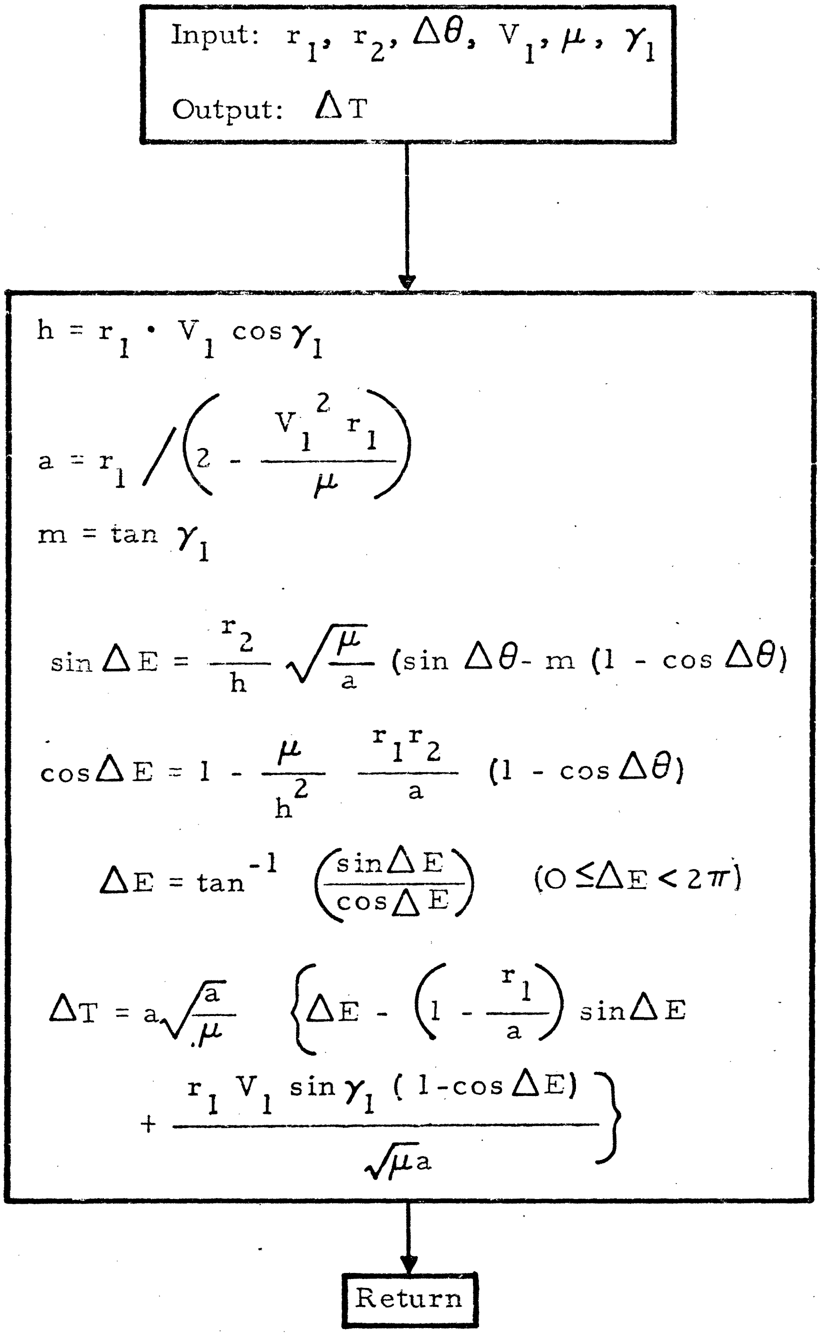
(Sheet 3 of 3)



$$V_\theta = \left[\frac{\mu \cdot r_2 (1 - \cos\theta)}{\frac{r_2}{r_1} - \cos\theta - m \sin\theta} \right]^{1/2} \cdot \frac{1}{r_1}$$

$$V_r = \frac{r_1}{r_2} \cdot V_\theta \cdot \left[\frac{\left(\frac{r_2}{r_1} - 1\right) \sin\theta}{1 - \cos\theta} - m \right]$$

Figure I-12 REQ Subroutine



Input: r_1, r_2, $\Delta\theta$, V_1, μ, γ_1

Output: ΔT

$$h = r_1 \cdot V_1 \cos\gamma_1$$

$$a = r_1 \Big/ \left(2 - \frac{V_1^2 r_1}{\mu}\right)$$

$$m = \tan\gamma_1$$

$$\sin\Delta E = \frac{r_2}{h}\sqrt{\frac{\mu}{a}}\,(\sin\Delta\theta - m\,(1 - \cos\Delta\theta)$$

$$\cos\Delta E = 1 - \frac{\mu}{h^2}\;\frac{r_1 r_2}{a}\;(1 - \cos\Delta\theta)$$

$$\Delta E = \tan^{-1}\left(\frac{\sin\Delta E}{\cos\Delta E}\right) \qquad (0 \leq \Delta E < 2\pi)$$

$$\Delta T = a\sqrt{\frac{a}{\mu}}\;\left\{\Delta E - \left(1 - \frac{r_1}{a}\right)\sin\Delta E + \frac{r_1\,V_1\,\sin\gamma_1\,(1 - \cos\Delta E)}{\sqrt{\mu}\,a}\right\}$$

Return

Figure I-13 TOF Subroutine

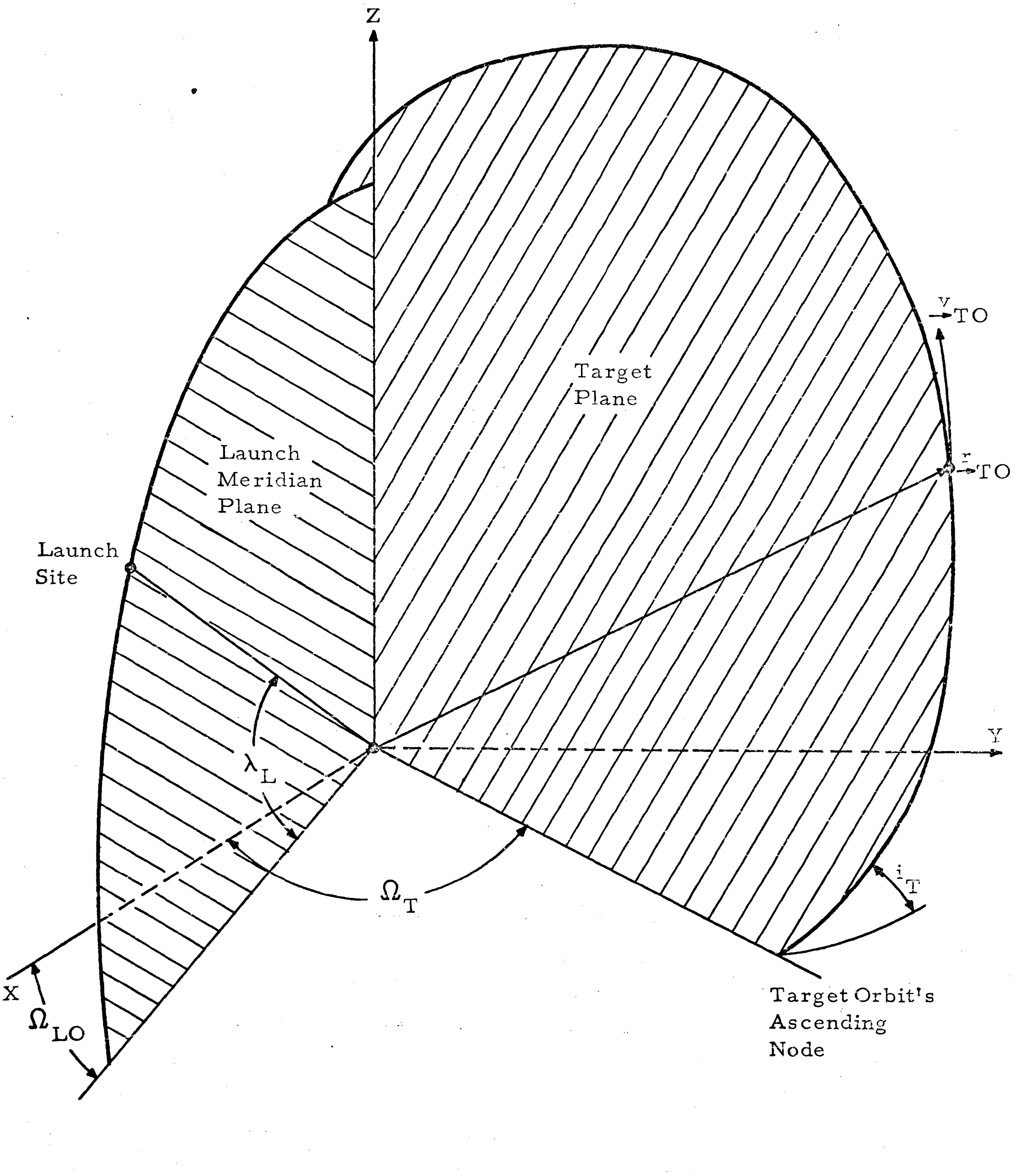


Figure I-14 Coordinate System and Initial Geometry

REFERENCE

1. Capen, E. B., Camiel, J. J., McNaughton, R. V., et. al., Variable Point Guidance and Targeting Final Report, CR-67-588-2, December 1966.

Appendix II

TOP - TRAJECTORY OPTIMIZER PROGRAM

2.1 INTRODUCTION

The Trajectory Optimizer Program (TOP) is a digital computer program written in FORTRAN IV to run on IBM 7090/7094 computers. The program is used in mission analysis and planning to generate local minimum time or minimum fuel trajectories from launch through various orbital maneuvers.

The program employs a direct search optimization algorithm which systematically varies the paramater of an initial (input) trajectory in order to minimize a pay-off function - the time or ΔV required to complete the mission. Inequality constraints on trajectory parameters are enforced in the search routine while functional constraints are handled by the addition of penalty terms in the calculation of the pay-off function.

This program was written to demonstrate feasibility of the Trajectory Optimization Mode of the On-Board Mission Planning Function (see Section V of the main report).

2.2 PURPOSE AND USE

TOP represents a flexible and efficient technique for generating local optimum time or fuel mission trajectories. The trajectories to be optimized can start at launch or from orbit and can include any number of orbital burns for orbit transfer with or without rendezvous (the present version of TOP is limited to four orbital burns). All orbital burns are treated as impulses while the ascent-to-orbit employs an explicit guidance algorithm which provides nearly optimal performance.

This program is used to optimize the trajectories generated by the Trajectory Generator Program (Appendix I), but it can be used with any initial trajectory if the proper trajectory parameters are specified (see Paragraph 2.5 for input requirements).

2.3 PROGRAM FEATURES

The present version of TOP does not represent a production level program. It does represent a possible technique for on-board generation of locally optimal mission trajectories. The program is the result of different efforts of Task II during this phase. The present program configuration is not designed for high efficiency, and certain approximation used in it must be studied in more detail before a final version is constructed.

Some of the features of the present version are:

- Ascent to orbit or intercept (e.g., direct-ascent rendezvous capability).
- Spherical gravity model.
- Kepler arcs for coast phases.
- Total ΔV or total mission time can be minimized.
- Maximum ΔV constraints enforced by a penalty term in the pay-off function (OBJECT) when minimizing total time.
- Components of the argument vector ($\underline{U}$) can be constrained in the search routine (TEXT).
- Up to four orbital burns can be considered.
- Transfers can be with or without rendezvous.
- Near-optimal vacuum ascent trajectories generated by an explicit guidance algorithm.
- Atmospheric ascent trajectories calculated with analytical equations and vehicle-dependent empirical parameters.
- All orbital burns treated as impulses.
- Initial parking orbit, after the ascent phase, is circular.
- Impulse splitting cases can be handled.
- 180 degree transfers are allowed.
- Optimum transfer plane determined for 180 degree transfers.

Some of the approximations implicit in these features are discussed in Paragraph 2.4.3 of this Appendix.

2.4 PROGRAM OPERATION

The over-all operation of TOP is illustrated in Figure II-1. The program consists of a Main Program (MAIN) and two major Subroutines, TEXT and OBJECT. The Main Program reads in all data, performs necessary conversions, sets up program paths, calls OBJECT and TEXT, and formats all printout. TEXT is the optimization subroutine which itself consists of a number of subroutines that can be used in various combinations. OBJECT is the subroutine which calculates the value of the pay-off function (P) corresponding to the present value of the trajectory parameters ($\underline{U}$) as determined in TEXT.

Referring to Figure II-1, MAIN reads in all data including the initial trajectory, which is specified by a set of parameters ($\underline{U}$ (I), I = 1, . . . 15). The initial trajectory specification, $\underline{U}_o$, is used with OBJECT to provide the initial value of the pay-off function, P_o. TEXT is then called to begin the search procedure, and control remains in TEXT until a solution has been found. At this time, MAIN formats and prints the output data and either proceeds to the next case or stops. In TEXT, the OBJECT subroutine is called to provide a new value of P whenever $\underline{U}$ has been changed.

2.4.1 MAIN

A math flow of the executive program MAIN is presented in Figure II-2. The symbols employed may be found in Table II-1. In addition to reading input, printing output, and calling OBJECT and TEXT, this program:

- Sets constraints on appropriate terms of the argument vector ($\underline{U}$) with the read-in values of U_{max} (I) and U_{min}(I).
- Identifies the components of $\underline{U}$ to be searched over (BVA(I) = 1 if U (I) is to be varied, BVA (J) = 0 if U (J) is to be held fixed).

- If an impulse-splitting maneuver is being considered, MAIN reverses the order of the orbital burns before calling TEXT and changes them back after returning from TEXT.

The last item relates to the problem of determining the velocity required to transfer between two points in space in a finite time when the points are coincident. This is the case for the so-called impulse splitting maneuver which is employed in rendezvous missions to introduce a phasing orbit. When OBJECT is first entered to determine the initial value of the pay-off function a check is made to see if the last two burns are within ρ (RHO)-feet (an input) of each other. If they are, an indicator is set which, on the return to MAIN,. causes the order of the orbital burns to be reversed (i. e., the program starts at the target orbit and works back to the parking orbit). This is shown on Figure II-2 just before TEXT is called. The optimized orbital burns are then changed back to the proper order before final printout. By reversing the order, the first two orbital burns are now coincident in space but, because of the technique employed in OBJECT to propagate trajectories, this causes no problems.

2.4.2 TEXT

TEXT is the optimization algorithm subroutine. It consists of several options which can be used in various combinations. This subroutine is completely described in Appendix IVwhich includes Math flows, Input/Output, and examples. In Appendix IVthis algorithm is entitled DSOP (Direct Search Optimization Program) and its various options are described. Only the Pattern Move Search (PMS) and UNIVAR options have been employed in TOP because of the ability of this combination to handle argument constraints. Any search type algorithm could be employed with TOP but some reprogramming would be required to interface it with MAIN and OBJECT.

2.4.3 OBJECT

The payoff function associated with a particular trajectory is computed in the OBJECT subroutine. The input to this routine is the argument vector $\underline{U}$ which describes the trajectory and the output is the corresponding value of the pay-off function P. The argument vector is supplied by TEXT while OBJECT calculates the time or ΔV (the presently available pay-off options) required on that trajectory.

OBJECT makes use of five subroutines: LAUNCH, UP1, UP2, EPHEM, and RVVR. The launch conditions are calculated in LAUNCH as a function of the launch site and launch time. UP1 and UP2, employing the explicit guidance algorithm described in Par. 6.4 of the main report, generate near optimal ascent trajectories to orbit injection or intercept. The determination of position, velocity and time on coast arcs performed in EPHEM. The RVVR routine is used to calculate the velocity required for the orbital burns. These subroutines are also used in the Trajectory Generator Program (TGP) and a complete description of them and TGP may be found in Appendix I.

Figure II-3 is a math flow of CBJECT and the symbols are defined in Table II-1.

The OBJECT subroutine employs certain approximations in order to provide an efficient technique for generating ascent and orbital segments (this routine is called hundreds of times in the process of optimizing a mission trajectory). These major approximations are related to three areas of trajectory generation: Ascent, Coast, and Orbital Burns.

- Ascent - the ascent segments are generated with UP1 or UP2 depending upon whether orbit injection or intercept is required. Both of these routines (which are really two entry points to one program) use the explicit guidance equations, developed during this study, for the vacuum phase and analytical equations for the atmospheric phase. The form of the explicit equations are such that they introduce almost no performance degradation when compared to an optimal solution. The analytical equations, which are necessary to initialize the explicit guidance equations, are merely spherical trigonometry equations relating conditions at first (zero) stage burnout to the launch conditions. The equations require three vehicle-dependent parameters which are used for all launch conditions. This analytical technique of handling the atmospheric phase of ascent is very efficient computationally but it does introduce prediction errors of the order of 1 - 3 seconds in the injection time. This aspect of the ascent phase can and should be improved with additional study of efficient techniques for generating conditions at the "top of the atmsophere" as a function of launch conditions and vehicle configuration.

- Coast - All coast arcs are computed with an ephemeris routine (EPHEM) which employs Kepler-type equations, that is, assumes a spherical earth. In the case of long mission times, this will result in sizable errors if uncorrected. The present program could be upgraded by considering the effects of oblateness on coast trajectories. Closed-form expressions for these effects are available.
- Orbital Burns - All orbital burns after ascent are treated as impulses. This approximation is justified because of the almost negligible performance penalty incurred. The results of Reference (1), for example, indicate this penalty is less than 1% for an extreme case (ΔV = 10,000 fps.) and provide a prescription for generating a nearly optimal finite-thrust trajectory for data derived from the impulsive trajectory.

One other feature of OBJECT should be mentioned. This involves the program's ability to generate 180-degree transfers when the transfer plane is undefined. This would be the case, for example, for a simple Hohnann transfer between non-coplanar circular orbits. Specifying the times (positions) of the two burns is not sufficient because the transfer plane is still arbitrary (i.e., the plane change can all be made at the first burn, or all at the second burn, or split between the two burns in an infinite number of ways). The logical solution, of course, is to divide the plane-change between the two burns so as to minimize the total ΔV required for the two burns. This is exactly what is done in OBJECT when the trajectory specification, $\underline{U}$ vector, calls for two burns that are within XK1-degrees (a program constant, presently set at approximately 0.08°) of being 180 degrees apart. When this situation arises, a simple one-dimensional search is performed over the angle describing the transfer plane, to find the value which minimizes the ΔV required for these two burns. This capability is required in order to accept such transfers as part of an initial trajectory specification. The VPG Bielliptic option, for example, always includes such a transfer.

2.5 PROBLEM SPECIFICATION

The coordinate system used in defining input and output mission data, and in which the problem is solved, is the XYZ ECI (Earth Centered Inertial) system illustrated in Figure II-4. The XY plane is the equatorial plane and the Z axis is through the north pole.

If the initial trajectory starts at launch (LCL = +1) the launch site is specified by four parameters which are varied during the optimization.

t_L = Time of launch = U(12)

A_L = Launch Azimuth = U(13)

R_p = Circular parking orbit radius = U(14)

α_L = Dog-leg angle = U(15)

The present program only handles circular parking orbits. The dog-leg angle α_L denotes a rotation about the launch radius which defines the desired ascent plane after the atmospheric phase. A_L defines the plane for the atmospheric phase.

The target orbit is specified by the target's ephemeris at some epoch:

$\underline{R}_{T_o}$ = Target's position at t_{T_o}

$\underline{V}_{T_o}$ = Target's velocity at t_{T_o}

t_{T_o} = Target's epoch

If the initial trajectory starts from a parking orbit, LCL = -1, this orbit is defined by appropriate ephemeris data:

$\underline{R}_o$ = Spacecraft position at t_o

$\underline{V}_o$ = Spacecraft velocity at t_o

t_o = Spacecraft's epoch

The specification of the orbital burns requires 3 + 4 (N-2) parameters, where N is number of burns ($N \geq 2$). If rendezvous is required, one less parameter is involved because of the phasing constraint (i.e., the spacecraft and target must be coincident at the time of the last burn). For all values of N, the first and last burns are defined by:

t_1 = Time of first orbital burn = U(1)
t_f = Time of last orbital burn = U(2)
ma_f = Slope oi arrival at last burn = U(3)

The position of the first burn is determined by t_1 and the initial parking orbit (input or result of the ascent phase). The position of the last burn is determined by t_f and the target's ephemeris. For a two burn orbit transfer, this and the slope (tangent of the flight path angle) at arrival at the last burn are sufficient to determine the two impulsive burns required. In the case of rendezvous, ma_f is not used; the phasing constraint is employed instead.

When there are more than two orbital burns, four more parameters are required for each additional burn. The parameters used are:

ΔV_i = Velocity impulse at i^{th} burn = U(4i), U(4i+1), U(4i+2)
$\Delta\theta_i$ = Range angle between i^{th} and $(i+1)^{st}$ burns = U(4i+3)

where i = 1, 2, ··· (N-2). This combination allows for efficient generation of the intermediate trajectories, because only the ephemeris routine (EPHEM) is required. For example, t_1 and the spacecraft's ephemeris with EPHEM yield the position and velocity just before the first impulse ($\underline{R}_1$ and $\underline{V}_1$). Adding $\Delta \underline{V}_1$ to $\underline{V}_1$ gives the velocity after the impulse which, with $\underline{R}_1$ and $\Delta\theta_1$, is sufficient for EPHEM to generate the position and velocity just before the second burn ($\underline{R}_2$ and $\underline{V}_2$). Adding $\Delta\underline{V}_2$ to $\underline{V}_2$, proceeding as before enables the next arc to be computed, and so on.

The initial trajectory (input) is thus specified by assigning the proper values to the initial value of the argument vector $\underline{U}(i)$, i = 1, 2, ··· 15. In addition, other input data is required to completely specify the problem. The total input required is indicated in Table II-1.

2.6 TEST CASES

To demonstrate the capability and performance of TOP, four test cases are included. Table II-2 contains a description of each case, and Table II-3 illustrates the results obtained from TOP.

Case 1 - This is a two burn orbit transfer without rendezvous between two similar orbits inclined at 5 degrees. The initial (input) trajectory corresponds to a perigee-to-perigee transfer arriving at the final orbit with zero flight path angle. This maneuver requires 7619 ft/sec of ΔV. The optimized trajectory requires 5663 ft/sec, a saving of 1856 ft/sec. This case required 155 evaluations of the pay-off function (i.e., passes through OBJECT) and 2.78 seconds of 7094 time. The running time figure contains I/O operations for writing input data, intermediate results, and final results. This case corresponds to Optimum 4, Table 1, page 1868 of Reference (2) which lists a value of 5800 for ΔV. This difference in results is due to the fact that the values for ΔV in Reference (2) are obtained by reading contour plots which have a contour interval of 500 ft/sec.

Case 2 - The problem here is to rendezvous with a target in a highly elliptical orbit with a large semi-major axis. The initial orbit is a 100 n.m. circular parking orbit inclined at 30 degrees to the target orbit. The starting solution is a three-burn bielliptic transfer which was generated with the Variable Point Guidance (VPG) routine. This routine is used in the Trajectory Generator Program and is described in Appendix I. The VPG planned trajectory requires 13,528 ft/sec of ΔV as compared to the optimized results of 12,715 ft/sec. The very large number of function evaluations (1090) are a result of slow convergence in the algorithm and, especially, because during many of the evaluations 180 degree transfers are involved and the optimum transfer plane must be determined (see Par. 2. 4. 3).

Case 3 - This is exactly the same as Case 2 except that the starting solution is a three-burn, full orbit phasing result from the VPG routine. In this case the input contains an impulse splitting maneuver because the second and thrid burns occur at the same point in space. The technique described in 2. 4. 1 was employed to handle this case. The optimized trajectory requires 11, 496 ft/sec as compared to 11, 764 for the initial trajectory. This case, in particular, demonstrates the near-optimal performance of VPG in certain applications

Case 4 - The target orbit in this case is the same as Cases 2 and 3 but now the missions start at launch. The input trajectory was generated with the Trajectory Generator Program (Appendix I) and involves a launch at t = 0, at an azimuth of 90 degrees into a 100 n. m. circular parking orbit with no dog-legging (U(15)=0). This ascent trajectory results in the minimum out-of-plane angle with the target orbit that can be obtained from the given launch site (latitude of 30 degrees). The input specification then involves a two-burn rendezvous maneuver with the target which results in a total ΔV of 31, 673 ft/sec and requires 15, 706 seconds of time. The optimized trajectory required 255 evaluations of the pay-off function (total ΔV) and resulted in a mission requiring 14, 769 seconds and 31, 417 ft/sec of ΔV. This is a 256 ft/sec savings in ΔV and 937 seconds in time. The optimized mission involves a launch at t = 0 (launch time was constrained to be positive) at an azimuth of 92. 7 degrees with a 1. 287 degree dog-leg after the atmospheric phase. The parking orbit altitude was reduced to its constrained minimum value of approximately 90 n. m. The two orbital burns were altered in time of occurrence, and thus in position because the parking orbit is different.

TABLE II-1

TRAJECTORY OPTIMIZATION PROGRAM SYMBOLS

Math * Symbol	Description	FORTRAN Program Symbol
	I. Inputs to Main Program	LCL
LCL	Equals +1 for Ascent, Equals -1 for maneuvers that start from Orbit	
	Note: The following inputs are required only for LCL = +1, i.e. for ascent.	
λ_L	Launch site latitude (degrees)	LAML
Ω_{LO}	Launch site longitude relative to x-axis at t_{LO} (degrees)	OLO
t_{LO}	Reference time	TLO
Ω_T	Longitude of ascending node of target orbit (degrees)	OT
$\Delta\Omega_T$	Longitude of target overflight point relative to launch site (degrees)	DELOT
λ_T	Latitude of target overflight point (deg.)	LAMT
i_T	Inclination of target orbit plane to x - y plane $(i \geq 0)$	EYET
R'_T	Radius of target overflight point (ft.)	RTP
ΔV_{max}	Maximum ΔV available (ft/sec)	DVM
t_L	Time of launch	U(12)
A_L	Launch azimuth (degrees)	U(13)

Table II-1 Trajectory Optimization Program Symbols (Continued)

Math * Symbol	Description	FORTRAN Program Symbol
R_P	Parking orbit radius (ft.)	U(14)
α_L	Dog-leg angle (degrees)	U(15)
M_T	Equals 1 for Intercept 2 for Direct Ascent Rendezvous 3 for Orbital Rendezvous 4 for Reconnaissance with parking orbit 5 for Reconnaissance with first pass overflight 6 for Orbit injection using a parking orbit	MT
t_{Lmin}	Lower bound on t_L	AC(1)
A_{Lmin}	Lower bound on A_L (deg.)	AC(2)
A_{Lmax}	Upper bound on A_L (deg.)	AC(3)
R_{Pmin}	Lower bound on R_P (ft.)	AC(4)
R_{Pmax}	Upper bound on R_P (ft.)	AC(5)
	Switch to control the use of above bounds.	NAC(1-5)
	Note: The remaining inputs pertain to orbit maneuvers.	
N BURNS	Number of orbital burns (0, 1, 2, 3, or 4)	NBURNS
	Desired orbital maneuver. Equals +1 for rendezvous and -1 for orbit transfer.	MANEUV
	Desired minimum Equals +1 for minimum fuel and -1 for minimum time.	MINREQ
$\underset{\rightarrow}{r}_o$	Spacecraft position at time t_o	RO

Table II-1 Trajectory Optimization Program Symbols (Continued)

Math * Symbol	Description	FORTRAN Program Symbol
$\underset{\rightarrow}{v}_o$	Spacecraft velocity at time	VO
t_o	Reference time for $\underset{\rightarrow}{r}_o$ and $\underset{\rightarrow}{v}_o$	TO
$\underset{\rightarrow}{r}_{To}$	Target position at time t_{To}	RTO
$\underset{\rightarrow}{v}_{To}$	Target velocity at time t_{To}	VTO
t_{To}	Reference time for $\underset{\rightarrow}{r}_{To}$ and $\underset{\rightarrow}{v}_{To}$	TTO
t_1	Time of first orbital burn	U(1)
t_f	Time of last orbital burn	U(2)
m_{af}	Slope at last orbital burn	U(3)
$\Delta \underset{\rightarrow}{V}_1$	Impulse at first orbital burn	U(4), U(5), U(6)
$\Delta\theta_{1-2}$	Range angle between first and second orbital burns	U(7)
$\Delta \underset{\rightarrow}{V}_2$	Impulse at second orbital burn	U(8), U(9), U(10)
$\Delta\theta_{2-3}$	Range angle between second and third orbital burns	U(11)
$t_{f\ max}$	Upper bound on t_f	AC(6)
	Switch to control use of AC(6)	NAC(6)
ΔV max	Maximum available ΔV	FC(1)
	Weighting factor to control use of FC(1)	NFC(1)
μ	Gravitation constant	U, GC
ϵ_T	Rendezvous iteration time tolerance (1 sec.)	EPT
ρ	Impulse splitting indicator (5000 ft.)	RHO
	Optimization search tolerance (=1)	BIAS
	End of inputs	

Table II-1 Trajectory Optimization Program Symbols (Continued)

Math * Symbol	Description	FORTRAN Program Symbol
	II. Internal and Output Symbols to Control and Object Programs	
ω_E	Rotation rate of earth = $7.292115 \cdot 10^{-5}$ radians/sec	WE
R_E	Earth equatorial radius	RE
θ_1	Range angle of S/C from epoch to first burn position	TH1
θ_f	Range angle of target from epoch to final burn position	THF
ΔV	Total velocity change	DV
NPASS	Number of passes through Object Subroutine	NPASS
$\underset{\rightarrow}{R}_i \; i = 1, 4$	Position vector of i^{th} orbital burn (ft.)	R(, i)
$\underset{\rightarrow}{V}_i \; i = 1, 4$	Velocity vector of i^{th} orbital burn (ft/sec)	V(, i)
$T_i \; i = 1, 4$	Time of i^{th} orbital burn (sec)	T (i)

$\underset{\rightarrow}{R}_5, \underset{\rightarrow}{V}_5, T_5$	Position, velocity, and time at cut-off for ascent	R(,5), V(,5)
		T(5)

$\lvert \underset{\rightarrow}{\Delta V}_i \rvert \; i = 1, 4$	Magnitude of velocity change due to i^{th} orbital burn (ft/sec)	DELV (, i)
$\lvert \underset{\rightarrow}{\Delta V}_5 \rvert$	Magnitude of velocity change needed during ascent (ft/sec)	DELV (,5)
Δi	Out of plane angle between S/C orbit (or launch plane) and target orbit (radians)	DELI

Table II-1 Trajectory Optimization Program Symbols (Continued)

Math * Symbol	Description	FORTRAN Program Symbol
$\Delta\theta$	Range angle from launch site to intersection with target orbit (radians)	DELTH
$\underrightarrow{r}_L$	Unit vector through launch site	RL
$\underrightarrow{v}_L$	Unit vector in launch plane and perpendicular to $\underrightarrow{r}_L$	VL
$\underrightarrow{n}_L$	Unit vector normal to launch plane such that $\underrightarrow{n}_L = \underrightarrow{r}_L \times \underrightarrow{v}_L$	NL
$\underrightarrow{e}_1$	Unit vector normal to orbital plane	UNIT 1, NI
θ_p	Longitude of perigee from descending equatorial node (radians)	THETP
$\underrightarrow{r}_{LB}$	Position vector of launch site	RLB
r_a	Orbit apogee radius	RA
$\underrightarrow{r}_c$	Cut-off position vector (ft)	RC
$\underrightarrow{v}_c$	Cut-off velocity vector (ft/sec)	VC
ΔV_R	ΔV remaining at cut-off (ft/sec)	DELVR
ΔT	Cut-off time relative to launch	DELT
TF	Desired time of intercept relative to launch (sec)	TF
$\underrightarrow{r}_R$	Final intercept position (ft)	RR
$\underrightarrow{v}_V$	S/C velocity at intercept (ft/sec)	VV
$\underrightarrow{v}_{VT}$	Target velocity at intercept (ft/sec)	VVT

Table II-1 Trajectory Optimization Program Symbols (Continued)

Math * Symbol	Description	FORTRAN Program Symbol
a	Semi-major axis of orbit (ft)	A, AX
TOF	Time of flight (sec)	TOF
θ_T	Range angle (radians)	THT
$\underrightarrow{\mathrm{VFIN}}$	Arrival velocity of S/C (ft/sec)	VFIN
m_a	Arrival slope	MA

* In some cases there is no math symbol corresponding to a FORTRAN program symbol. Such cases arise when in the math flow chart, descriptive phases are used instead of FORTRAN symbols.

TABLE II-2

TEST CASES

Case	Initial State	Final Orbit	Out-of-Plane Angle (deg.)	Comments
1	Elliptical Parking Orbit Semi-Major Axis = 5208 miles Eccentricity = 0.2	Elliptical Orbit Semi-Major Axis = 6250 miles Eccentricity = 0.2	5.0	Two-Burn Orbit Transfer Without Rendezvous
2	100 n.m. circular parking orbit	Semi-Major Axis = 8444 n.m. Eccentricity = 0.5 Line of Apsides 30° from Line of Nodes	30.0	Three Burn Rendezvous with Target in Final Orbit
3	Same as 2	Same as 2	30.0	Same as above
4	Launch Site Latitude = 30°	Same Orbit as 2 and 3, Lying in the Equatorial Plane	30.0 (This is the minimum value obtainable from the Launch Site - if Launch Azimuth = 90°)	Ascent to Parking Orbit and Rendezvous with Target in Final Orbit

TABLE II-3

RESULTS OF TEST CASES

		Time of First Orbital Burn - t_1	Time of Last Orbital Burn - t_f	Slope of Arrival at Last Burn - m_{af} (1)	X - Component of First Orbital Burn - ΔV_{1x} (2)	Y - Component of First Orbital Burn - ΔV_{1y} (2)	Z - Component of First Orbital Burn - ΔV_{1z} (2)	Range Angle Between First and Second Burns (2)	Launch Time - t_L	Launch Azimuth - A_L	Circular Parking Orbit Orbit Radius - R_p	Dog - Leg Angle - α_L	Total ΔV Required	Function Evaluations	7094 Execution Time (3)
		U(1) (sec)	U(2) (sec)	U(3)	U(4) (ft/sec)	U(5) (ft/sec)	U(6) (ft/sec)	U(7) (deg)	U(12) (sec)	U(13) (deg)	U(14) (ft)	U(15) (deg)	ΔV (ft/sec)	Passes	Time
1	Input	0	2213	0	-	-	-	-	-	-	-	-	7619	-	-
	Output	552	2116	.07216	-	-	-	-	-	-	-	-	5663	155	2.78
2	Input	7515	19920	-	-173.9	-4317	-2108	208.7	-	-	-	-	13528	-	-
	Output	7825	19114	-	2262.0	-4797	-645	189.9	-	-	-	-	12715	1090	-
3	Input	2305	33026	-	358.4	-5668	-2788	203.2	-	-	-	-	11764	-	-
	Output	2370	33373	-	879.3	-6329	-1070	201.7	-	-	-	-	11496	234	9.83
4	Input	3996.2	15706	-	-	-	-	-	0	90.0	21532438	0	31673	-	-
	Output	3970.2	14769	-	-	-	-	-	0	92.7	21472348	1.287	31417	255	-

Notes:

(1.) Not required for Rendezvous Cases

(2.) Used for Three-Burn Cases

(3.) Includes I/O Time to Write Output

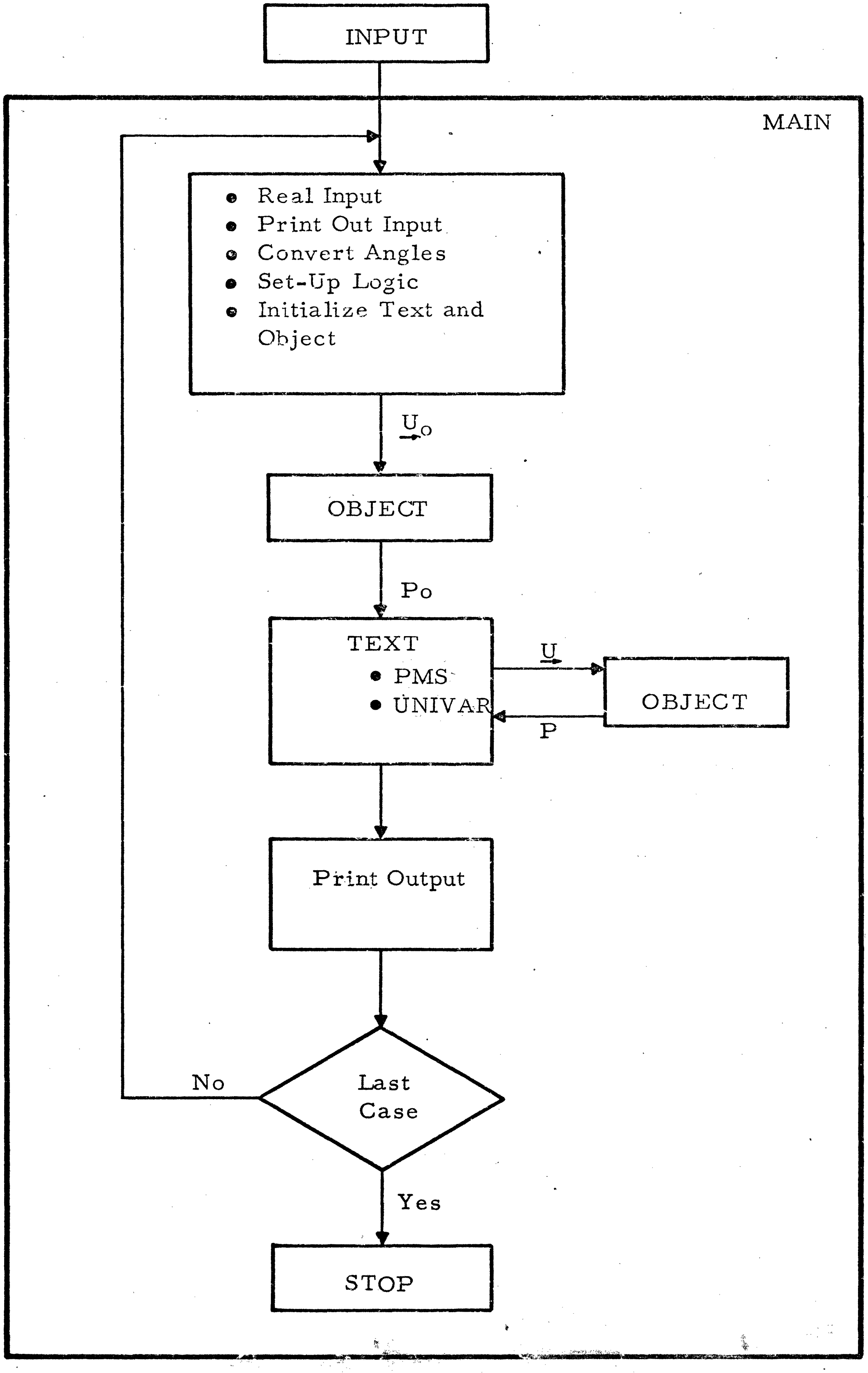


Figure II-1 Trajectory Optimizer Program

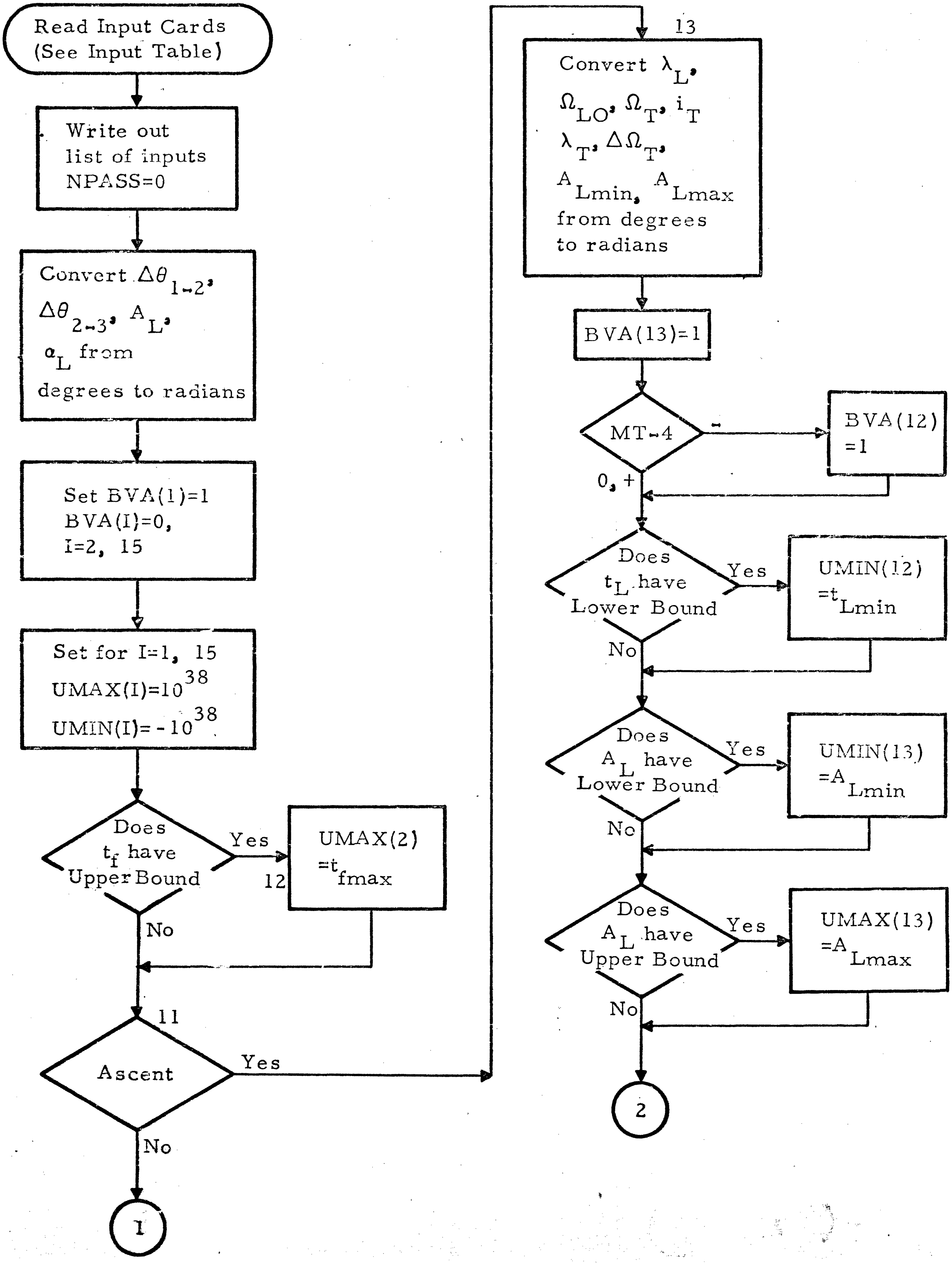

Read Input Cards (See Input Table)
Write out list of inputs NPASS=0
Convert $\Delta\theta_{1-2}$, $\Delta\theta_{2-3}$, A_L, α_L from degrees to radians
Set BVA(1)=1 BVA(I)=0, I=2, 15
Set for I=1, 15 UMAX(I)=10^{38} UMIN(I)=-10^{38}
Does t_f have Upper Bound
Yes
12
UMAX(2) =t_{fmax}
No
11
Ascent
Yes
No
1
13
Convert λ_L, Ω_{LO}, Ω_T, i_T λ_T, $\Delta\Omega_T$, A_{Lmin}, A_{Lmax} from degrees to radians
BVA(13)=1
MT-4
-
BVA(12) =1
0, +
Does t_L have Lower Bound
Yes
UMIN(12) =t_{Lmin}
No
Does A_L have Lower Bound
Yes
UMIN(13) =A_{Lmin}
No
Does A_L have Upper Bound
Yes
UMAX(13) =A_{Lmax}
No
2

Figure II-2 MAIN Program (Sheet 1 of 3)

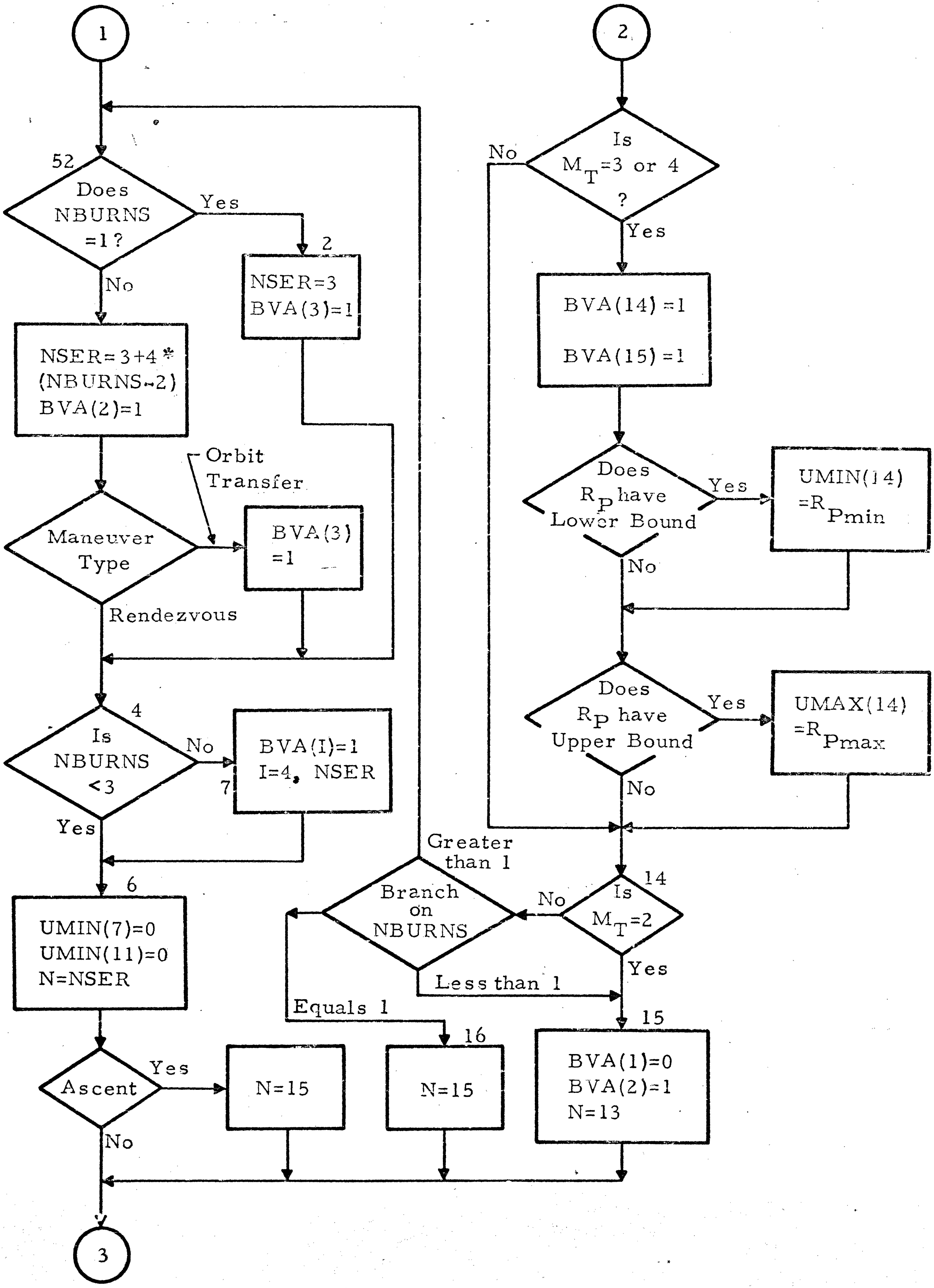


Figure II-2 MAIN Program (Sheet 2 of 3)

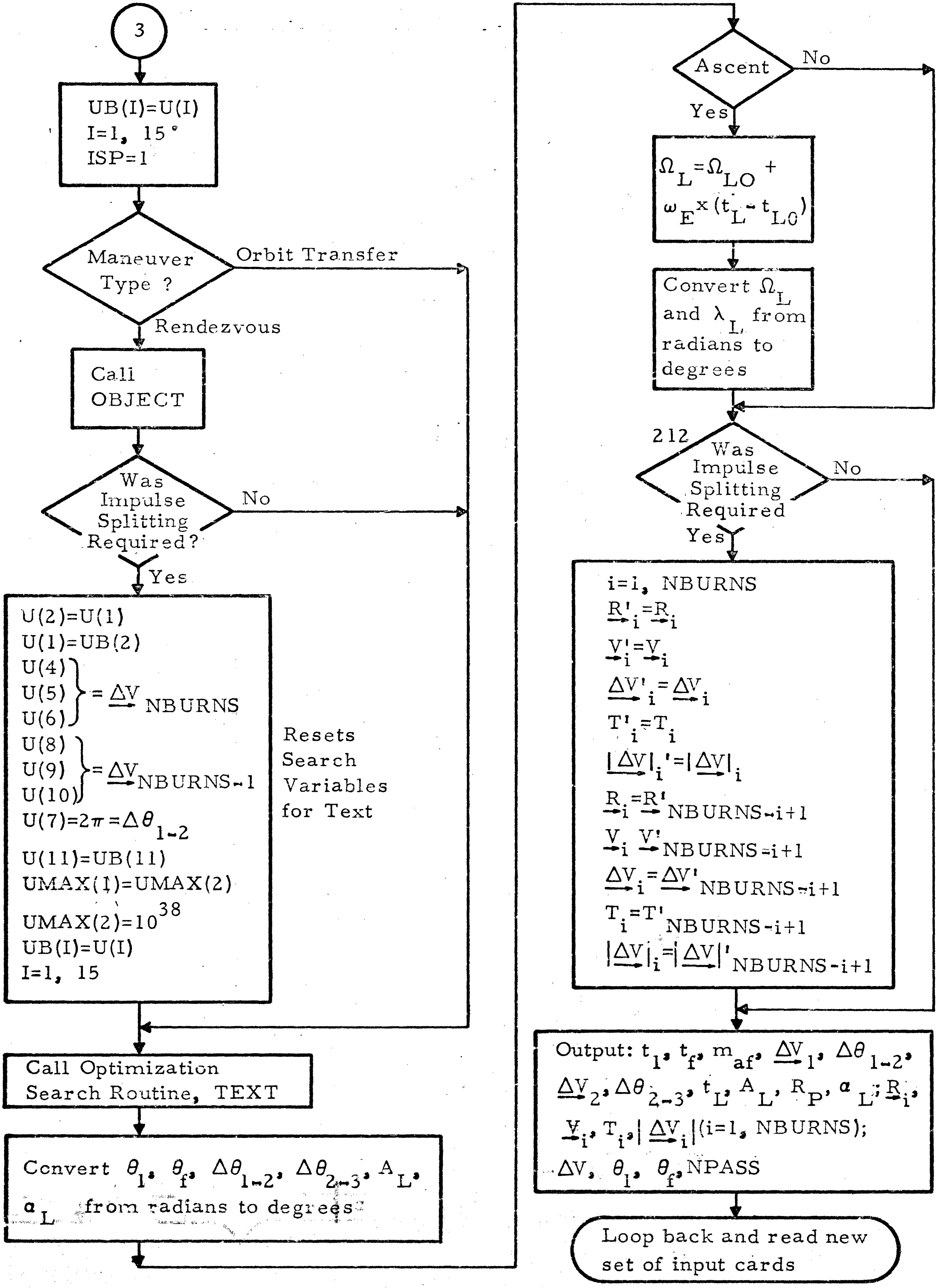

3
UB(I)=U(I)
I=1, 15
ISP=1
Maneuver Type ?
Orbit Transfer
Rendezvous
Call OBJECT
Was Impulse Splitting Required?
No
Yes
U(2)=U(1)
U(1)=UB(2)
U(4), U(5), U(6) = $\underrightarrow{\Delta V}_{NBURNS}$
U(8), U(9), U(10) = $\underrightarrow{\Delta V}_{NBURNS-1}$
U(7)=2π=$\Delta\theta_{1-2}$
U(11)=UB(11)
UMAX(1)=UMAX(2)
UMAX(2)=10^{38}
UB(I)=U(I)
I=1, 15
Resets Search Variables for Text
Call Optimization Search Routine, TEXT
Convert θ_1, θ_f, $\Delta\theta_{1-2}$, $\Delta\theta_{2-3}$, A_L, a_L from radians to degrees
Ascent
No
Yes
$\Omega_L=\Omega_{LO}$ + ω_Ex(t_L-t_{LO})
Convert Ω_L and λ_L from radians to degrees
212
Was Impulse Splitting Required
No
Yes
i=i, NBURNS
$\underrightarrow{R'}_i=\underrightarrow{R}_i$
$\underrightarrow{V'}_i=\underrightarrow{V}_i$
$\underrightarrow{\Delta V'}_i=\underrightarrow{\Delta V}_i$
$T'_i=T_i$
$|\underrightarrow{\Delta V}|_i'=|\underrightarrow{\Delta V}|_i$
$\underrightarrow{R}_i=\underrightarrow{R'}_{NBURNS-i+1}$
$\underrightarrow{V}_i$ $\underrightarrow{V'}_{NBURNS-i+1}$
$\underrightarrow{\Delta V}_i=\underrightarrow{\Delta V'}_{NBURNS-i+1}$
$T_i=T'_{NBURNS-i+1}$
$|\underrightarrow{\Delta V}|_i=|\underrightarrow{\Delta V}|'_{NBURNS-i+1}$
Output: t_1, t_f, m_{af}, $\underrightarrow{\Delta V}_1$, $\Delta\theta_{1-2}$, $\underrightarrow{\Delta V}_2$, $\Delta\theta_{2-3}$, t_L, A_L, R_P, a_L; $\underrightarrow{R}_i$, $\underrightarrow{V}_i$, T_i, $|\underrightarrow{\Delta V}_i|$(i=1, NBURNS); ΔV, θ_1, θ_f, NPASS
Loop back and read new set of input cards

Figure II-2 MAIN Program (Sheet 3 of 3)

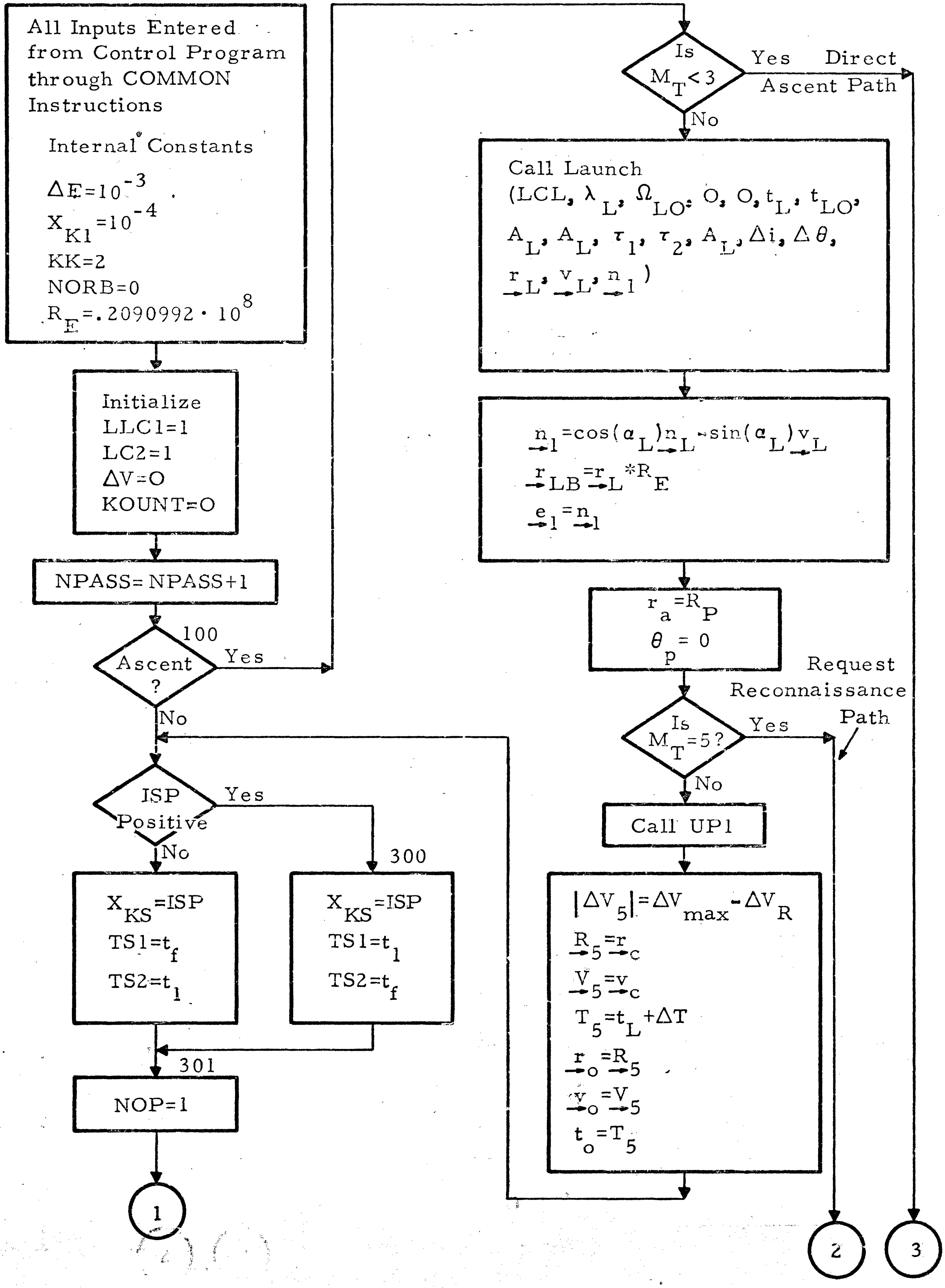

All Inputs Entered from Control Program through COMMON Instructions
Internal Constants
$\Delta E=10^{-3}$
$X_{K1}=10^{-4}$
KK=2
NORB=0
$R_E=.2090992 \cdot 10^8$
Initialize
LLC1=1
LC2=1
$\Delta V=0$
KOUNT=0
NPASS=NPASS+1
100
Ascent ?
Yes
No
ISP Positive
Yes
No
300
X_{KS}=ISP
$TS1=t_f$
$TS2=t_1$
X_{KS}=ISP
$TS1=t_1$
$TS2=t_f$
301
NOP=1
1
Is $M_T<3$
Yes
Direct Ascent Path
No
Call Launch
$(LCL, \lambda_L, \Omega_{LO}, 0, 0, t_L, t_{LO}, A_L, A_L, \tau_1, \tau_2, A_L, \Delta i, \Delta\theta, \underrightarrow{r}_L, \underrightarrow{v}_L, \underrightarrow{n}_1)$
$\underrightarrow{n}_1=\cos(\alpha_L)\underrightarrow{n}_L-\sin(\alpha_L)\underrightarrow{v}_L$
$\underrightarrow{r}_{LB}=\underrightarrow{r}_L*R_E$
$\underrightarrow{e}_1=\underrightarrow{n}_1$
$r_a=R_P$
$\theta_p=0$
Is $M_T=5$?
Yes
Request Reconnaissance Path
No
Call UP1
$|\Delta V_5|=\Delta V_{max}-\Delta V_R$
$\underrightarrow{R}_5=\underrightarrow{r}_c$
$\underrightarrow{V}_5=\underrightarrow{v}_c$
$T_5=t_L+\Delta T$
$\underrightarrow{r}_o=\underrightarrow{R}_5$
$\underrightarrow{v}_o=\underrightarrow{V}_5$
$t_o=T_5$
2
3

Figure II-3 OBJECT Program

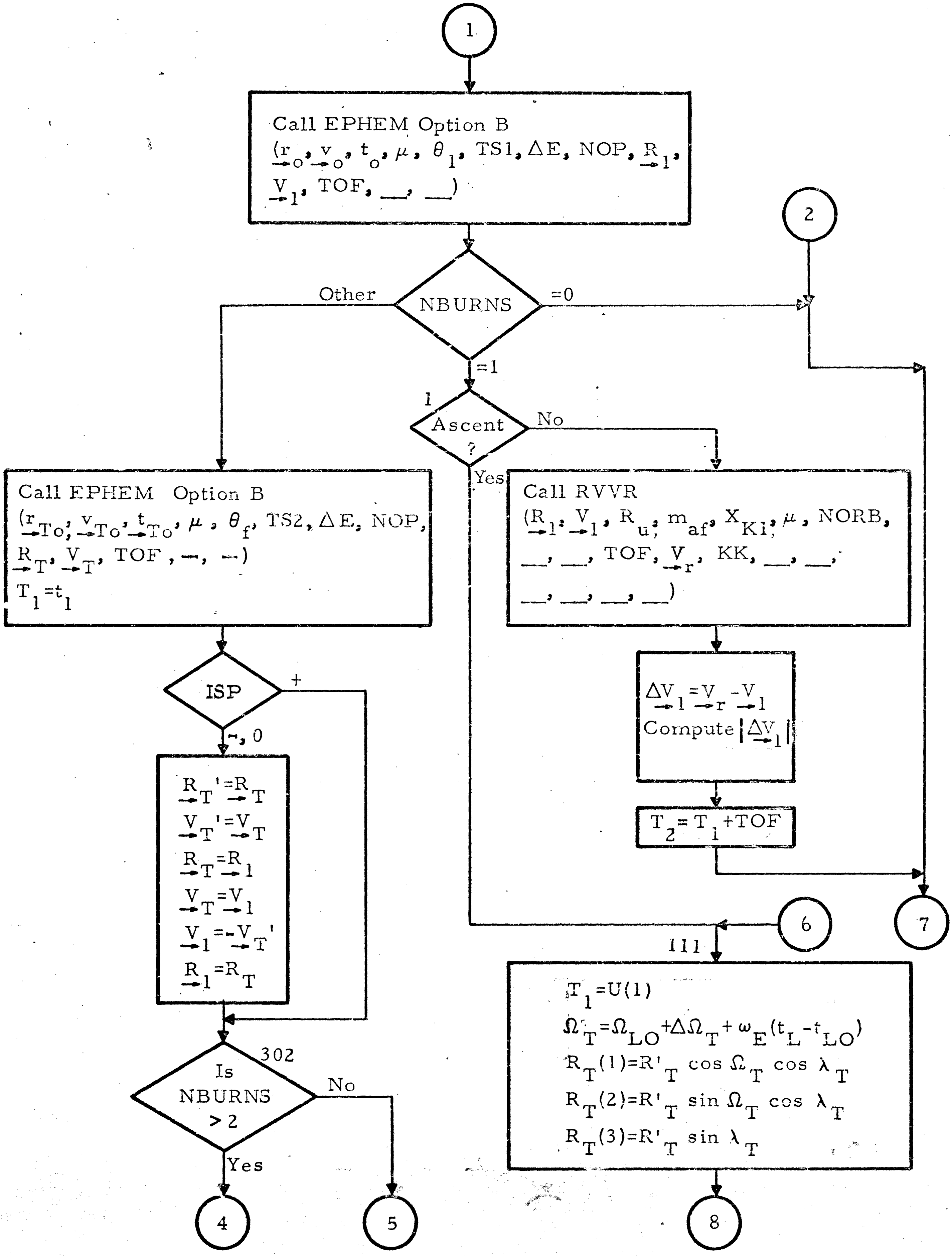

1
Call EPHEM Option B ($\underline{r}_o$, $\underline{v}_o$, t_o, μ, θ_1, TS1, ΔE, NOP, $\underline{R}_1$, $\underline{V}_1$, TOF, __, __)
2
Other
NBURNS
=0
=1
1
Ascent ?
No
Yes
Call EPHEM Option B ($\underline{r}_{To}$, $\underline{v}_{To}$, $\underline{t}_{To}$, μ, θ_f, TS2, ΔE, NOP, $\underline{R}_T$, $\underline{V}_T$, TOF, —, —)
$T_1 = t_1$
Call RVVR ($\underline{R}_1$, $\underline{V}_1$, R_u, m_{af}, X_{Ki}, μ, NORB, __, __, TOF, $\underline{V}_r$, KK, __, __, __, __, __, __)
$\Delta \underline{V}_1 = \underline{V}_r - \underline{V}_1$
Compute $|\Delta \underline{V}_1|$
$T_2 = T_1 + TOF$
ISP
+
−, 0
$\underline{R}_T' = \underline{R}_T$
$\underline{V}_T' = \underline{V}_T$
$\underline{R}_T = \underline{R}_1$
$\underline{V}_T = \underline{V}_1$
$\underline{V}_1 = -\underline{V}_T'$
$\underline{R}_1 = R_T$
6
7
111
$T_1 = U(1)$
$\Omega_T = \Omega_{LO} + \Delta\Omega_T + \omega_E(t_L - t_{LO})$
$R_T(1) = R'_T \cos \Omega_T \cos \lambda_T$
$R_T(2) = R'_T \sin \Omega_T \cos \lambda_T$
$R_T(3) = R'_T \sin \lambda_T$
302
Is NBURNS > 2
No
Yes
4
5
8

Figure II-3 OBJECT Program

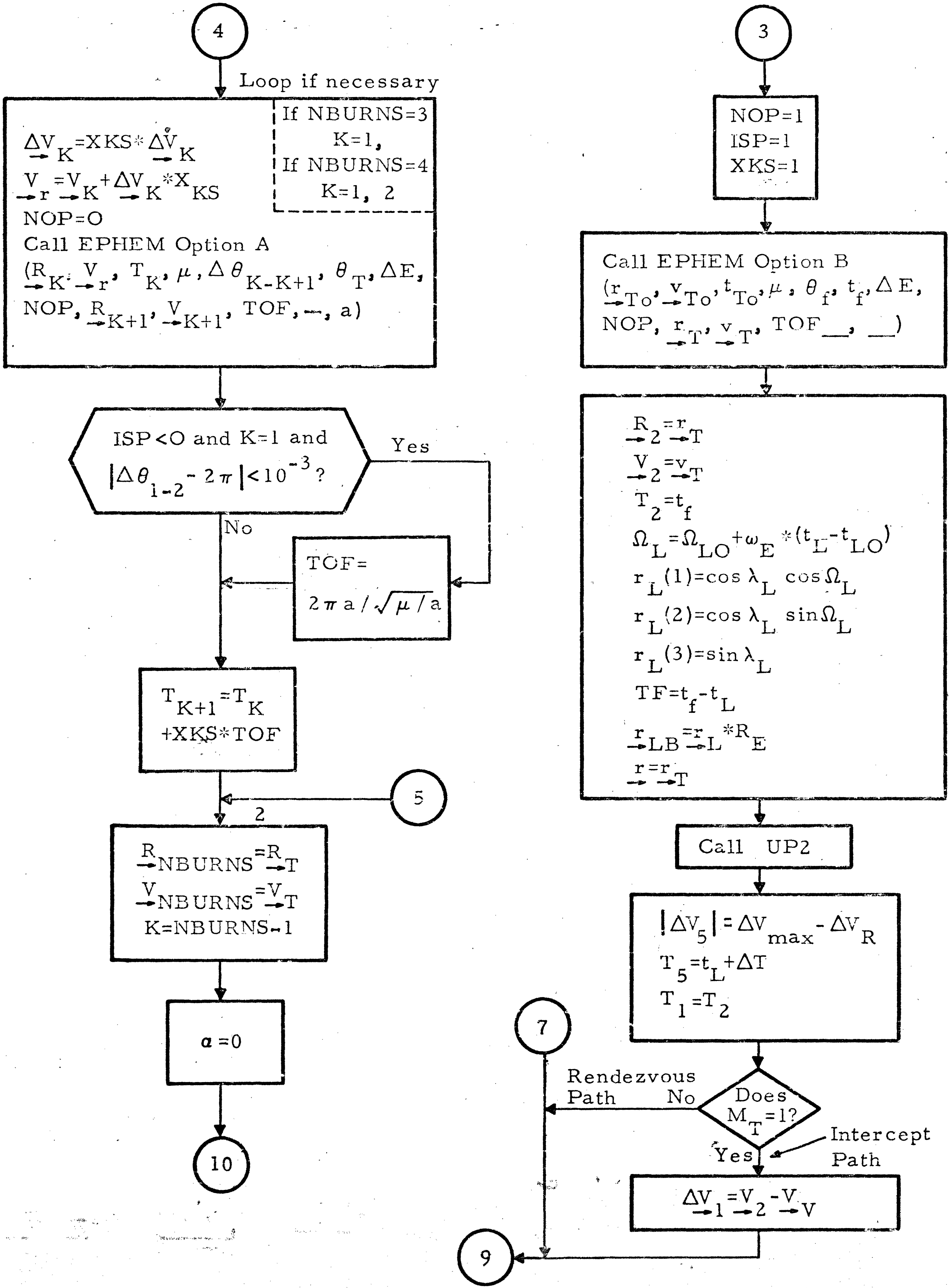

4
Loop if necessary
$\Delta \underset{\rightarrow}{V}_K = XKS * \Delta \overset{\circ}{\underset{\rightarrow}{V}}_K$
$\underset{\rightarrow}{V}_r = \underset{\rightarrow}{V}_K + \Delta \underset{\rightarrow}{V}_K * X_{KS}$
NOP=O
Call EPHEM Option A
$(\underset{\rightarrow}{R}_K, \underset{\rightarrow}{V}_r, T_K, \mu, \Delta\theta_{K-K+1}, \theta_T, \Delta E,$
$NOP, \underset{\rightarrow}{R}_{K+1}, \underset{\rightarrow}{V}_{K+1}, TOF, —, a)$
If NBURNS=3
K=1,
If NBURNS=4
K=1, 2
ISP<O and K=1 and
$|\Delta\theta_{1-2} - 2\pi| < 10^{-3}$?
Yes
No
$TOF = 2\pi a / \sqrt{\mu / a}$
$T_{K+1} = T_K + XKS * TOF$
5
2
$\underset{\rightarrow}{R}_{NBURNS} = \underset{\rightarrow}{R}_T$
$\underset{\rightarrow}{V}_{NBURNS} = \underset{\rightarrow}{V}_T$
K=NBURNS-1
$a = 0$
10
3
NOP=1
ISP=1
XKS=1
Call EPHEM Option B
$(\underset{\rightarrow}{r}_{To}, \underset{\rightarrow}{v}_{To}, t_{To}, \mu, \theta_f, t_f, \Delta E,$
$NOP, \underset{\rightarrow}{r}_T, \underset{\rightarrow}{v}_T, TOF \_\_, \_\_)$
$\underset{\rightarrow}{R}_2 = \underset{\rightarrow}{r}_T$
$\underset{\rightarrow}{V}_2 = \underset{\rightarrow}{v}_T$
$T_2 = t_f$
$\Omega_L = \Omega_{LO} + \omega_E * (t_L - t_{LO})$
$r_L(1) = \cos\lambda_L \cos\Omega_L$
$r_L(2) = \cos\lambda_L \sin\Omega_L$
$r_L(3) = \sin\lambda_L$
$TF = t_f - t_L$
$\underset{\rightarrow}{r}_{LB} = \underset{\rightarrow}{r}_L * R_E$
$\underset{\rightarrow}{r} = \underset{\rightarrow}{r}_T$
Call UP2
$|\Delta V_5| = \Delta V_{max} - \Delta V_R$
$T_5 = t_L + \Delta T$
$T_1 = T_2$
7
Rendezvous Path
No
Does $M_T = 1$?
Intercept Path
Yes
$\Delta \underset{\rightarrow}{V}_1 = \underset{\rightarrow}{V}_2 - \underset{\rightarrow}{V}_V$
9

Figure II-3 OBJECT Program

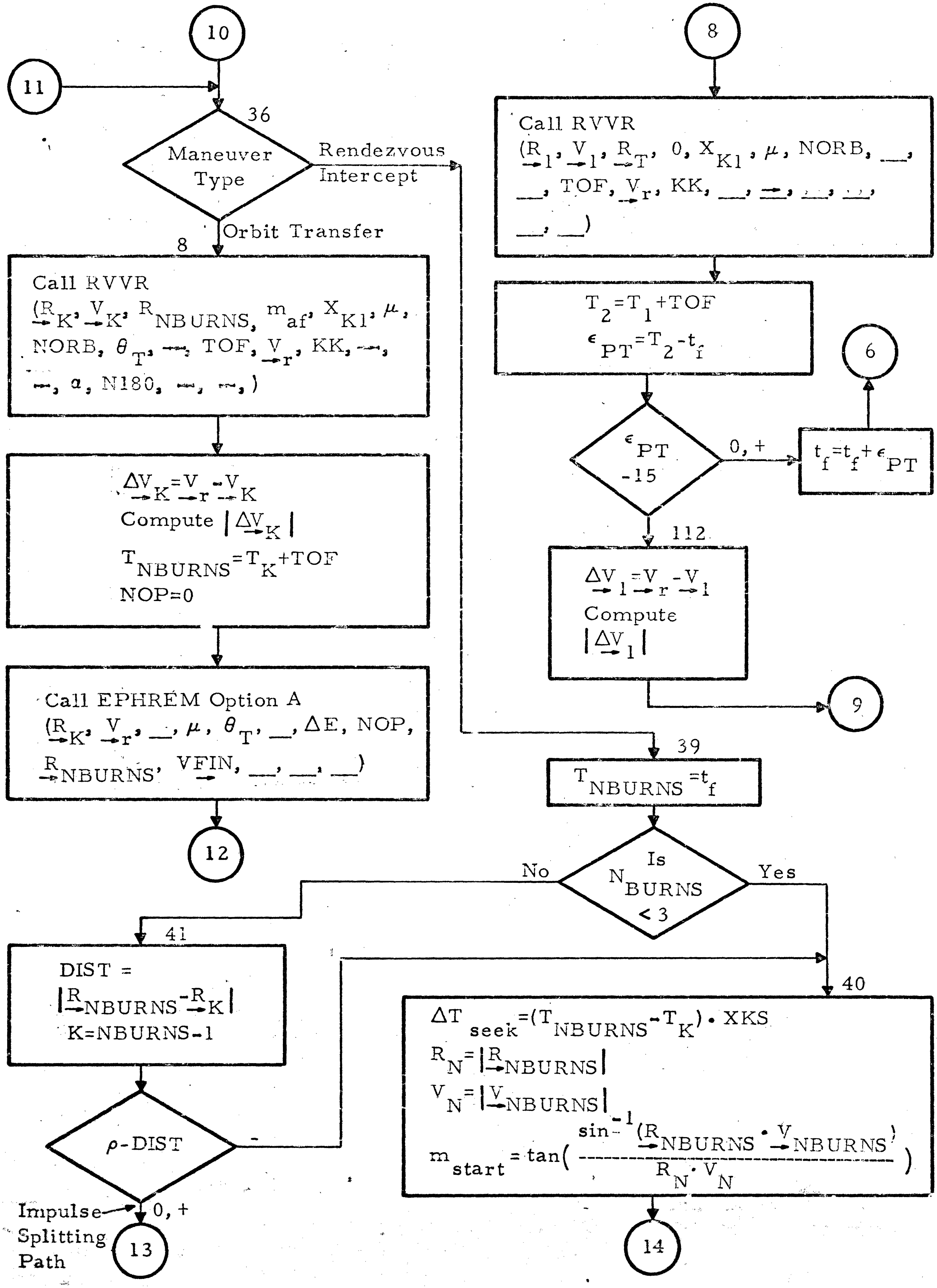

10
11
36
Maneuver Type
Rendezvous Intercept
Orbit Transfer
8
Call RVVR ($\underset{\rightarrow}{R}_K$, $\underset{\rightarrow}{V}_K$, R_{NBURNS}, m_{af}, X_{K1}, μ, NORB, θ_T, —, TOF, $\underset{\rightarrow}{V}_r$, KK, —, —, a, N180, —, —,)
$\underset{\rightarrow}{\Delta V}_K = \underset{\rightarrow}{V}_r - \underset{\rightarrow}{V}_K$
Compute $|\underset{\rightarrow}{\Delta V}_K|$
$T_{NBURNS} = T_K + TOF$
NOP=0
Call EPHREM Option A ($\underset{\rightarrow}{R}_K$, $\underset{\rightarrow}{V}_r$, —, μ, θ_T, —, ΔE, NOP, $\underset{\rightarrow}{R}_{NBURNS}$, $\underset{\rightarrow}{VFIN}$, —, —, —)
12
8
Call RVVR ($\underset{\rightarrow}{R}_1$, $\underset{\rightarrow}{V}_1$, $\underset{\rightarrow}{R}_T$, 0, X_{K1}, μ, NORB, —, —, TOF, $\underset{\rightarrow}{V}_r$, KK, —, —, —, —, —, —)
$T_2 = T_1 + TOF$
$\epsilon_{PT} = T_2 - t_f$
ϵ_{PT} -15
0, +
$t_f = t_f + \epsilon_{PT}$
6
112
$\underset{\rightarrow}{\Delta V}_1 = \underset{\rightarrow}{V}_r - \underset{\rightarrow}{V}_1$
Compute $|\underset{\rightarrow}{\Delta V}_1|$
9
39
$T_{NBURNS} = t_f$
Is $N_{BURNS} < 3$
No
Yes
41
DIST = $|\underset{\rightarrow}{R}_{NBURNS} - \underset{\rightarrow}{R}_K|$
K=NBURNS-1
ρ-DIST
-
0, +
Impulse Splitting Path
13
40
$\Delta T_{seek} = (T_{NBURNS} - T_K) \cdot XKS$
$R_N = |\underset{\rightarrow}{R}_{NBURNS}|$
$V_N = |\underset{\rightarrow}{V}_{NBURNS}|$
$m_{start} = \tan\left(\frac{\sin^{-1}(\underset{\rightarrow}{R}_{NBURNS} \cdot \underset{\rightarrow}{V}_{NBURNS})}{R_N \cdot V_N}\right)$
14

Figure II-3 OBJECT Program

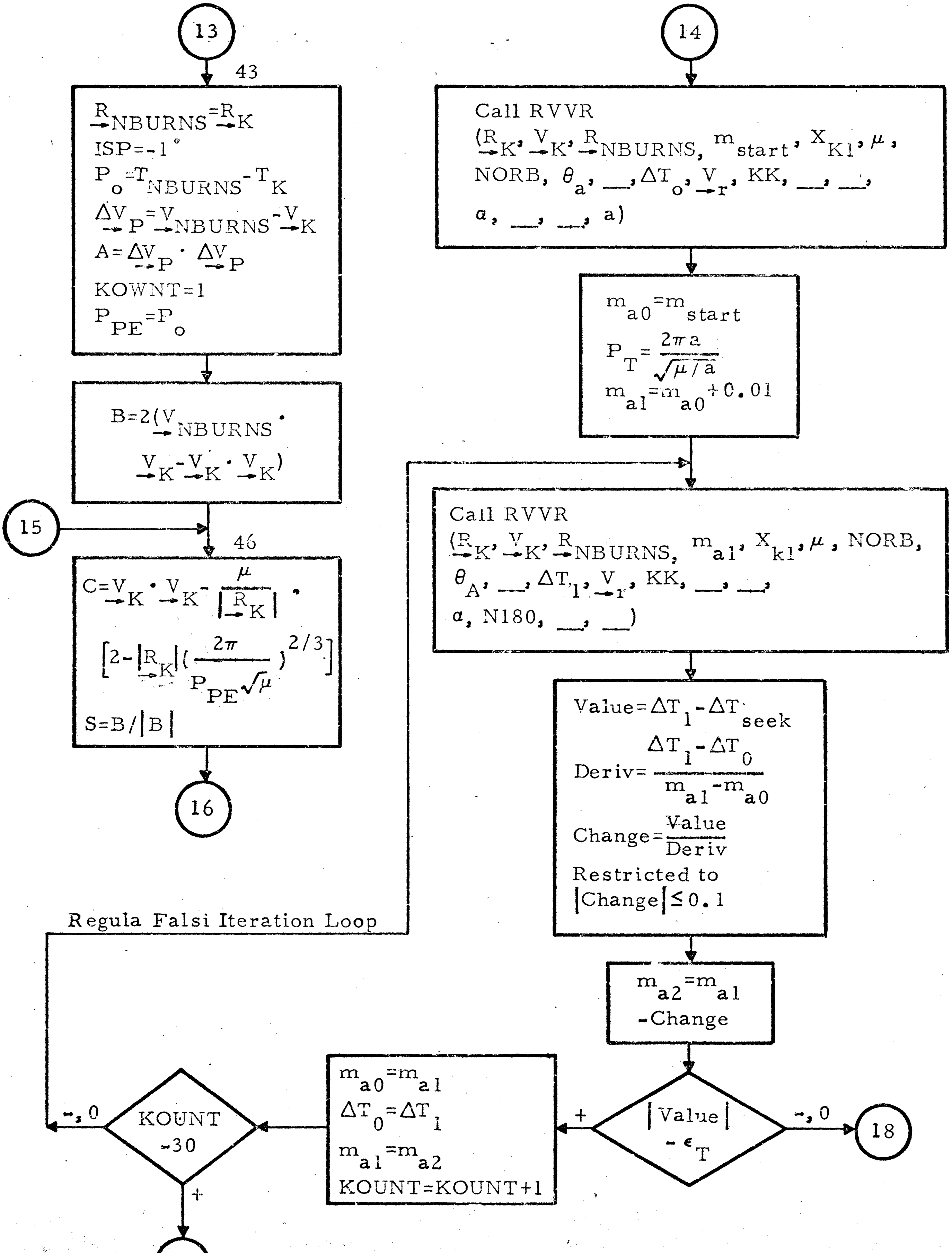

13
43
$\vec{R}_{NBURNS}=\vec{R}_K$
ISP=-1
$P_o=T_{NBURNS}-T_K$
$\Delta\vec{V}_P=\vec{V}_{NBURNS}-\vec{V}_K$
$A=\Delta\vec{V}_P\cdot\Delta\vec{V}_P$
KOWNT=1
$P_{PE}=P_o$
$B=2(\vec{V}_{NBURNS}\cdot\vec{V}_K-\vec{V}_K\cdot\vec{V}_K)$
15
46
$C=\vec{V}_K\cdot\vec{V}_K-\frac{\mu}{|\vec{R}_K|}\cdot\left[2-|\vec{R}_K|\left(\frac{2\pi}{P_{PE}\sqrt{\mu}}\right)^{2/3}\right]$
$S=B/|B|$
16
14
Call RVVR
$(\vec{R}_K, \vec{V}_K, \vec{R}_{NBURNS}, m_{start}, X_{K1}, \mu,$
$NORB, \theta_a, \_\_, \Delta T_o, \vec{V}_r, KK, \_\_, \_\_,$
$\alpha, \_\_, \_\_, a)$
$m_{a0}=m_{start}$
$P_T=\frac{2\pi a}{\sqrt{\mu/a}}$
$m_{a1}=m_{a0}+0.01$
Call RVVR
$(\vec{R}_K, \vec{V}_K, \vec{R}_{NBURNS}, m_{a1}, X_{k1}, \mu, NORB,$
$\theta_A, \_\_, \Delta T_1, \vec{V}_r, KK, \_\_, \_\_,$
$\alpha, N180, \_\_, \_\_)$
$Value=\Delta T_1-\Delta T_{seek}$
$Deriv=\frac{\Delta T_1-\Delta T_0}{m_{a1}-m_{a0}}$
$Change=\frac{Value}{Deriv}$
Restricted to
$|Change|\leq 0.1$
$m_{a2}=m_{a1}-Change$
$|Value|-\epsilon_T$
-, 0
18
+
$m_{a0}=m_{a1}$
$\Delta T_0=\Delta T_1$
$m_{a1}=m_{a2}$
KOUNT=KOUNT+1
KOUNT-30
-, 0
+
17
Regula Falsi Iteration Loop

Figure II-3 OBJECT Program

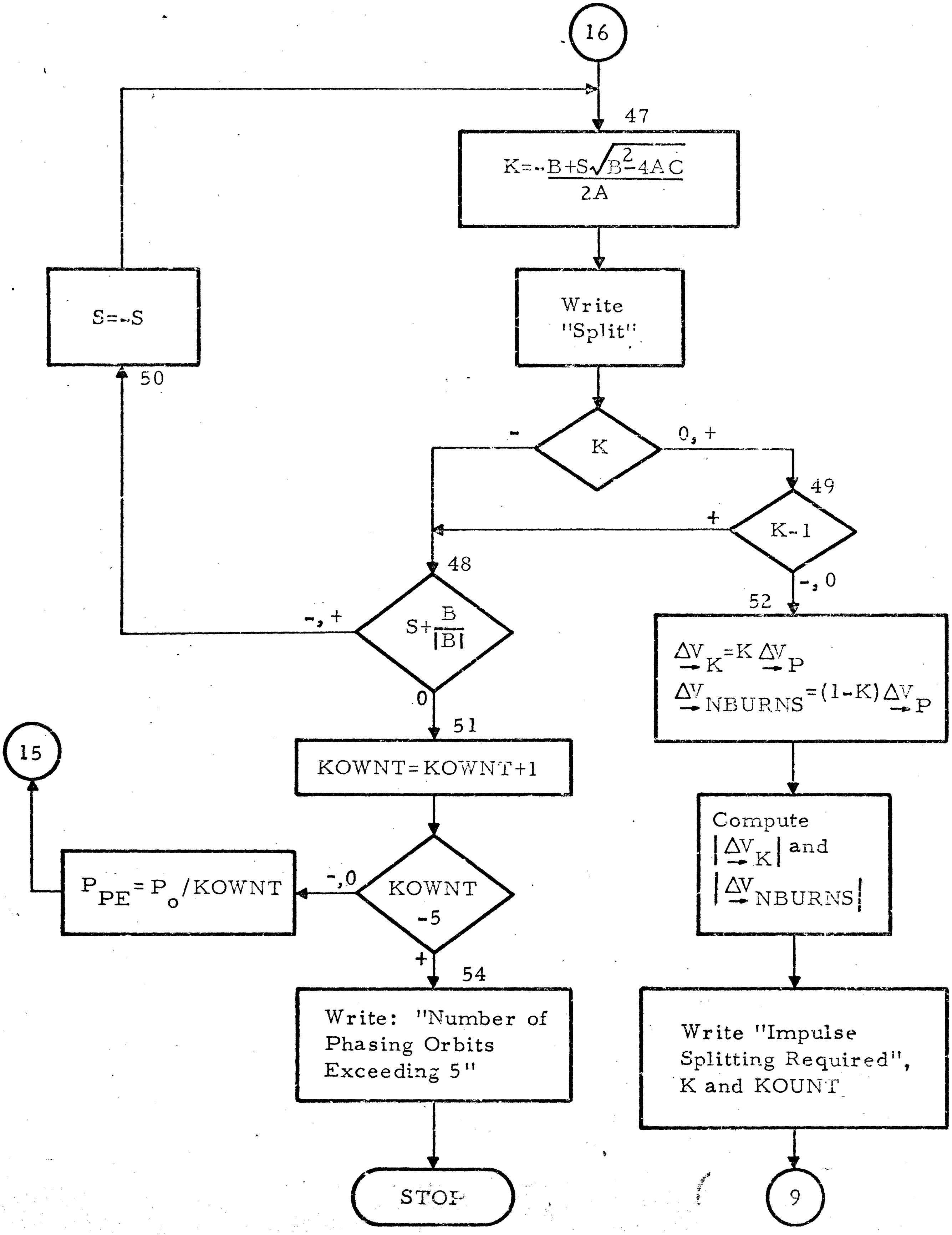
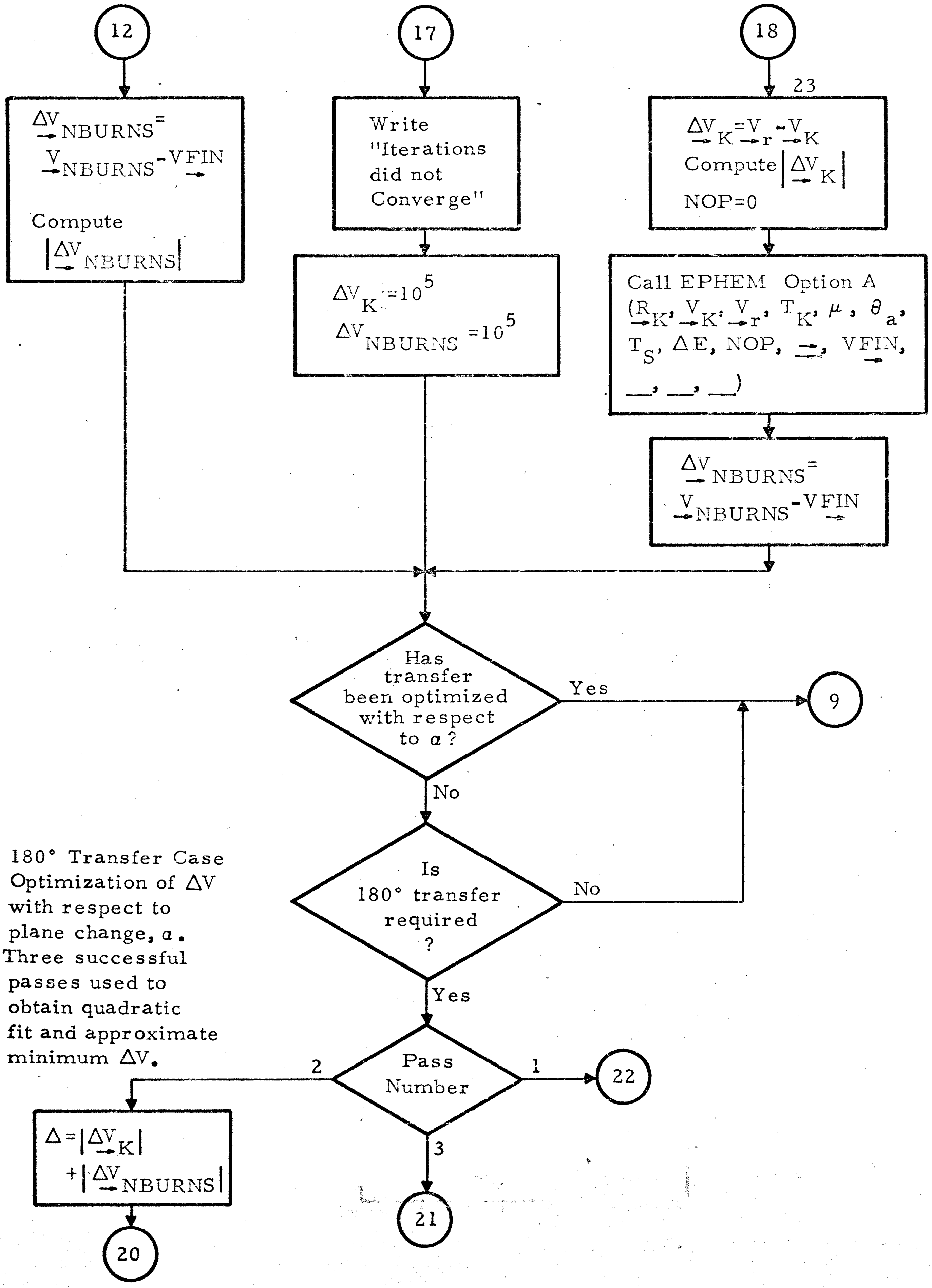


Figure II-3 OBJECT Program


12
17
18
23
$\underset{\rightarrow}{\Delta V}_{NBURNS} = \underset{\rightarrow}{V}_{NBURNS} - \underset{\rightarrow}{VFIN}$
Compute $|\underset{\rightarrow}{\Delta V}_{NBURNS}|$
Write "Iterations did not Converge"
$\Delta V_K = 10^5$
$\Delta V_{NBURNS} = 10^5$
$\underset{\rightarrow}{\Delta V}_K = \underset{\rightarrow}{V}_r - \underset{\rightarrow}{V}_K$
Compute $|\underset{\rightarrow}{\Delta V}_K|$
NOP=0
Call EPHEM Option A ($\underset{\rightarrow}{R}_K$, $\underset{\rightarrow}{V}_K$, $\underset{\rightarrow}{V}_r$, T_K, μ, θ_a, T_S, ΔE, NOP, $\underset{\rightarrow}{\_\_}$, $\underset{\rightarrow}{VFIN}$, __, __, __)
$\underset{\rightarrow}{\Delta V}_{NBURNS} = \underset{\rightarrow}{V}_{NBURNS} - \underset{\rightarrow}{VFIN}$
Has transfer been optimized with respect to α ?
Yes
9
No
Is 180° transfer required ?
No
Yes
Pass Number
2
1
22
3
21
$\Delta = |\underset{\rightarrow}{\Delta V}_K| + |\underset{\rightarrow}{\Delta V}_{NBURNS}|$
20

180° Transfer Case Optimization of ΔV with respect to plane change, α. Three successful passes used to obtain quadratic fit and approximate minimum ΔV.

Figure II-3 OBJECT Program

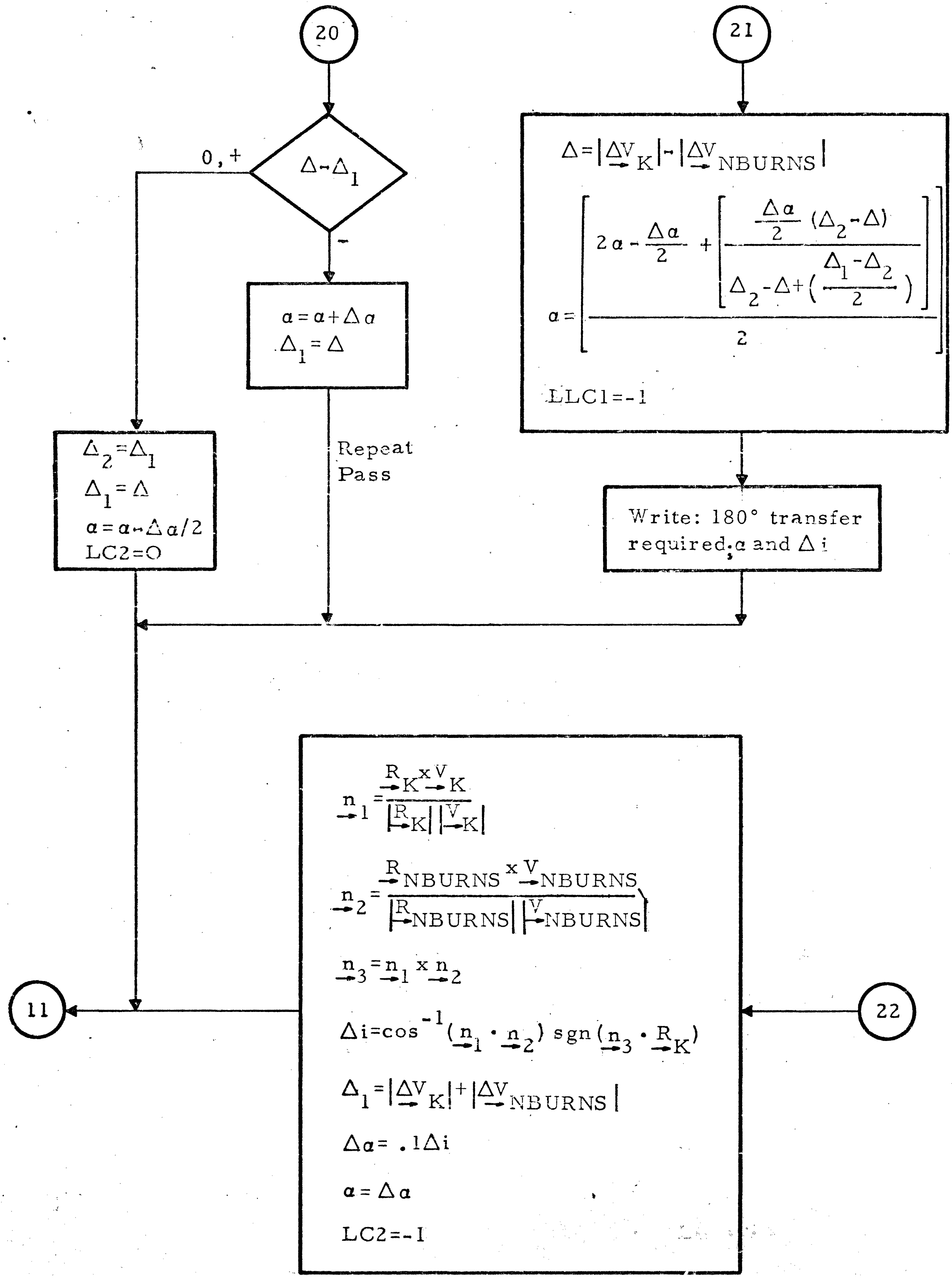

20
$\Delta - \Delta_1$
0, +
-
$a = a + \Delta a$
$\Delta_1 = \Delta$
Repeat Pass
$\Delta_2 = \Delta_1$
$\Delta_1 = \Delta$
$a = a - \Delta a/2$
LC2=O
21
$\Delta = |\underset{\rightarrow}{\Delta V}_K| - |\underset{\rightarrow}{\Delta V}_{NBURNS}|$
$a = \left[\dfrac{2a - \dfrac{\Delta a}{2} + \left[\dfrac{\dfrac{\Delta a}{2}(\Delta_2 - \Delta)}{\Delta_2 - \Delta + \left(\dfrac{\Delta_1 - \Delta_2}{2}\right)}\right]}{2}\right]$
LLC1=-1
Write: 180° transfer required; a and Δ i
11
22
$\underset{\rightarrow}{n}_1 = \dfrac{\underset{\rightarrow}{R}_K \times \underset{\rightarrow}{V}_K}{|\underset{\rightarrow}{R}_K| |\underset{\rightarrow}{V}_K|}$
$\underset{\rightarrow}{n}_2 = \dfrac{\underset{\rightarrow}{R}_{NBURNS} \times \underset{\rightarrow}{V}_{NBURNS}}{|\underset{\rightarrow}{R}_{NBURNS}| |\underset{\rightarrow}{V}_{NBURNS}|}$
$\underset{\rightarrow}{n}_3 = \underset{\rightarrow}{n}_1 \times \underset{\rightarrow}{n}_2$
$\Delta i = \cos^{-1}(\underset{\rightarrow}{n}_1 \cdot \underset{\rightarrow}{n}_2) \,\mathrm{sgn}(\underset{\rightarrow}{n}_3 \cdot \underset{\rightarrow}{R}_K)$
$\Delta_1 = |\underset{\rightarrow}{\Delta V}_K| + |\underset{\rightarrow}{\Delta V}_{NBURNS}|$
$\Delta a = .1 \Delta i$
$a = \Delta a$
LC2=-1

Figure II-3 OBJECT Program

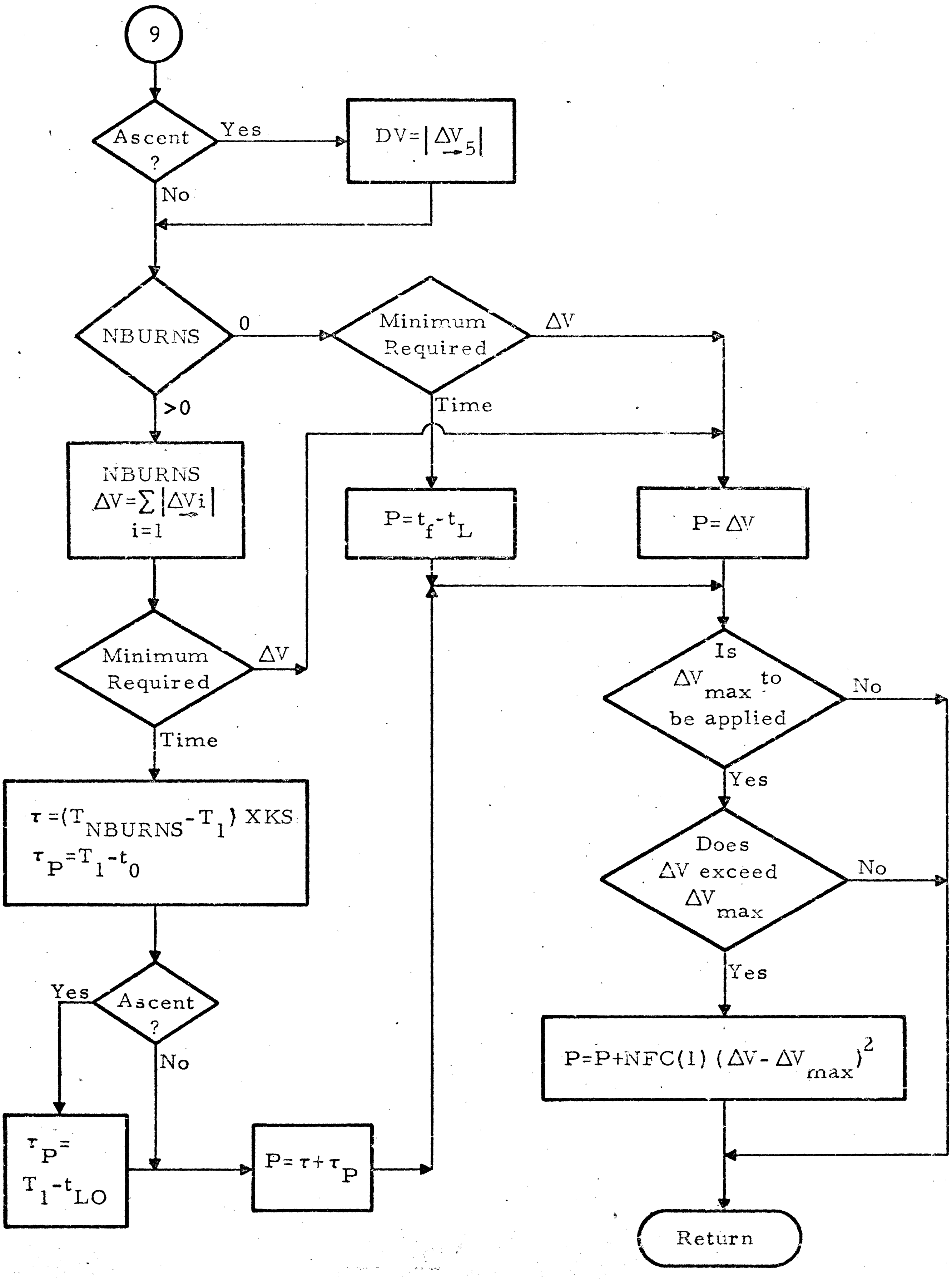


Figure II-3 OBJECT Program

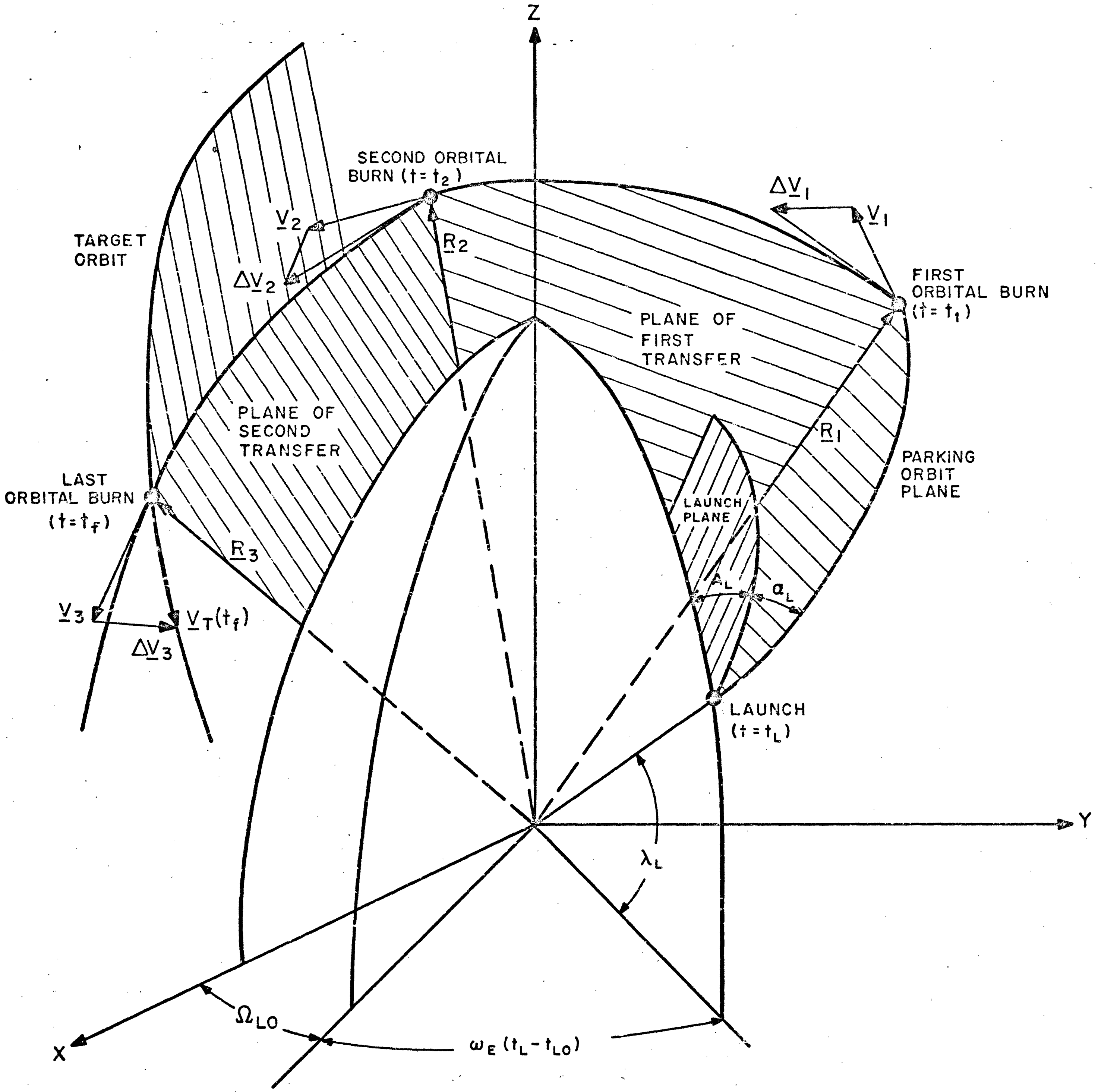


Figure II-4 COORDINATE SYSTEM AND TRAJECTORY SPECIFICATION

REFERENCES

1. Robbins, H. M., "An Analytical Study of the Impulsive Approximation," AIAA Journal, Vol. 4, No. 8, August 1966, pp 1417 - 1423.

2. McCue, G. A., "Optimum Two - Impulse Orbital Transfer and Rendezvous Between Inclinded Elliptical Orbits," AIAA, August 1963, pp 1865 - 1872.

Appendix III (Unclassified)

OPTIMAL-EXPLICIT GUIDANCE (OP-EX) FOR POWERED FLIGHT OUTSIDE THE ATMOSPHERE

This appendix describes the theory and implementation of OP-EX, an IBM-developed guidance algorithm for all phases of powered flight outside the atmosphere, including the exoatmospheric phase of ascent (i.e., that phase of ascent where the dynamic pressure is less than 30 p.s.f.). As its name implies, OP-EX is both optimal and explicit. That is, it accepts as inputs a given "present" state and propulsion performance models for the present and future stages of the rocket, and generates an optimal (fuel-minimal) trajectory which satisfies explicitly stated constraints and final conditions, without dependence on off-board precomputations. OP-EX guidance is compatible with and is used by the Optimal-Explicit Flight Planner.

3.1 SUMMARY OF CAPABILITIES AND ADVANTAGES

OP-EX is a versatile guidance algorithm usable for exoatmospheric ascent, orbit transfers, direct rendezvous, direct intercept, deboost - in fact, for all powered phases outside the atmosphere. For each of these guidance requirements, the resulting fuel expenditure is minimal.

Also, for any given guidance phase, a variety of alternative terminal conditions can be easily specified. For example, orbit insertions can be achieved with specification of (1) orbital plane and orbit orientation, size and shape in that plane, or (2) orbit inclination, latitude of perigee, size and shape, or (3) orbit inclination and latitude and longitude of orbit perigee. In fact, almost any reasonable combination of termination conditions imaginable can be easily implemented. All that is required is the specification of six "terminal-error" equations (according to a set of rules which will be derived) which completely define the mission type. These equations, when satisfied, determine the unique optimal trajectory.

OP-EX can optimally perform any magnitude of plane change within the vehicle's capabilities during ascent. It can also optimally perform any orbital maneuvers for plane change, orbit transfer, rendezvous, or intercept.

OP-EX has flexible provisions for propulsion performance description (thrusts, stage masses, mass rates). An unlimited number of stages can be specified and, in theory, any modelable function of time can be used to represent engine mass flow and exhaust velocity. Even exotic schemes like the Saturn V PU (propellant-utilization) system, which involves a change of thrust level at a variable time after stage ignition, can be implemented satisfactorily. In fact, the optimal time at which to switch the level of thrust could be specified by OP-EX.

The OP-EX algorithm is computationally efficient, to be practical for real-time guidance, yet is sufficiently accurate to be used for rapid prediction of ΔV requirements for flight-planning purposes.

OP-EX provides predicted vehicle states at future staging times (and other critical times) which can be used in spent-booster impact prediction.

OP-EX has growth capability for the optimal inclusion of additional constraints such as fixed final attitude, recovery ceiling, etc. Implementation would require little more than a modification of the control law; the routines for trajectory generation and for iterative adjustment of trajectory parameters would be essentially unchanged.

3.2 DERIVATION OF EQUATIONS FOR OPTIMAL TRAJECTORIES

The equations of motion for rocket-powered flight outside the atmosphere are

$$\dot{\underline{r}} = \underline{v} \tag{1}$$

$$\dot{\underline{v}} = \underline{g}\,(\underline{r}, t) + a\,(t)\,\hat{\underline{u}}\,(t) \tag{2}$$

where $\underline{r}$ is the position vector, $\underline{v}$ is the velocity vector, $\underline{g}$ is the acceleration due to gravity, $a\,(t)$ is the magnitude of thrust acceleration, and $\hat{\underline{u}}(t)$ is a unit vector giving the direction of thrust acceleration. For the one-burn case with non-throttleable engines, $a\,(t)$ may be regarded as a given function of time, but in more complicated cases a propulsion-model must be included in the problem formulation. For an n-stage rocket, equations determining $a\,(t)$ are

$$\left.\begin{aligned} a &= T_j/M_j \\ \dot{M}_j &= -T_j/c^*_j + \omega_j \\ O \le T_j &\le T_{j\,max} \end{aligned}\right\} \quad j = 1, 2, \ldots n \tag{3}$$

where M is vehicle mass, T is thrust, T_{max} is full-throttle thrust, c* is the effective exhaust velocity, and ω is inert mass flow rate due to scheduled dumping of expendables and propellant leakage and evaporation. The effect of ω is, in general, very small, so to avoid needlessly complicating the discussion, it will be neglected. (Should ω become significant, the theory can, of course, be expanded to include its effect.) c*, T_{max}, and M may change discontinuously at staging times. Also, c* and T_{max} may be functions of the burning time since stage ignition. (For solid fuel rockets, this is usually the case.) The engines may be non-throttleable, non-restateable or both. However, for formulating the optimization theory, it is convenient to begin by considering perfectly throttleable engines. For cases in which staging is triggered by fuel depletion, c* and T_{max} may be regarded as given functions of the mass M. Also, (as a mathematical artifice) mass discontinuities at staging may be regarded as brief intervals with very small c* and correspondingly high values of $\dot{M}$.

Introducing a throttle variable S, Equation (3) can be rewritten as

$$a = S a_{max}(t) \tag{4}$$

$$a_{max} = T_{max}(M)/M \tag{5}$$

$$\dot{M} = -S\,T_{max}(M)/c^*(M) \tag{6}$$

$$O \leq S \leq 1 \tag{7}$$

The optimization problem is to choose control policies $\hat{\underline{u}}(t)$ and S (t) such that the resulting trajectory satisfies given final conditions and extremizes some function of the final values of $\underline{r}$, $\underline{v}$, M and t. Usually, but not always, the quantity to be extremized is the final mass $M(t_f)$.

This is an optimization problem of the Mayer type. It can be expressed in standard form by introducing the seven-component state vector

$$\underline{x} = \begin{bmatrix} \underline{r} \\ \underline{v} \\ M \end{bmatrix} \tag{8}$$

and the four-component vector

$$\underline{\upsilon} = \begin{bmatrix} \hat{\underline{u}} \\ \\ S \end{bmatrix} \tag{9}$$

Then the state equations can be expressed in the standard form

$$\dot{\underline{x}} = \underline{f}\,(\underline{x},\ \underline{\upsilon},\ t) \tag{10}$$

where

$$\underline{f} = \begin{bmatrix} \underline{v} \\ \underline{g}(\underline{r},\ t) + S\hat{\underline{u}}\, a_{max}\ (M) \\ -S\, T_{max}\ (M)/c^{*}(M) \end{bmatrix} \tag{11}$$

The control constraints $|\hat{\underline{u}}| = 1$ and $O \leq S \leq 1$ are independent of the state variables, so Pontriagin's maximum principle is applicable. Introducing the adjoint vector

$$\underline{p} = \begin{bmatrix} \underline{\eta} \\ \underline{\lambda} \\ \sigma \end{bmatrix} \tag{12}$$

the Hamiltonian can be written as

$$H(\underline{x},\ \underline{p},\ \underline{\upsilon},\ t) = \underline{p}^{T}\underline{f}\,(\underline{x},\ \underline{\upsilon},\ t) \tag{13a}$$

$$= \underline{\eta} \cdot \underline{v} + \underline{\lambda} \cdot \underline{g} + S\left[\underline{\lambda} \cdot \hat{\underline{u}}\, a_{max} - \sigma T_{max}/c^{*}\right] \tag{13b}$$

It is convenient to introduce the abbreviation K for the bracketed quantity in Equation (13), and G as the symbol for the gravity-gradient matrix. Then the standard adjoint equation

$$\dot{\underline{p}} = -\partial H/\partial x \tag{14}$$

gives

$$\dot{\underline{\lambda}} = -\underline{\eta} \tag{15}$$

$$\dot{\underline{\eta}} = -G\underline{\lambda} \tag{16}$$

$$\dot{\sigma} = -S\,\partial K/\partial M \tag{17}$$

The vector $\underline{\lambda}$ is Derek Lawden's "primer vector."

The maximum principle requires

$$\hat{\underline{u}} = \underline{\lambda}/\lambda \tag{18}$$

and

$$\dot{S} = \begin{cases} I \text{ if } K > O \\ O \text{ if } K < O \end{cases} \tag{19}$$

An immediate consequence of Equations (6) and (17) is

$$\dot{K} = (\underline{\lambda} \cdot \underline{\hat{u}})^{\cdot} \; a_{max} \tag{20}$$

which simplifies to

$$\dot{K} = |\lambda|^{\cdot} \; a_{max} \tag{21}$$

when use is made of Equation (18). This relation enables K to be computed directly, so the quantity σ becomes unnecessary and need not be computed.

In many cases, the control of thrust magnitude is prescribed or trivial (full on till cutoff, zero thereafter) so only the optimization of steering need be considered. For such cases, the differential equations for an optimal trajectory reduce to

$$\ddot{\underline{r}} = \dot{\underline{v}} = \underline{g}(\underline{r},\ t) + a\,(t)\,\underline{\lambda}/\lambda \tag{22}$$

$$\ddot{\underline{\lambda}} = G\underline{\lambda} \tag{23}$$

For an inverse-square central field, the quantities $\underline{g}$ and G are given by

$$\underline{g} = -(\mu/r^3)\underline{r} \tag{24}$$

$$G = -(\mu/r^3)\left[I - (3/r^2)\underline{r}\underline{r}^T\right] \tag{25}$$

where $\underline{r}^T$ is the transpose of the column vector $\underline{r}$. For multiburn cases, or cases in which the trajectory starts from a parking orbit with a freely chosen ignition time, Equations (22) through (25) must be supplemented by (19) and (21).

Equation (21) shows that the quantities K and $|\underline{\lambda}|$ always increase or decrease together, so local maxima or minima of K along an optimum trajectory are maxima or minima of $|\underline{\lambda}|$ also. During coast phases, M and a_{max} (M) are assumed constant, so K is linearly related to $|\underline{\lambda}|$. (Should an appreciable mass change occur during coast, due to scheduled dumping of expendables, propellant leakage or evaporation, or all three, the formulation could be expanded to include its effect.) Consequently, the magnitude of $|\underline{\lambda}|$ must be the same at both ends of a coast phase. The powered phases occur at local maxima of $|\underline{\lambda}(t)|$. These maxima may occur at the beginning or end of the trajectory (exterior maxima) or at intermediate points (interior maxima). For interior maxima (i.e., burns whose beginning and end times are not determined by any consideration other than optimality), the conditions for optimal thrust control require the integral

$$J = \int_{burn} \dot{|\lambda|} a_{max} dt \qquad (26)$$

to vanish. This requiremenc does not hold for a burn whose beginning time or end time is constrained.

The optimality conditions given above reduce the problem of trajectory optimization to a two-point boundary value problem (TPBVP). The initial conditions generally consist of given values of $\underline{r}(t_o)$, $\underline{v}(t_o)$, $M(t_o)$, and t_o. For cases in which the final mass is to be maximized, the final conditions consist of $k \leq 6$ mission conditions (prescribed relations among the final values of $\underline{r}(t_f)$, $\underline{v}(t_f)$, and t_f plus 6-k transversality conditions which come from optimal control theory. The transversality conditions can be analytically derived from the mission conditions by the requirement that

$$\dot{\underline{\lambda}} \cdot \underline{\delta r} - \underline{\lambda} \cdot \underline{\delta v} = 0 \qquad t \geq t_f \qquad (27)$$

must hold for every pair of infinitesimal variations $\underline{\delta r}$, $\underline{\delta v}$ consistent with the mission conditions and at every point on the coast trajectory.

3.3 NUMERICAL SOLUTION OF THE OPTIMAL TRAJECTORY PROBLEM

As derived in the previous section of this appendix, the differential equations defining the optimal trajectory of a rocket powered vehicle are

$$\ddot{\underline{r}} = \dot{\underline{v}} = \underline{g}(r) + a(t)\,\hat{\underline{u}} \qquad (28a)$$

$$\ddot{\underline{\lambda}} = G\underline{\lambda} \qquad (28b)$$

$$\hat{\underline{u}} = \underline{\lambda}/\lambda \qquad (28c)$$

A guidance scheme also based on Equations (28), but differing from OP-EX in computational techniques, has been described by Brown and Johnson (Reference 1). Numerical integrating of these equations from some initial time t_o to an estimated final time t_f, starting from the initial state $\underline{r}(t_o)$ and $\underline{v}(t_o)$ and estimated initial costate $\underline{\lambda}(t_o)$ and $\dot{\underline{\lambda}}(t_o)$, determines the final state, $\underline{r}(t_f)$ and $\underline{v}(t_f)$, and the final costate variables $\underline{\lambda}(t_f)$ and $\dot{\underline{\lambda}}(t_f)$. Seven error quantities, derived from k specified final state conditions and 6-k transversality conditions and a desired normalization of $\underline{\lambda}(t_o)$, must be brought to null by iterative adjustment of the seven parameters $\underline{\lambda}(t_o)$, $\dot{\underline{\lambda}}(t_o)$ and t_f. To this end a Newton-Raphson iteration procedure will be used.

Since the same trajectory integration algorithm is to be used for both flight planning of all powered phases and for operational guidance during the powered phases, the integration must be both fast and accurate. An obvious method of increasing the speed of solution is to increase the step-size used in numerical integration of Equations (28). Since the guidance is "closed loop" and can be made accurate near cut-off at small cost, this does not cause guidance errors. However, it has three possible adverse and unavoidable effects:

(1) The steering policy becomes non-optimal, incurring a fuel penalty.

(2) The initial estimate of t_f becomes inaccurate, which distorts flight planning.

(3) Truncation errors in the computation of sensitivity coefficients (which are required for iterative correction of the estimated parameters $\underline{\lambda}(t_o)$, $\dot{\underline{\lambda}}(t_o)$ and t_f) cause them to differ from the actual sensitivities of the numerically integrated Equations (28). This degrades the convergence of the iterative solution, and in extreme instances, destroys convergence completely.

However, it is known that present day explicit guidance schemes, which are essentially one-step integrations of the trajectory equations improved by closed formulae for certain thrust integrals, have small performance penalties. It may therefore be expected that an "explicit-predictor" integration scheme which resembles the application of explicit guidance to each integration step will have negligible performance penalty for very large integration steps. This, in fact, is the case. It has been demonstrated that one integration step per stage of the Titan IIIC booster produces a ΔV performance penalty of only 0.1 ft/sec.

Errors in initial estimates of t_f, made when the time remaining until thrust cut-off is large, are quite significant for explicit guidance schemes, but have been found to be acceptably small when the explicit-predictor integration scheme is used with large steps. It has been repeatedly demonstrated, using the Titan IIIC booster, that integration steps of 100 seconds produced an error in predicted cut-off time of less than 0.3 seconds in cases where the time remaining until cut-off was greater than 700 seconds. Even larger time steps are believed useable for mission planning.

The remaining factor governing useable step-size is the effect of truncation error on the convergence of the iterative solution. This effect disappears if the sensitivity coefficients are computed by finite differences. This is done by first generating an "unperturbed" trajectory and evaluating the error quantities. Then "perturbed" trajectories are generated by changing components of the initial adjoint vector one at a time by a small specified variations. Subtracting the error quantities of each perturbed trajectory from the corresponding error quantities of the unperturbed trajectory, and dividing by the change in the initial adjoint component, gives one column of the sensitivity matrix.

Computing sensitivities by finite difference is similar to performing experiments on the numerical trajectory solution. Regardless of truncation errors, approximations in the equations, etc., the sensitivity coefficients generated by finite differences will give an accurate prediction (except for roundoff and nonlinearity effects) of the way in which the numerical trajectory solution will change if its parameters are changed.

Experience with the ascent case has shown that there is no difficulty in choosing parameter changes large enough to avoid trouble from roundoff errors, but small enough for linearity to hold. In trial experiments, the sizes of the parameter increments were varied over a range of more than three orders of magnitude without appreciable degradation of convergence.

3.4 EXPLICIT-PREDICTOR INTEGRATION ALGORITHM

The "explicit-predictor" integration algorithm is designed to take advantage of closed formulae for integrals involving thrust acceleration. The formulae presented here assume that rocket engine mass flow rate $\dot{M}$ and effective exhaust velocity c* are constant. This should not be construed as a limitation of the technique. In theory, at least, any modelable function of burning time can be used for either $\dot{M}$ or c*. For example, a "propellant-utilization" model in which the level of thrust is altered sometime after engine ignition could be implemented within the framework of the algorithm. Also there is no limit imposed on the number of rocket stages.

The equations of the integration algorithm will be shown for the jth rocket stage and the nth time step. Since engine parameters and mass are generally discontinuous at staging, the integration time step is selected so that no more than one stage is ever included in any integral evaluation. In other words

$$\begin{aligned} \Delta t_n &= \text{Min}\ (\Delta t_{max},\ t_j - t_n) \\ t_{n+1} &\equiv t_n + \Delta t_n \end{aligned} \tag{29}$$

where t_j is time at the end of stage j, t_n is present time and Δt_{max} is the largest time step allowed.

The thrust acceleration term $a(t)\ \hat{\underline{u}}$ is separated into two parts, that which defines its magnitude a(t) and that which defines its direction $\hat{\underline{u}}$. Over each integration step, $\hat{\underline{u}}$ is approximated by a second-order vector function of time.

$$\hat{\underline{u}}(t) = \hat{\underline{u}}(t_m) + \underline{\alpha}_n\ (t-t_m) + \underline{\beta}_n\ (t-t_m)^2 \qquad t_n \leq t \leq t_{n+1} \tag{30}$$

where

$$t_m = t_n + \Delta t_n/2$$

As will be shown subsequently, it is possible to predict $\hat{\underline{u}}(t_m)$ and $\hat{\underline{u}}(t_{n+1})$. Given the three values of $\hat{\underline{u}}$, the vector coefficients are

$$\begin{aligned} \underline{\alpha}_n &= \left[\hat{\underline{u}}(t_{n+1}) - \hat{\underline{u}}(t_n)\right]\ /\ \Delta t_n \\ \underline{\beta}_n &= 2\left[\hat{\underline{u}}(t_{n+1}) + \hat{\underline{u}}(t_n) - 2\ \hat{\underline{u}}(t_m)\right]\ /\ \Delta t_n^{\ 2} \end{aligned} \tag{31}$$

To complete the single and double integrals of thrust acceleration from t_n to t_{n+1}, it is necessary to evaluate six integrals:

$$I_1 = \int_{t_n}^{t_{n+1}} \frac{c^*_j}{\tau_j + t_{j-1} - t} \, dt$$

$$I_2 = \int_{t_n}^{t_{n+1}} \frac{c^*_j (t-t_m)}{\tau_j + t_{j-1} - t} \, dt$$

$$I_3 = \int_{t_n}^{t_{n+1}} \frac{c^*_j (t-t_m)^2}{\tau_j + t_{j-1} - t} \, dt$$

$$I_4 = \int_{t_n}^{t_{n+1}} \int_{t_n}^{t} \frac{c^*_j}{\tau_j + t_{j-1} - s} \, ds dt \qquad (32)$$

$$I_5 = \int_{t_n}^{t_{n+1}} \int_{t_n}^{t} \frac{c^*_j (s-t_m)}{\tau_j + t_{j-1} - s} \, ds dt$$

$$I_6 = \int_{t_n}^{t_{n+1}} \int_{t_n}^{t} \frac{c^*_j (s-t_m)^2}{\tau_j + t_{j-1} - s} \, ds dt$$

where

$$\tau_j = -M_{oj}/\dot{M}_j$$

M_{oj} is the mass at the ignition of stage j.

For the assumptions made above, namely that c^*_j and $\dot{M}_j$ are constant

$$
\begin{aligned}
I_1 &= c^*_j \log_e (1 + \rho) \\
I_4 &= c^*_j \; \Delta t_n - (\tau_j + t_{j-1} - t_{n+1}) I_1 \\
I_2 &= -I_4 - (t_m - t_{n+1}) I_1 \\
I_3 &= (\tau_j + t_{j-1} - t_m)(I_2 - c^*_j \; \Delta t_n) \\
&\quad + c^*_j (\tau_j + t_{j-1} - t_{n+1})^2 (2\rho + \rho^2)/2 \qquad (33) \\
I_5 &= (\tau_j + t_{j-1} - t_m) I_4 - c^*_j \; \Delta t_n^{\;2} /2 \\
I_6 &= (\tau_j + t_{j-1} - t_m)(I_5 - c^*_j \; \Delta t_n^{\;2} /2) \\
&\quad - c^*_j \Big[(\tau_j + t_{j-1} - t_{n+1})^3 (3\rho + 3\rho^2 + \rho^3)/6 \\
&\quad - (\tau_j + t_{j-1} - t_n)^2 \; \Delta t_n /2 \Big]
\end{aligned}
$$

where

$$\rho = \Delta t_n / (\tau_j + t_{j-1} - t_{n+1})$$

Having constructed the closed formulae for the first and second integrals of thrust, the total "explicit-predictor" algorithm can now be shown. The contributions to position and velocity due to thrust acceleration, $\Delta \underline{r}'$ and $\Delta \underline{v}'$, and due to gravitational acceleration, $\Delta \underline{r}''$ and $\Delta \underline{v}''$, will be isolated as a hedge against requiring extended precision integral accumulation. The primer vector can now be predicted at t_m and t_{n+1} by using the second order equations:

$$
\begin{aligned}
\underline{\lambda}_m &= \underline{\lambda}_n + \dot{\underline{\lambda}}_n \; \Delta t_n /2 + \ddot{\underline{\lambda}}_n \; \Delta t_n^{\;2} /8 \\
\underline{\hat{u}}_m &= \underline{\lambda}_m / \lambda_m \\
\underline{\lambda}_{n+1} &= \underline{\lambda}_n + \dot{\underline{\lambda}}_n \; \Delta t_n + \ddot{\underline{\lambda}}_n \; \Delta t_n^{\;2} /2 \qquad (34) \\
\underline{\hat{u}}_{n+1} &= \underline{\lambda}_{n+1} / \lambda_{n+1}
\end{aligned}
$$

After evaluating $\underline{\alpha}$ and $\underline{\beta}$ according to Equations (31) and $I_1, \cdots, I_6$ according to Equations (33), the thrust acceleration contributions to position and velocity are:

$$
\begin{aligned}
\Delta \underline{r}'_{n+1} &= \Delta \underline{r}'_n + \Delta \underline{v}_n' \Delta t_n + I_4 \hat{\underline{u}}_m + I_5 \underline{\alpha}_n + I_6 \underline{\beta}_n \\
\Delta \underline{v}'_{n+1} &= \Delta \underline{v}'_n + I_1 \hat{\underline{u}}_m + I_2 \underline{\alpha}_n + I_3 \underline{\beta}_n
\end{aligned}
\qquad (35)
$$

The gravitational acceleration contributions to position are:

$$
\Delta \underline{r}''_{n+1} = \Delta \underline{r}''_n + \Delta \underline{v}''_n \Delta t_n + \underline{g}_n \Delta t_n^{\,2}/2 + G_n \underline{v}_n \Delta t_n^{\,3}/6 \qquad (36)
$$

Total position is

$$
\underline{r}_{n+1} = \underline{r}_o + \underline{v}_o (t_{n+1} - t_o) + \Delta \underline{r}'_{n+1} + \Delta \underline{r}''_{n+1} \qquad (37)
$$

Gravitational acceleration and the gravity gradient matrix can now be evaluated using the predicted value of $\underline{r}_{n+1}$; then $\ddot{\underline{\lambda}}_{n+1}$ can be evaluated

$$
\begin{aligned}
\underline{g}_{n+1} &= -(\mu / r_{n+1}^{\,3}) \, \underline{r}_{n+1} \\
G_{n+1} &= -(\mu / r_{n+1}^{\,3}) \left[I - (3/r_{n+1}^{\,2}) \, \underline{r}_{n+1} \, \underline{r}_{n+1}^{\,T} \right] \\
\ddot{\underline{\lambda}}_{n+1} &= G_{n+1} \, \underline{\lambda}_{n+1}
\end{aligned}
\qquad (38)
$$

Finally, velocity and primer vector rate are

$$
\begin{aligned}
\dot{\underline{\lambda}}_{n+1} &= \dot{\underline{\lambda}}_n + (\ddot{\underline{\lambda}}_n + \ddot{\underline{\lambda}}_{n+1}) \, \Delta t_n / 2 \\
\Delta \underline{v}''_{n+1} &= \Delta \underline{v}''_n + (\underline{g}_{n+1} + \underline{g}_n) \, \Delta t_n / 2 \\
\underline{v}_{n+1} &= \underline{v}_o + \Delta \underline{v}''_{n+1} + \Delta \underline{v}'_{n+1}
\end{aligned}
\qquad (39)
$$

Equations (34) to (39) are the explicit-predictor integration technique. As the problem is integrated, the index j must refer to the "current" stage. It is worth noting that no restrictions are placed on the time at which the problem is begun. In this way the algorithm is applicable to the real-time guidance requirement.

3.5 ITERATIVE DETERMINATION OF ESTIMATED PARAMETERS

As pointed out previously, seven error quantities must be brought to null by iteratively adjusting the estimated parameters $\underline{\lambda}(t_o)$, $\dot{\underline{\lambda}}(t_o)$ and t_f.

Also, since the normalization of the primer vector at t_o is irrelevant, by an expeditious choice of coordinates a "preferred coordinate" can be established along which the component of $\underline{\lambda}$ can be left invariant. What this means is that the "preferred axis" should not be chosen orthogonal or nearly orthogonal to the expected direction of thrust acceleration. In general, this is an easy requirement to meet and is worthwhile since it reduces iteratively corrected parameters from seven to six.

It is now necessary to establish the six-by-six matrix of sensitivities, the so called Jacobian matrix, which relates the variation of estimated parameters to the variation in error quantities. The error quantities used are unique to, and indeed define, the type of trajectory being generated. Error quantities for several trajectory types will be presented in a subsequent section.

The Jacobian matrix will be computed, as was stated previously, by employing finite difference techniques. Only five "perturbed" and one "unperturbed" trajectories must be generated. (At this point, one can see the obvious advantage of an extremely rapid trajectory generation technique.) Each perturbed trajectory yields one column of the Jacobian matrix. The sixth column, which represents the changes induced in the error quantities by a unit charge of the final time, can be evaluated by extrapolating the final state of the unperturbed trajectory over some small time increment and then reevaluating the error quantities.

If the error quantities for the six perturbed trajectories are designated by column vectors $\underline{\epsilon}_1, \ldots, \underline{\epsilon}_6$ of dimension 6, and the unperturbed trajectory error quantities by $\underline{\epsilon}_o$, the six-by-six Jacobian matrix is

$$J = \Big[(\underline{\epsilon}_o - \underline{\epsilon}_1)/\Delta\dot{\lambda} \;, (\underline{\epsilon}_o - \underline{\epsilon}_2)/\Delta\dot{\lambda} \;, (\underline{\epsilon}_o - \underline{\epsilon}_3)/\Delta\dot{\lambda} \;, (\underline{\epsilon}_o - \underline{\epsilon}_4)/\Delta\lambda \;, \\ (\underline{\epsilon}_o - \underline{\epsilon}_5)/\Delta\lambda \;, (\underline{\epsilon}_o - \underline{\epsilon}_6)/\Delta t_f \Big] \tag{40}$$

The Newton - Raphson iteration equation in vector form is

$$\begin{bmatrix} \dot{\lambda}_1 \\ \dot{\lambda}_2 \\ \dot{\lambda}_3 \\ \lambda_2 \\ \lambda_3 \\ t_f \end{bmatrix} \leftarrow \begin{bmatrix} \dot{\lambda}_2 \\ \dot{\lambda}_1 \\ \dot{\lambda}_3 \\ \lambda_2 \\ \lambda_3 \\ t_f \end{bmatrix} + J^{-1} (\Gamma \underline{\epsilon}_o) \tag{41}$$

where the numbered subscripts on λ and $\dot{\lambda}$ refer to the first, second, and third components. Note that λ_1 is arbitrarily chosen to be along the "preferred" coordinate.

In Equation (41), Γ is a convergence progress control parameter which is adjusted according to a "performance indicator". The performance indicator is defined as

$$P = \sum_{i=1}^{6} W_i \, \epsilon_{oi}^{2} \tag{42}$$

which is the weighted sum square of the unperturbed error quantities. The weighting factors W_i, are required because the error quantities are not (in general) of the same dimensions.

It can be argued that if the Jacobian matrix is correct and non singular, progress must be made by Equation (41) in reducing the error quantities if a sufficiently small Γ is used. On the other hand, if Γ is too small, the reduction process will be unnecessarily sluggish. The rule used to adaptively adjust Γ is as follows: if the performance indicator on the present iteration is less than it was on the last iteration, Γ is set

$$\Gamma \leftarrow \text{Min}\ (2\Gamma,\ 1)$$

and the evaluation of a new Jacobian proceeds. If the performance indicator is greater than it was on the last iteration, Γ is set

$$\Gamma \leftarrow \Gamma/2$$

3.6 SPECIFICATION OF ERROR QUANTITIES

As shown previously, it is necessary to specify six error quantities which uniquely define a mission type and, when brought to zero, satisfy the mission objectives. Of these six, k are specified final state conditions (or functions of final state conditions) and 6-k are transversality conditions which insure an optimal trajectory.

Three basic options most used by QRGT will be described below. The other orbit injection options described in the summary of capabilities can be derived in a straightforward manner but are not presented here.

1) Injection into Orbit, with Specification of Orbit Size, Shape, and Orientation.

For injection into orbit with prescribed orbit size, shape and orientation, the only degrees of freedom remaining are the time and location at which orbit insertion is to occur. Let us assume that the desired orbit is specified by its radius and velocity vectors at perigee, $\underline{r}_p$ and $\underline{v}_p$. With this specification, the required velocity vector at any location in the orbit is given by

$$\underline{v}_R = \frac{1}{r_p}\sqrt{\frac{\mu}{s}} \sin\theta \, \underline{r}_p + \left\{1 - \frac{r_p}{s}\left[1 - \cos\theta\right]\right\}\underline{v}_p \tag{45}$$

where θ is the true anomaly. The semilatus rectum, s, is given by

$$s = v_p^2 r_p^2 / \mu \tag{46}$$

The magnitude of the required radius at any point in the orbit is

$$r_R = \frac{s}{(1 + e\cos\theta)} \tag{47}$$

where the orbit eccentricity is given by

$$e = (s/r_p - 1)^{1/2} \tag{48}$$

The unit vector normal to the orbital plane is

$$\hat{\underline{n}} = \widehat{\underline{r}_p \times \underline{v}_p} \tag{49}$$

As shown above, the required velocity vector and required radius are functions of the true anomaly. The true anomaly of the generated trajectory is given by

$$\theta = \text{Tan}^{-1} \left\{ \frac{\underline{r}(t_f) \cdot \underline{v}_p / v_p}{\underline{r}(t_f) \cdot \underline{r}_p / r_p} \right\} \tag{50}$$

Five of the six required error quantities are

$$\begin{aligned} \begin{bmatrix} \epsilon_{n4} \\ \epsilon_{n5} \\ \epsilon_{n6} \end{bmatrix} &= \underline{v}_R\,(\theta) - \underline{v}(t_f) \\ \epsilon_{n2} &= r_R\,(\theta) - r(t_f) \\ \epsilon_{n3} &= -\hat{\underline{n}} \cdot \underline{r}(t_f) \end{aligned} \tag{51}$$

These five quantities are sufficient to satisfy all mission specification. The sixth quantity ϵ_{n1} gives the deviation from satisfaction of a transversality condition, which is

$$\underline{\lambda}\ (t_f) \cdot \underline{g}\,(t_f) - \dot{\underline{\lambda}}(t_f) \cdot \underline{v}(t_f) = 0 \tag{52}$$

and expresses the fact that phase-in-orbit is not specified. This is the well-known transversality condition for "time-free" cases.

2) Direct Ascent to Rendezvous, with Rendezvous Time Specified

Direct-ascent-to-rendezvous means that, after thrust is initially terminated, only a velocity matching burn at the specified rendezvous radius is required. For this case, the required velocity, $\underline{v}_R$ is determined by means of Lambert's theorem. One statement of Lambert's theorem is that, if two position vectors, the time required to traverse between them, and whether the true anomaly to be traversed is greater than or less than 180° are specified, the orbit of traverse is unique and deterministic. The equations used to determine

$\underline{v}_R$ can be found in Reference 2 and are not repeated here. The remaining three error quantities must be specified by transversality conditions.

To determine the applicable transversality conditions let us first examine the equation of perturbed motions obtained from Equation (28a).

$$\delta \ddot{\underline{r}} = \delta \dot{\underline{v}} = G\,\delta \underline{r} + \delta \underline{a} \tag{53}$$

To obtain Equation (53), it is assumed that $|\delta \underline{r}|$ is small enough to justify ignoring terms of the order of $|\delta \underline{r}|^2$.

Recalling Equation (28b), we see that on a coast trajectory, where $\delta a = 0$, Equation (28b) is identical in form to Equation (53). Hence, the well known state transition matrix solution for Equation (53) must also be a solution for Equation (28b). Symbolically,

$$\begin{bmatrix} \underline{\lambda}(t_r) \\ \dot{\underline{\lambda}}(t_r) \end{bmatrix} = \Phi\,(t_r, t_f) \begin{bmatrix} \underline{\lambda}(t_f) \\ \dot{\underline{\lambda}}(t_f) \end{bmatrix} \tag{54}$$

where t_r is the time of rendezvous. If $\dot{\underline{\lambda}}(t_r)$ is assumed to be zero (i.e., a fixed attitude orbit matching burn) the required primer vector rate at thrust cut-off is given by

$$\dot{\underline{\lambda}}_R(t_f) = \Phi_{12}^{-1} \left\{ \underline{\lambda}(t_r) - \Phi_{11}\,\underline{\lambda}(t_f) \right\} \tag{55}$$

where

$$\Phi\,(t_r, t_f) = \left[\begin{array}{c|c} \Phi\,11 & \Phi\,12 \\ \hline \Phi\,21 & \Phi\,22 \end{array}\right]$$

All that remains is to determine $\underline{\lambda}(t_r)$. Recalling Equation (21), we see that for an optimal trajectory the magnitudes of the primer vector at the first thrust termination and at the beginning of the velocity matching burn must be equal. The direction of the primer vector at the beginning of the velocity matching burn must be approximately in the direction of the velocity-to-be-gained vector. Symbolically,

$$\underline{\lambda}(t_r) = \lambda(t_f) \left\lfloor \widehat{\underline{v}_T(t_r) - \underline{v}(t_r)} \right\rfloor \tag{56}$$

where $\underline{v}_T(t_r)$ is the target velocity and $\underline{v}(t_r)$ is the interceptors velocity at rendezvous. Hence, the three additional error quantities are

$$\begin{bmatrix} \epsilon_{n1} \\ \epsilon_{n2} \\ \epsilon_{n3} \end{bmatrix} = \Phi_{12}^{-1} \left\{ \lambda(t_f) \left[\overline{\underline{v}_T(t_r) - \underline{v}(t_r)} \right] - \Phi_{11} \underline{\lambda}(t_f) \right\} - \dot{\underline{\lambda}}(t_f) \qquad (57)$$

Reference 3 describes a very simple and elegant way of computing the required state transition matrix which is valid for all orbits except the degenerate rectilinear orbit.

3) Direct Ascent to Intercept--Intercept Time Specified

Direct-ascent-to-intercept is very similar to direct-ascent-to-rendezvous just described. The principal difference is that no velocity matching burn is performed at intercept. The required velocity for intercept $\underline{v}_R$, is computed in the same manner as for rendezvous but the transversality conditions are not.

Recalling Equation (27), we have

$$\dot{\underline{\lambda}} \cdot \underline{\delta r} = \underline{\lambda} \cdot \underline{\delta v} \qquad t \geq t_f \qquad (58)$$

This equation must hold at all points on the coast arc. If we choose the point at intercept, the perturbation in position, $\delta\underline{r}$, must, by definition, be zero. Since there is no constraint on velocity at intercept, the only way Equation (58) can be satisfied is if $\underline{\lambda}$ at intercept is also zero.

Equation (57) can now be rewritten for the intercept case and is

$$\begin{bmatrix} \epsilon_{n1} \\ \epsilon_{n2} \\ \epsilon_{n3} \end{bmatrix} = - \Phi_{12}^{-1} \Phi_{11} \underline{\lambda}(t_f) - \dot{\underline{\lambda}}(t_f) \qquad (59)$$

3.7 USE OF OP-EX FOR ORBITAL MANEUVERS

OP-EX has several attributes which make it ideally suitable for guiding orbit maneuvers as well as ascent. Principal among these is its ability to optimally perform large plane changes. In fact, the direct-ascent-to-rendezvous option can be directly applied (except for minor changes in initialization) to all intermediate burns and their subsequent coast arcs. As has been shown previously, additional guidance options can be easily added without sacrificing the optimal performance of OP-EX.

In keeping with the QRGT mission planning policy, the time of initiation of each thrusting maneuver and the location and time of the subsequent thrusting maneuver are determined by the mission planner. Since the planner assumes impulsive changes in velocity, the actual time of engine ignition during active guidance will be modified to make the nominal centroid time of the burn correspond with the impulse time specified by the planner.

3.8 EFFECT ON OP-EX OF MEASUREMENT OF ENGINE PERFORMANCE DURING ACTIVE GUIDANCE

OP-EX guidance relies on propulsion performance descriptions for the various stages to extrapolate the trajectory forward to thrust termination. In active guidance, variation of engine performance from that specified will cause the extrapolated trajectory to be in error. However, because the guidance is "closed-loop" by design, (that is, the trajectory is recursively extrapolated every ten seconds during early stages and more frequently in the final stage) such performance variations do not seriously effect the satisfaction of guidance objectives, but the fuel optimizing ability of OP-EX is handicapped (although certainly not disastrously) by the variations.

A "mass-rate" prediction scheme, which will be derived subsequently, was implemented for study purposes and was found to enhance the fuel efficiency of OP-EX guidance in the presence off-nominal engine performance. The amount of ΔV saved, in every case, was a few ft/sec.

The "mass-rate" predictor is formulated as follows: the theoretical ΔV gained in stage j between the k and (k+1) predictions of OP-EX is (if c^* is constant)

$$\Delta V = c^*_j \log_e \left\{ \frac{M_{oj} + \tilde{\dot{M}}_j \left[t_{k+1} - t_{j-1} \right]}{M_{oj} + \tilde{\dot{M}}_j \left[t_k - t_{j-1} \right]} \right\} \qquad (60)$$

where M_{oj} is initial mass, c^*_j is effective exhaust velocity, t_{j-1} is time of ignition (a measured quantity) and $\tilde{M}_j$ is estimated mass rate of the jth stage. The measured $\Delta V'$ gained is approximately

$$\Delta V' \approx \hat{\underline{u}}(t_k) \cdot \left[w(t_{k+1}) - w(t_k) \right] \tag{61}$$

where $\hat{\underline{u}}(t)$ is the direction of thrust acceleration and w(t) is the velocity-meter output vector at time t. Predicted average mass rate is then

$$\tilde{M}_j = \frac{- M_{oj} \left[1 - \exp \left\{ - \Delta V'/c^*_j \right\} \right]}{\left[t_{k+1} - t_{j-1} \right] - \left[t_k - t_{j-1} \right] \exp \left\{ - \Delta V'/c^*_j \right\}} \tag{62}$$

It can be seen from Equation (62) that errors in initial mass are absorbed in the estimated mass rate.

If every last ounce of fuel is to be saved, perhaps a multistate filter of the Kallman type could be used to estimate M_{oj}, $\dot{M}_j$ and c^*_j.

3.9 GROWTH CAPABILITY FOR TREATMENT OF CONSTRAINTS

For some vehicles and missions, the flight guidance must satisfy constraints other than the usual constraints on final position and velocity. Examples are (1) recovery-ceiling constraints and (2) requirements for a prescribed attitude at cutoff, together with a low angular rate. The structure of the OP-EX guidance algorithm facilitates the introduction of modifications to make it capable of handling such constraints. The principal changes required are (1) modification of the form of the control policy, in a way that introduces new trajectory parameters, and (2) providing equations for computing new error quantities which measure constraint violation. The explicit-predictor algorithm of OP-EX can be used, without change, to generate trial trajectories by which the old and new error quantities can be evaluated for any given values of the old and new trajectory parameters.

Before active guidance begins, the on-board flight planner (using part of OP-EX as a subroutine) will generate a feasible, optimized nominal trajectory and so provide an initial set of values for the trajectory parameters. Some of these trajectory parameters will remain constant during active guidance, others will be iteratively adjusted by OP-EX. For this purpose, the matrix of sensitivity coefficients must be enlarged by additional columns corresponding to new error quantities, and additional rows for new trajectory parameters. Since the matrix elements are evaluated by finite-difference methods, this enlargement is straightforward.

Appropriate forms for the control-policy modifications can be derived from optimal control theory. It is not necessary to implement the true optimal solutions (which would require, in general, the computation of new adjoint variables) since it is generally not difficult to find simple approximations which give satisfactory performance.

For example, for the recovery-ceiling constraint, a preliminary analysis indicates that a satisfactory scheme may be realizable as follows: the control policy is modified by introducing discontinuous changes of $\underline{\lambda}$ and $\dot{\underline{\lambda}}$ at a scheduled future time t_s which is prior to the predicted cutoff but after the latest time at which the recovery-ceiling constraint might be active. These changes are of the form

$$\begin{aligned} \Delta\underline{\lambda} &= k\underline{b}_1 \\ \Delta\dot{\underline{\lambda}} &= k\underline{b}_2 \end{aligned} \qquad (63)$$

where k is a parameter to be iteratively adjusted by OP-EX. The vectors $\underline{b}_i$ and $\underline{b}_2$ are generated by the flight planner, together with the time t_s, and are not changed during active guidance. When the danger of violating the recovery-ceiling has passed (which will occur before time t_s) the parameter k is set to zero, restoring the normal control policy. At times immediately after this switch-over, the rate of change of commanded thrust direction must be limited to prevent undesirable transient behavior.

The problem of satisfying attitude constraints at cutoff involves both guidance and control. Optimal control theory indicates that the optimal trajectories consist of two arcs. On the first arc, which covers most of the trajectory, the present OP-EX control policy is optimal, and constraints on angular rates or angular accelerations have no direct influence. On the second arc, the attitude control system is working at maximum effort to achieve the desired final attitude and attitude rate. In an actual implementation, the maximum-effort phase must be followed by a precision-control phase that tries to realize accurate end conditions. A reasonable approach to the over-all problem is to first consider the attitude-change maneuver as a separate optimization problem (intimately related to the autopilot design) and devise a quasi-optimal policy (with margins for performance uncertainties) for achieving a given final orientation with a low final angular rate. This gives a formula for the maneuver-time required, as a function of angle-to-be-turned-through, which is valid when the initial angular rate is low. This formula (and approximate forms for the angular rate profile during the maneuver) can be used in the explicit-predictive algorithm for generating predicted trajectories. When the attitude-change maneuver begins, control of cutoff position will probably have to be relaxed. However, accurate control of both attitude and velocity is possible by a policy which rapidly nulls the components of velocity-to-be-gained which are normal to the desired final roll-axis direction, and controls the parallel component by cutoff time.

3.10 SIMULATION RESULTS

The OP-EX guidance algorithm has been tested by simulation, using the IBM-developed GISMO simulation program, with the following guidance options:

1) Direct ascent to intercept; initial ascent plane coincident with target plane

2) Direct ascent to intercept; target plane rotated 40° from initial ascent plane

3) Direct ascent to rendezvous; initial ascent plane coincident with target plane

4) Direct ascent to rendezvous; initial ascent plane rotated 40° from target plane

5) Insertion into circular orbit; no dog-leg during ascent

6) Insertion into circular orbit; 40° dog-leg during ascent

7) Insertion into noncircular orbit (eccentricity 0.047) with no dog-leg during ascent

8) Insertion into noncircular orbit (eccentricity 0.047) with 40° dog-leg during ascent

9) Direct rendezvous from orbit; 40° plane change

10) Orbit insertion from orbit; 40° plane change.

In all cases, the Titan IIIC booster was used. Three payload options were used for the various missions. A spherical Earth Model in GISMO was used for all tests since gravity oblatness correction equations were not complete at the time of testing.

In every case the guidance algorithm performed with aplomb. No evidence of non convergence was indicated. The number of iterations for initial convergence ranged from one for option 4) to nine for option 1). This difference is probably due to the fact that all ascent cases were initialized with the same standard set of initial parameter values, and this set is closer to the requirements for injection into a circular orbit than to those for intercept. For all orbit-to-orbit burns, the primer vector is initially aligned along the estimated velocity-to-be-gained vector furnished by the mission planner. It is possible that this rule (or an adaptation of it) should be followed for direct-ascent intercept cases also.

In all intercept or rendezvous cases, the error in position at the specified time of intercept was less than 20 feet. In all orbit insertions, the error in radial and out of plane position components was less than 20 feet, and the error in velocity was less than 0.05 ft/sec. These errors were, of course, for nominal engine performance and probably represent, to a high degree, the numerical resolution of the 7094 computer.

3.11 OP-EX MATH FLOW DIAGRAMS

Math flow diagrams of OP-EX guidance are presented in Figure III-1. Table III-1 gives math flow symbol definitions.

OP-EX GUIDANCE ALGORITHM

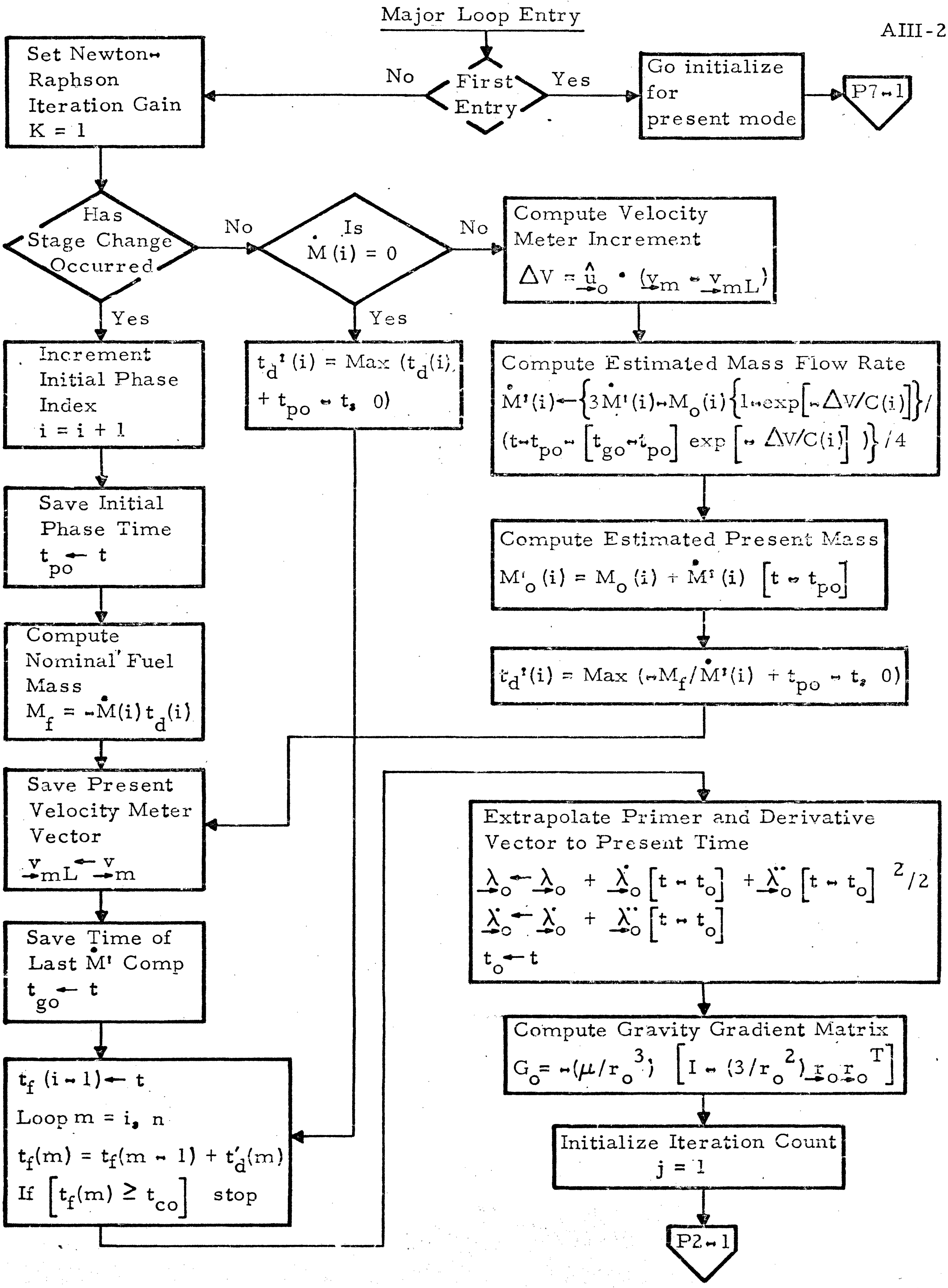

Major Loop Entry
First Entry
No
Yes
Set Newton-Raphson Iteration Gain K = 1
Go initialize for present mode
P7-1
Has Stage Change Occurred
No
Is $\dot{M}(i) = 0$
No
Compute Velocity Meter Increment
$\Delta V = \hat{\underline{u}}_o \cdot (\underline{v}_m - \underline{v}_{mL})$
Yes
Increment Initial Phase Index
i = i + 1
Yes
$t_d'(i) = \text{Max}\ (t_d(i) + t_{po} - t, 0)$
Compute Estimated Mass Flow Rate
$\dot{M}'(i) \leftarrow \{3\dot{M}'(i) - M_o(i)\{1 - \exp[-\Delta V/C(i)]\}/(t - t_{po} - [t_{go} - t_{po}] \exp[-\Delta V/C(i)])\}/4$
Save Initial Phase Time
$t_{po} \leftarrow t$
Compute Estimated Present Mass
$M'_o(i) = M_o(i) + \dot{M}'(i)\ [t - t_{po}]$
Compute Nominal Fuel Mass
$M_f = -\dot{M}(i) t_d(i)$
$t_d'(i) = \text{Max}\ (-M_f/\dot{M}'(i) + t_{po} - t, 0)$
Save Present Velocity Meter Vector
$\underline{v}_{mL} \leftarrow \underline{v}_m$
Extrapolate Primer and Derivative Vector to Present Time
$\underline{\lambda}_o \leftarrow \underline{\lambda}_o + \dot{\underline{\lambda}}_o [t - t_o] + \ddot{\underline{\lambda}}_o [t - t_o]^2/2$
$\dot{\underline{\lambda}}_o \leftarrow \dot{\underline{\lambda}}_o + \ddot{\underline{\lambda}}_o [t - t_o]$
$t_o \leftarrow t$
Save Time of Last $\dot{M}'$ Comp
$t_{go} \leftarrow t$
Compute Gravity Gradient Matrix
$G_o = -(\mu/r_o^3)\ [I - (3/r_o^2)\underline{r}_o \underline{r}_o^T]$
$t_f(i-1) \leftarrow t$
Loop m = i, n
$t_f(m) = t_f(m-1) + t'_d(m)$
If $[t_f(m) \geq t_{co}]$ stop
Initialize Iteration Count
j = 1
P2-1

Figure III-1 OP-EX Guidance Algorithm Sheet 1 of 9

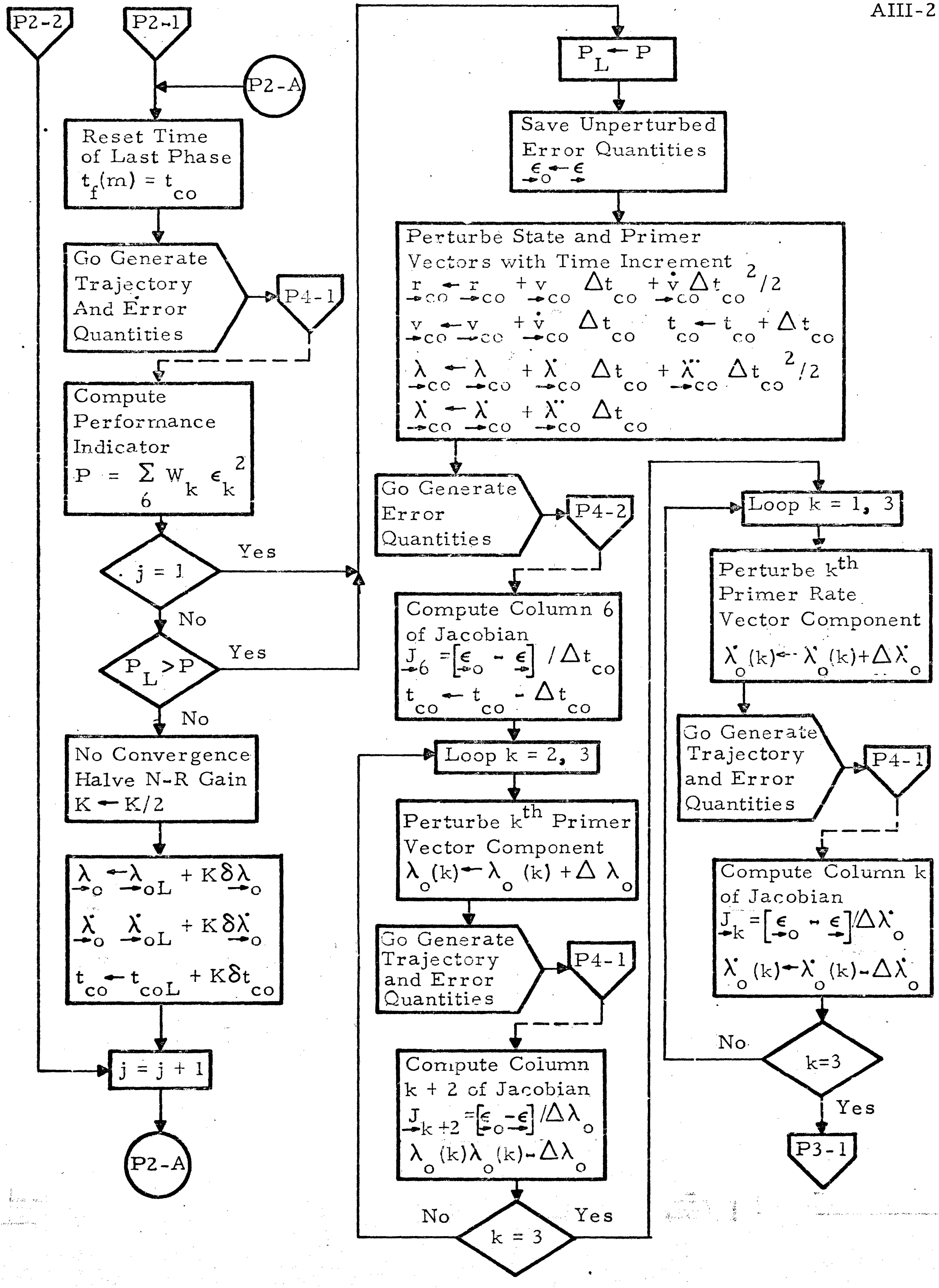

P2-2
P2-1
P2-A
Reset Time of Last Phase $t_f(m) = t_{co}$
Go Generate Trajectory And Error Quantities
P4-1
Compute Performance Indicator $P = \sum_6 W_k \epsilon_k^2$
j = 1
Yes
No
$P_L > P$
Yes
No
No Convergence Halve N-R Gain $K \leftarrow K/2$
$\vec{\lambda}_o \leftarrow \vec{\lambda}_{oL} + K\delta\vec{\lambda}_o$
$\dot{\vec{\lambda}}_o \quad \dot{\vec{\lambda}}_{oL} + K\delta\dot{\vec{\lambda}}_o$
$t_{co} \leftarrow t_{coL} + K\delta t_{co}$
j = j + 1
P2-A
$P_L \leftarrow P$
Save Unperturbed Error Quantities $\vec{\epsilon}_o \leftarrow \vec{\epsilon}$
Perturbe State and Primer Vectors with Time Increment
$\vec{r}_{co} \leftarrow \vec{r}_{co} + \vec{v}_{co}\,\Delta t_{co} + \dot{\vec{v}}_{co}\,\Delta t_{co}^2/2$
$\vec{v}_{co} \leftarrow \vec{v}_{co} + \dot{\vec{v}}_{co}\,\Delta t_{co} \qquad t_{co} \leftarrow t_{co} + \Delta t_{co}$
$\vec{\lambda}_{co} \leftarrow \vec{\lambda}_{co} + \dot{\vec{\lambda}}_{co}\,\Delta t_{co} + \ddot{\vec{\lambda}}_{co}\,\Delta t_{co}^2/2$
$\dot{\vec{\lambda}}_{co} \leftarrow \dot{\vec{\lambda}}_{co} + \ddot{\vec{\lambda}}_{co}\,\Delta t_{co}$
Go Generate Error Quantities
P4-2
Compute Column 6 of Jacobian
$\vec{J}_6 = [\vec{\epsilon}_o - \vec{\epsilon}] / \Delta t_{co}$
$t_{co} \leftarrow t_{co} - \Delta t_{co}$
Loop k = 2, 3
Perturbe k th Primer Vector Component $\lambda_o(k) \leftarrow \lambda_o(k) + \Delta\lambda_o$
Go Generate Trajectory and Error Quantities
P4-1
Compute Column k + 2 of Jacobian
$\vec{J}_{k+2} = [\vec{\epsilon}_o - \vec{\epsilon}] / \Delta\lambda_o$
$\lambda_o(k)\lambda_o(k) - \Delta\lambda_o$
No
k = 3
Yes
Loop k = 1, 3
Perturbe k th Primer Rate Vector Component
$\dot{\lambda}_o(k) \leftarrow \dot{\lambda}_o(k) + \Delta\dot{\lambda}_o$
Go Generate Trajectory and Error Quantities
P4-1
Compute Column k of Jacobian
$\vec{J}_k = [\vec{\epsilon}_o - \vec{\epsilon}] / \Delta\dot{\lambda}_o$
$\dot{\lambda}_o(k) \leftarrow \dot{\lambda}_o(k) - \Delta\dot{\lambda}_o$
No
k=3
Yes
P3-1

Figure III-1 OP-EX Guidance Algorithm Sheet 2 of 9

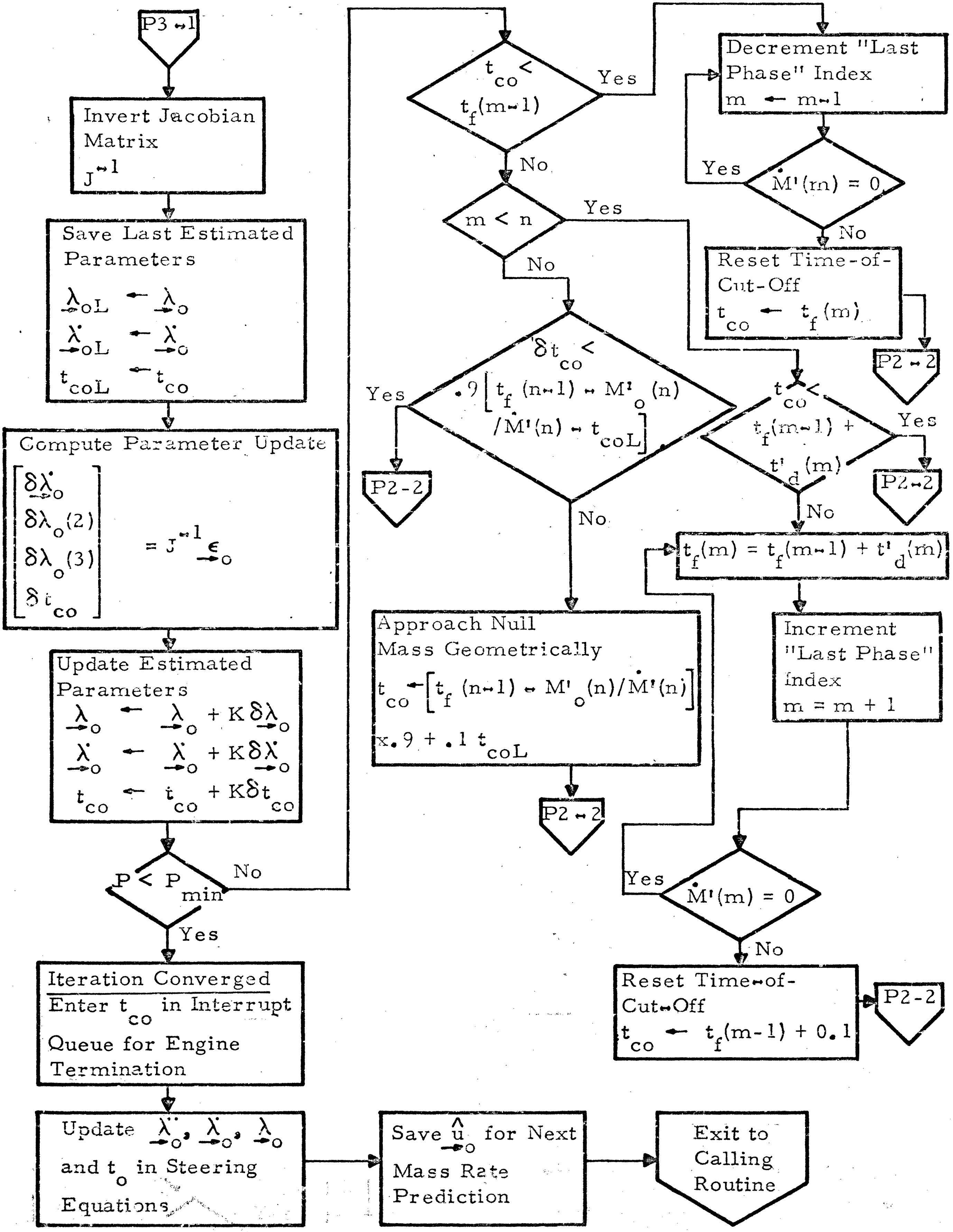

P3 - 1
Invert Jacobian Matrix J^{-1}
Save Last Estimated Parameters
$\underset{\rightarrow}{\lambda}_{oL} \leftarrow \underset{\rightarrow}{\lambda}_o$
$\underset{\rightarrow}{\dot{\lambda}}_{oL} \leftarrow \underset{\rightarrow}{\dot{\lambda}}_o$
$t_{coL} \leftarrow t_{co}$
Compute Parameter Update
$\begin{bmatrix} \delta\underset{\rightarrow}{\dot{\lambda}}_o \\ \delta\lambda_o(2) \\ \delta\lambda_o(3) \\ \delta t_{co} \end{bmatrix} = J^{-1} \underset{\rightarrow}{\epsilon}_o$
Update Estimated Parameters
$\underset{\rightarrow}{\lambda}_o \leftarrow \underset{\rightarrow}{\lambda}_o + K\delta\underset{\rightarrow}{\lambda}_o$
$\underset{\rightarrow}{\dot{\lambda}}_o \leftarrow \underset{\rightarrow}{\dot{\lambda}}_o + K\delta\underset{\rightarrow}{\dot{\lambda}}_o$
$t_{co} \leftarrow t_{co} + K\delta t_{co}$
$P < P_{min}$
No
Yes
Iteration Converged
Enter t_{co} in Interrupt Queue for Engine Termination
Update $\underset{\rightarrow}{\ddot{\lambda}}_o$, $\underset{\rightarrow}{\dot{\lambda}}_o$, $\underset{\rightarrow}{\lambda}_o$ and t_o in Steering Equations
Save $\underset{\rightarrow}{\hat{u}}_o$ for Next Mass Rate Prediction
Exit to Calling Routine
$t_{co} < t_f(m-1)$
Yes
No
$m < n$
Yes
No
$\delta t_{co} < .9\left[t_f(n-1) - M'_o(n)/\dot{M}'(n) - t_{coL}\right]$
Yes
P2-2
No
Approach Null Mass Geometrically
$t_{co} \leftarrow \left[t_f(n-1) - M'_o(n)/\dot{M}'(n)\right] x.9 + .1\, t_{coL}$
P2 - 2
Decrement "Last Phase" Index
$m \leftarrow m-1$
Yes
$\dot{M}'(m) = 0$
No
Reset Time-of-Cut-Off
$t_{co} \leftarrow t_f(m)$
P2 - 2
$t_{co} < t_f(m-1) + t'_d(m)$
Yes
P2-2
No
$t_f(m) = t_f(m-1) + t'_d(m)$
Increment "Last Phase" Index
$m = m + 1$
Yes
$\dot{M}'(m) = 0$
No
Reset Time-of-Cut-Off
$t_{co} \leftarrow t_f(m-1) + 0.1$
P2-2

Figure III-1 OP-EX Guidance Algorithm Sheet 3 of 9

Trial Trajectory Generation Executive Subroutine

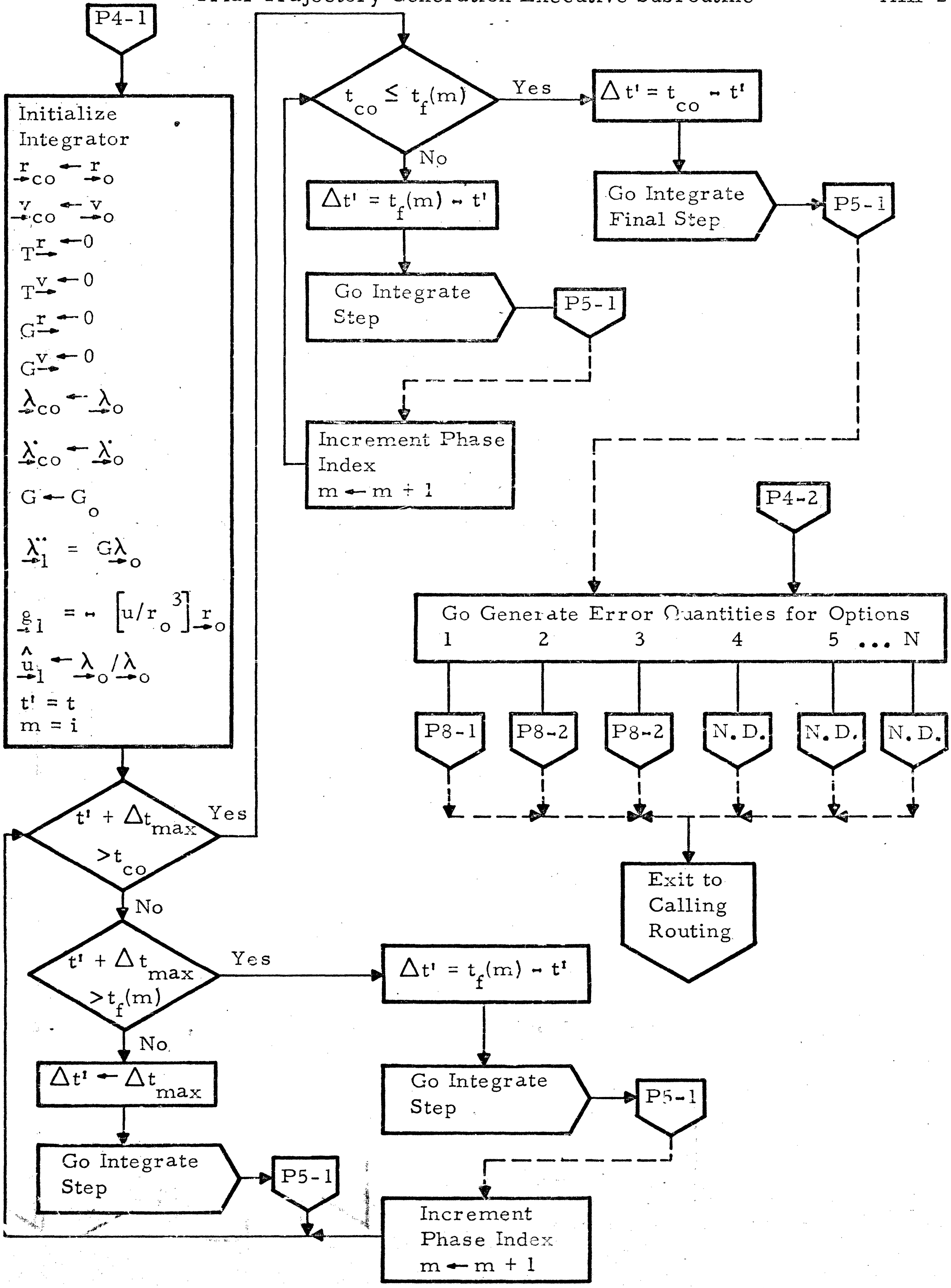
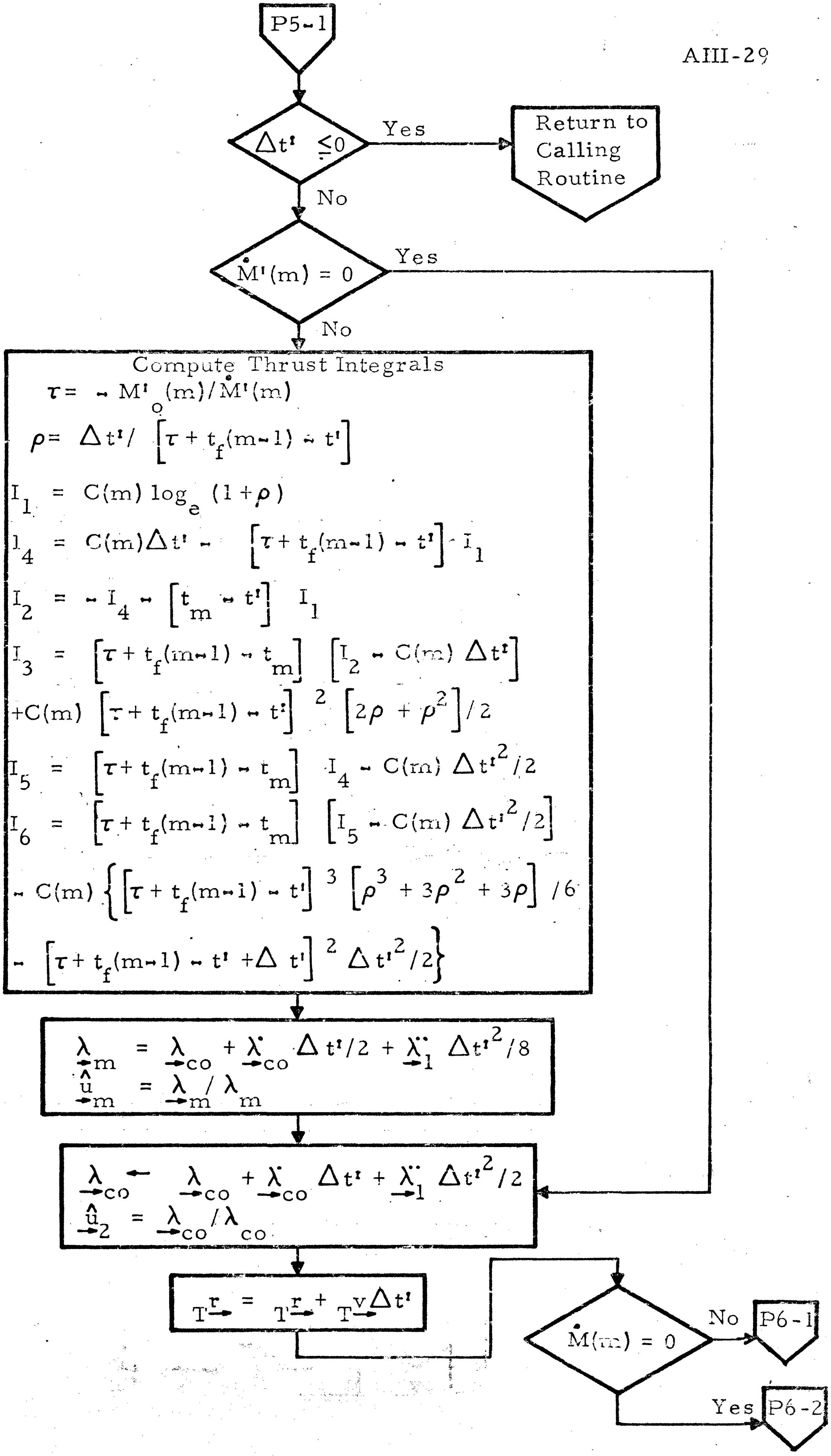

P4-1
Initialize Integrator
$\underset{\rightarrow}{r}_{co} \leftarrow \underset{\rightarrow}{r}_{o}$
$\underset{\rightarrow}{v}_{co} \leftarrow \underset{\rightarrow}{v}_{o}$
$T^{\underset{\rightarrow}{r}} \leftarrow 0$
$T^{\underset{\rightarrow}{v}} \leftarrow 0$
$G^{\underset{\rightarrow}{r}} \leftarrow 0$
$G^{\underset{\rightarrow}{v}} \leftarrow 0$
$\underset{\rightarrow}{\lambda}_{co} \leftarrow \underset{\rightarrow}{\lambda}_{o}$
$\dot{\underset{\rightarrow}{\lambda}}_{co} \leftarrow \dot{\underset{\rightarrow}{\lambda}}_{o}$
$G \leftarrow G_{o}$
$\ddot{\underset{\rightarrow}{\lambda}}_{1} = G\underset{\rightarrow}{\lambda}_{o}$
$\underset{\rightarrow}{g}_{1} = - \left[u/r_{o}^{3}\right] \underset{\rightarrow}{r}_{o}$
$\hat{\underset{\rightarrow}{u}}_{1} \leftarrow \underset{\rightarrow}{\lambda}_{o}/\lambda_{o}$
$t' = t$
$m = i$
$t' + \Delta t_{max} > t_{co}$
Yes
No
$t' + \Delta t_{max} > t_{f}(m)$
Yes
No
$\Delta t' \leftarrow \Delta t_{max}$
Go Integrate Step
P5-1
$\Delta t' = t_{f}(m) - t'$
Go Integrate Step
P5-1
Increment Phase Index
$m \leftarrow m + 1$
$t_{co} \leq t_{f}(m)$
Yes
No
$\Delta t' = t_{co} - t'$
Go Integrate Final Step
P5-1
$\Delta t' = t_{f}(m) - t'$
Go Integrate Step
P5-1
Increment Phase Index
$m \leftarrow m + 1$
P4-2
Go Generate Error Quantities for Options
1
2
3
4
5 ... N
P8-1
P8-2
P8-2
N. D.
N. D.
N. D.
Exit to Calling Routing

Figure III-1 OP-EX Guidance Algorithm Sheet 4 of 9



$$\tau = - M'_o(m)/\dot{M}'(m)$$

$$\rho = \Delta t' / \left[\tau + t_f(m-1) - t'\right]$$

$$I_1 = C(m) \log_e (1+\rho)$$

$$I_4 = C(m)\Delta t' - \left[\tau + t_f(m-1) - t'\right] I_1$$

$$I_2 = - I_4 - \left[t_m - t'\right] I_1$$

$$I_3 = \left[\tau + t_f(m-1) - t_m\right] \left[I_2 - C(m) \Delta t'\right]$$
$$+C(m) \left[\tau + t_f(m-1) - t'\right]^2 \left[2\rho + \rho^2\right]/2$$

$$I_5 = \left[\tau + t_f(m-1) - t_m\right] I_4 - C(m) \Delta t'^2/2$$

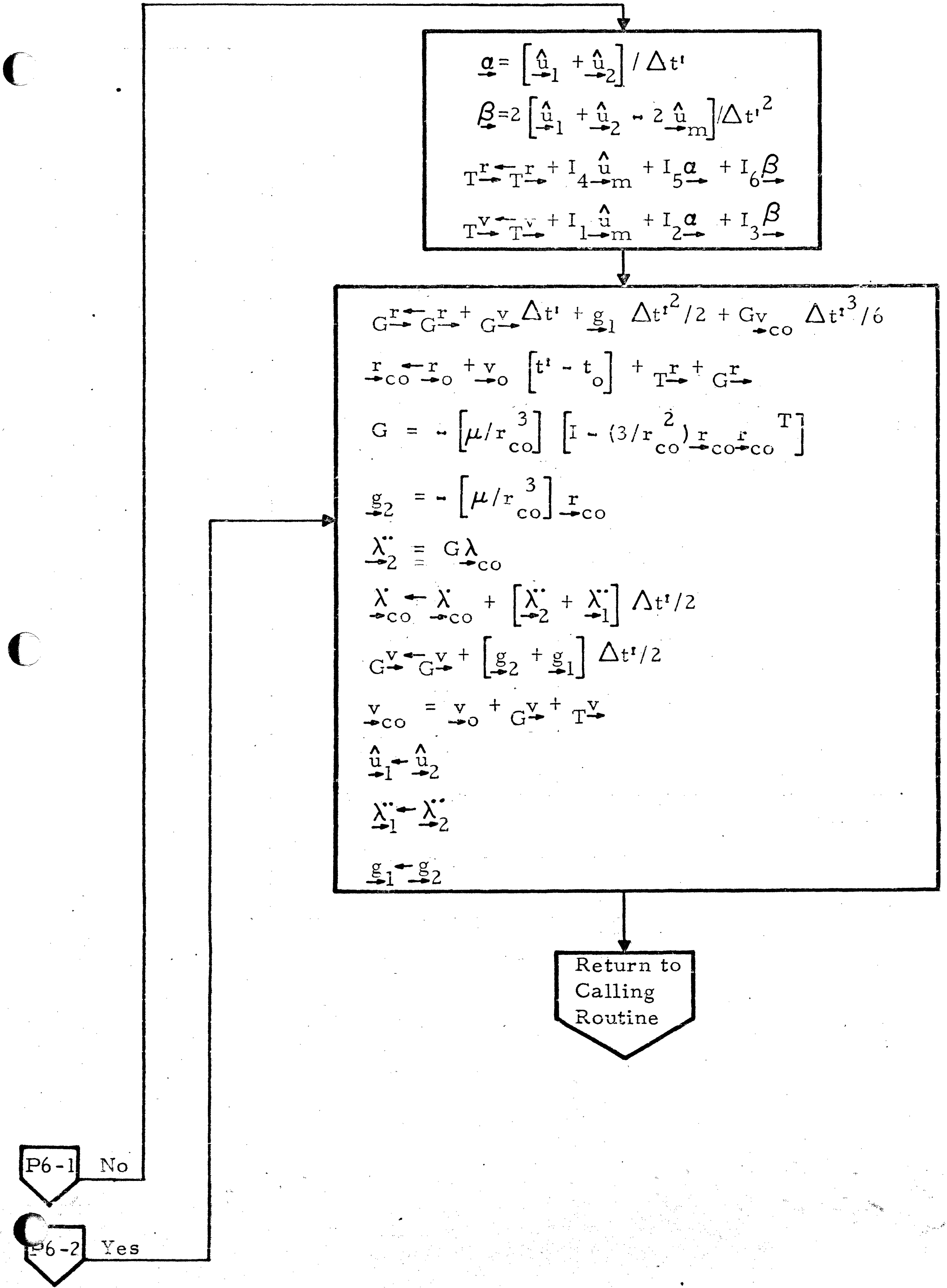
$$I_6 = \left[\tau + t_f(m-1) - t_m\right] \left[I_5 - C(m) \Delta t'^2/2\right]$$
$$- C(m) \left\{\left[\tau + t_f(m-1) - t'\right]^3 \left[\rho^3 + 3\rho^2 + 3\rho\right] /6\right.$$
$$\left. - \left[\tau + t_f(m-1) - t' + \Delta t'\right]^2 \Delta t'^2/2\right\}$$

$$\underrightarrow{\lambda}_m = \underrightarrow{\lambda}_{co} + \underrightarrow{\dot{\lambda}}_{co} \Delta t'/2 + \underrightarrow{\ddot{\lambda}}_1 \Delta t'^2/8$$
$$\underrightarrow{\hat{u}}_m = \underrightarrow{\lambda}_m / \lambda_m$$

$$\underrightarrow{\lambda}_{co} \leftarrow \underrightarrow{\lambda}_{co} + \underrightarrow{\dot{\lambda}}_{co} \Delta t' + \underrightarrow{\ddot{\lambda}}_1 \Delta t'^2/2$$
$$\underrightarrow{\hat{u}}_2 = \underrightarrow{\lambda}_{co} / \lambda_{co}$$

$$T\underrightarrow{r} = T\underrightarrow{r} + T\underrightarrow{v}\Delta t'$$

Figure III-1 OP-EX Guidance Algorithm Sheet 5 of 9



$$\underrightarrow{\alpha} = \left[\underrightarrow{\hat{u}}_1 + \underrightarrow{\hat{u}}_2\right] / \Delta t'$$

$$\underrightarrow{\beta} = 2\left[\underrightarrow{\hat{u}}_1 + \underrightarrow{\hat{u}}_2 - 2\,\underrightarrow{\hat{u}}_m\right] / \Delta t'^2$$

$$T\underrightarrow{r} \leftarrow T\underrightarrow{r} + I_4 \underrightarrow{\hat{u}}_m + I_5 \underrightarrow{\alpha} + I_6 \underrightarrow{\beta}$$

$$T\underrightarrow{v} \leftarrow T\underrightarrow{v} + I_1 \underrightarrow{\hat{u}}_m + I_2 \underrightarrow{\alpha} + I_3 \underrightarrow{\beta}$$

$$G\underrightarrow{r} \leftarrow G\underrightarrow{r} + G\underrightarrow{v}\,\Delta t' + \underrightarrow{g}_1\,\Delta t'^2/2 + G\underrightarrow{v}_{co}\,\Delta t'^3/6$$

$$\underrightarrow{r}_{co} \leftarrow \underrightarrow{r}_o + \underrightarrow{v}_o\left[t' - t_o\right] + T\underrightarrow{r} + G\underrightarrow{r}$$

$$G = -\left[\mu / r_{co}^3\right]\left[I - (3/r_{co}^2)\,\underrightarrow{r}_{co}\,\underrightarrow{r}_{co}^{T}\right]$$

$$\underrightarrow{g}_2 = -\left[\mu / r_{co}^3\right]\underrightarrow{r}_{co}$$

$$\underrightarrow{\ddot{\lambda}}_2 = G\,\underrightarrow{\lambda}_{co}$$

$$\underrightarrow{\dot{\lambda}}_{co} \leftarrow \underrightarrow{\dot{\lambda}}_{co} + \left[\underrightarrow{\ddot{\lambda}}_2 + \underrightarrow{\ddot{\lambda}}_1\right]\Delta t'/2$$

$$G\underrightarrow{v} \leftarrow G\underrightarrow{v} + \left[\underrightarrow{g}_2 + \underrightarrow{g}_1\right]\Delta t'/2$$

$$\underrightarrow{v}_{co} = \underrightarrow{v}_o + G\underrightarrow{v} + T\underrightarrow{v}$$

$$\underrightarrow{\hat{u}}_1 \leftarrow \underrightarrow{\hat{u}}_2$$

$$\underrightarrow{\ddot{\lambda}}_1 \leftarrow \underrightarrow{\ddot{\lambda}}_2$$

$$\underrightarrow{g}_1 \leftarrow \underrightarrow{g}_2$$

Figure III-1 OP-EX Guidance Algorithm Sheet 6 of 9

Guidance Initializer

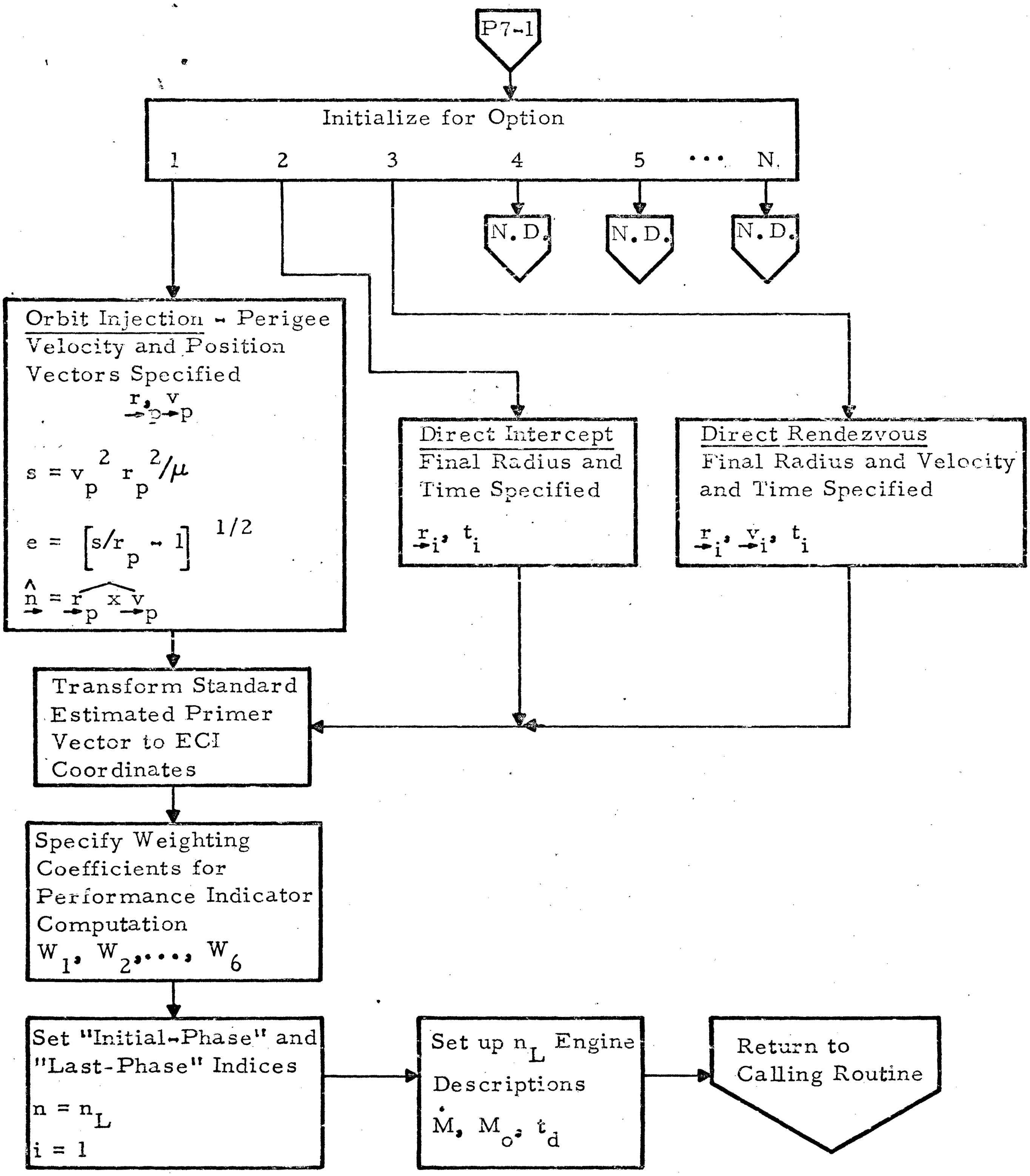

P7-1
Initialize for Option
1
2
3
4
5
• • •
N.
N.D.
N.D.
N.D.
Orbit Injection - Perigee Velocity and Position Vectors Specified
$\underset{\rightarrow}{r}_p, \underset{\rightarrow}{v}_p$
$s = v_p{}^2 \; r_p{}^2/\mu$
$e = \left[s/r_p - 1\right]^{1/2}$
$\underset{\rightarrow}{\hat{n}} = \widehat{\underset{\rightarrow}{r}_p \times \underset{\rightarrow}{v}_p}$
Direct Intercept
Final Radius and Time Specified
$\underset{\rightarrow}{r}_i, t_i$
Direct Rendezvous
Final Radius and Velocity and Time Specified
$\underset{\rightarrow}{r}_i, \underset{\rightarrow}{v}_i, t_i$
Transform Standard Estimated Primer Vector to ECI Coordinates
Specify Weighting Coefficients for Performance Indicator Computation
$W_1, W_2, \ldots, W_6$
Set "Initial-Phase" and "Last-Phase" Indices
$n = n_L$
$i = 1$
Set up n_L Engine Descriptions
$\dot{M}, M_o, t_d$
Return to Calling Routine

Figure III-1 OP-EX Guidance Algorithm Sheet 7 of 9

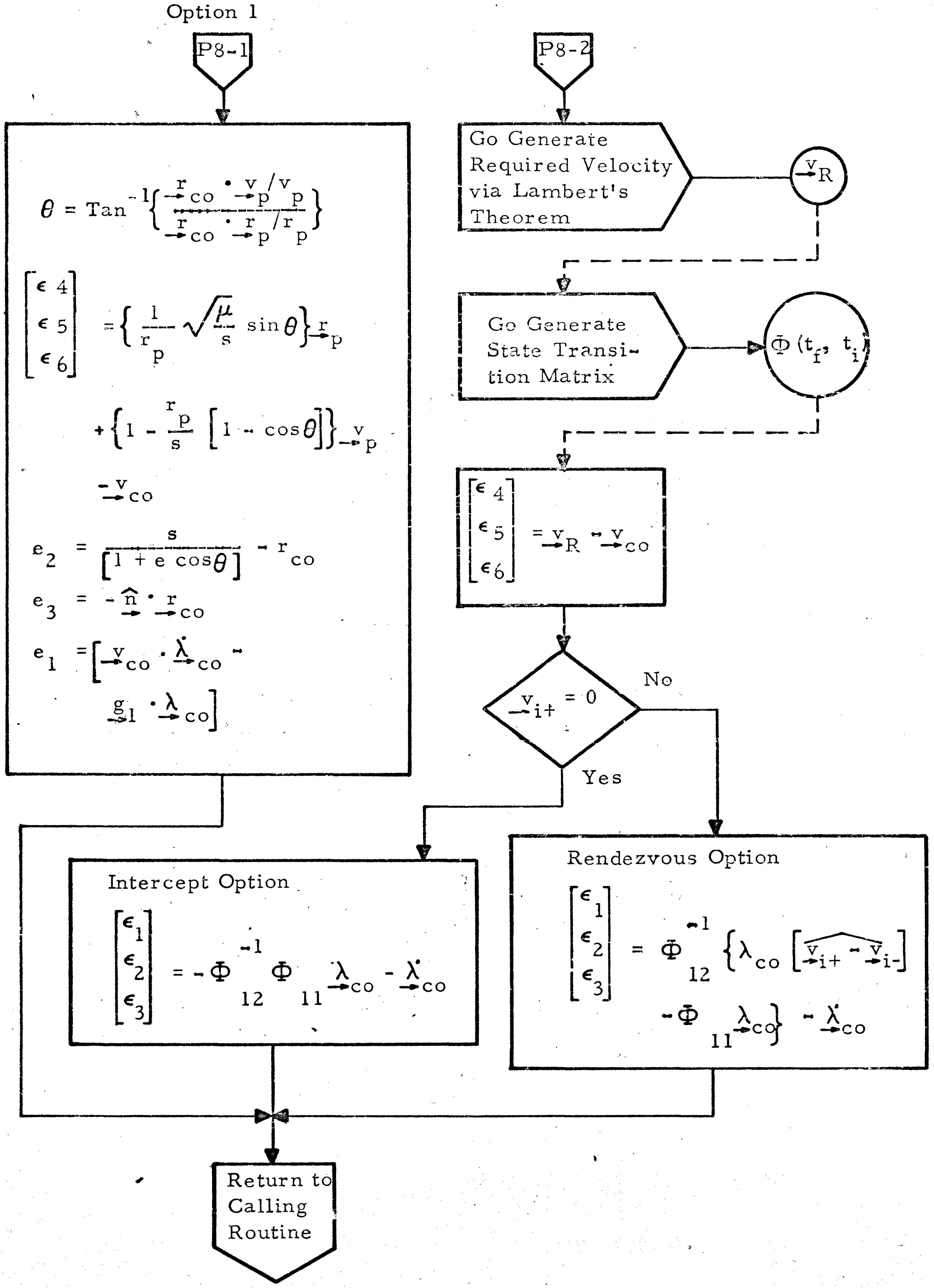

Option 1
P8-1
$\theta = \mathrm{Tan}^{-1}\left\{\dfrac{\underrightarrow{r}_{co} \cdot \underrightarrow{v}_p / v_p}{\underrightarrow{r}_{co} \cdot \underrightarrow{r}_p / r_p}\right\}$
$\begin{bmatrix}\epsilon 4\\ \epsilon 5\\ \epsilon 6\end{bmatrix} = \left\{\dfrac{1}{r_p}\sqrt{\dfrac{\mu}{s}}\sin\theta\right\}\underrightarrow{r}_p + \left\{1 - \dfrac{r_p}{s}\left[1 - \cos\theta\right]\right\}\underrightarrow{v}_p - \underrightarrow{v}_{co}$
$e_2 = \dfrac{s}{\left[1 + e\cos\theta\right]} - r_{co}$
$e_3 = -\underrightarrow{\hat{n}} \cdot \underrightarrow{r}_{co}$
$e_1 = \left[\underrightarrow{v}_{co} \cdot \underrightarrow{\dot{\lambda}}_{co} - \underrightarrow{g}_1 \cdot \underrightarrow{\lambda}_{co}\right]$
P8-2
Go Generate Required Velocity via Lambert's Theorem
$\underrightarrow{v}_R$
Go Generate State Transition Matrix
$\Phi(t_f, t_i)$
$\begin{bmatrix}\epsilon 4\\ \epsilon 5\\ \epsilon 6\end{bmatrix} = \underrightarrow{v}_R - \underrightarrow{v}_{co}$
$\underrightarrow{v}_{i+} = 0$
No
Yes
Intercept Option
$\begin{bmatrix}\epsilon_1\\ \epsilon_2\\ \epsilon_3\end{bmatrix} = -\Phi_{12}^{-1}\,\Phi_{11}\,\underrightarrow{\lambda}_{co} - \underrightarrow{\dot{\lambda}}_{co}$
Rendezvous Option
$\begin{bmatrix}\epsilon_1\\ \epsilon_2\\ \epsilon_3\end{bmatrix} = \Phi_{12}^{-1}\left\{\lambda_{co}\left[\widehat{\underrightarrow{v}_{i+} - \underrightarrow{v}_{i-}}\right] - \Phi_{11}\underrightarrow{\lambda}_{co}\right\} - \underrightarrow{\dot{\lambda}}_{co}$
Return to Calling Routine

Figure III-1 OP-EX Guidance Algorithm Sheet 8 of 9

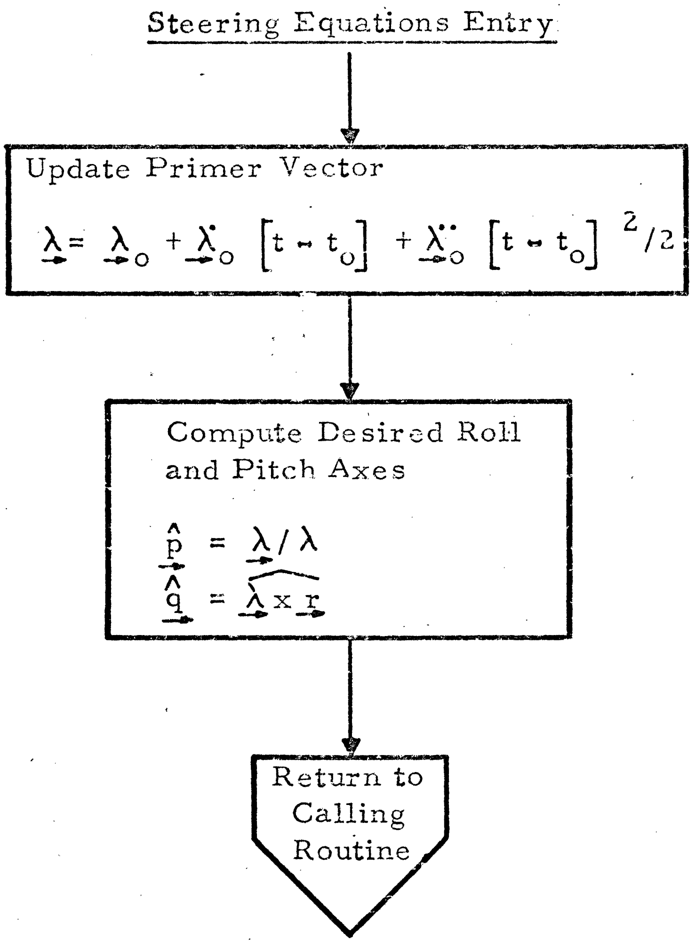

Steering Equations Entry
Update Primer Vector
$\underrightarrow{\lambda} = \underrightarrow{\lambda}_o + \dot{\underrightarrow{\lambda}}_o \left[t - t_o\right] + \ddot{\underrightarrow{\lambda}}_o \left[t - t_o\right]^2/2$
Compute Desired Roll and Pitch Axes
$\hat{\underrightarrow{p}} = \underrightarrow{\lambda}/\lambda$
$\hat{\underrightarrow{q}} = \widehat{\underrightarrow{\lambda} \times \underrightarrow{r}}$
Return to Calling Routine

Figure III-1 OP-EX Guidance Algorithm Sheet 9 of 9

Table III-1

OP-EX GUIDANCE MATH FLOW SYMBOL TABLE

$C(m)$	Effective exhaust velocity of stage m
e	Eccentricity of specified orbit (option dependent)
$\underset{\rightarrow}{g}_1, \underset{\rightarrow}{g}_2$	Gravitational acceleration at beginning and end of trajectory integration step
G_o	Gravity gradient matrix at t_o
G	Gravity gradient matrix during integration
i	Initial stage index
$I_1, \ldots, I_6$	Thrust integrals used during integration step
j	Newton-Raphson iteration count
J	Jacobian matrix of sensitivity coefficients
$\underset{\rightarrow}{J}_n$	nth column of Jacobian matrix
K	Newton-Raphson convergence control coefficient
$M_o(m)$	Nominal initial mass of stage m
$M'_o(m)$	Estimated present mass of stage m
$\dot{M}(m)$	Nominal mass flow rate of stage m
$\dot{M}'(m)$	Estimated mass flow rate of stage m
M_f	Nominal fuel mass in present stage
m	Present stage index
n	Final stage index
$\underset{\rightarrow}{\hat{n}}$	Unit vector normal to specified orbital plane (option dependent)

Table III-1. OP-EX Guidance Math Flow Symbol Table (continued)

$\underset{\rightarrow}{\hat{p}}$	Unit vector along desired body-fixed roll axis
P	Present value of performance indicator
P_L	Last value of performance indicator
P_{min}	Required value of performance indicator for successful convergence
$\underset{\rightarrow}{\hat{q}}$	Unit vector along desired body-fixed pitch axis
$\underset{\rightarrow}{r}_o$	Position vector at t_o
$G^{\underset{\rightarrow}{r}}$	Gravitational acceleration contribution to position vector
$T^{\underset{\rightarrow}{r}}$	Thrust acceleration contribution to position vector
$\underset{\rightarrow}{r}_{co}$	Position vector at t_{co}
$\underset{\rightarrow}{r}_i$	Position vector at intercept or rendezvous (option dependent)
$\underset{\rightarrow}{r}_p$	Specified perigee position vector (option dependent)
s	Semilatus rectum of specified orbit (option dependent)
t	Present time
t_o	Time at start of trajectory generation
t_{co}	Present predicted time of thrust termination
t_{coL}	Last predicted time of thrust termination
$t_d(m)$	Nominal burning time of stage m
$t'_d(m)$	Estimated remaining burning time of stage m
$t_f(m)$	Estimated time at end of stage m
t_{go}	Time of last guidance iteration
t_i	Time of intercept or rendezvous (option dependent)

Table III-1. OP-EX Guidance Math Flow Symbol Table (continued)

t_m	Time at midpoint of trajectory integration step
t_{po}	Time at beginning of present stage
t'	Running time during trajectory integration
$\hat{\underrightarrow{u}}_o$	Steering vector at t_o
$\hat{\underrightarrow{u}}_1, \hat{\underrightarrow{u}}_m, \hat{\underrightarrow{u}}_2$	Steering vectors at beginning, midpoint and end of trajectory integration step
$\underrightarrow{v}_o$	Velocity vector at t_o
$G^{\underrightarrow{v}}$	Gravitational acceleration contribution to velocity vector
$T^{\underrightarrow{v}}$	Thrust acceleration contribution to velocity vector
$\underrightarrow{v}_{co}$	Velocity vector at t_{co}
$\underrightarrow{v}_{i+}$	Target velocity at rendenzvous (option dependent)
$\underrightarrow{v}_{i-}$	Interceptor velocity at intercept (option dependent)
$\underrightarrow{v}_p$	Specified perigee velocity vector (option dependent)
$\underrightarrow{v}_R$	Required velocity (option dependent)
$\underrightarrow{v}_m$	Present velocity meter output
$\underrightarrow{v}_{mL}$	Last velocity meter output
W_k	Weighting factors for performance indicator computation
$\underrightarrow{\alpha}, \underrightarrow{\beta}$	Coefficients of parabolic approximation of steering vector over one trajectory integration step
δt_{co}	Correction to estimated time of thrust termination
$\delta \underrightarrow{\lambda}_o$	Correction to estimated initial primer vector

Table III-1. OP-EX Guidance Math Flow Symbol Table (continued)

$\Delta t'$	Trajectory integration time step
Δt_{max}	Maximum trajectory integration time step
Δt_{co}	Perturbation value for t_{co}
ΔV	Velocity meter increment added since last guidance iteration
$\Delta \lambda_o$	Perturbation value for components of $\underrightarrow{\lambda}_o$
$\underrightarrow{\epsilon}_o$	Error quantities for unperturbed trajectory
$\underrightarrow{\epsilon}$	Error quantities for perturbed trajectories
θ	True anomaly in specified orbital plane (option dependent)
$\underrightarrow{\lambda}_o$	Estimated primer vector at t_o
$\underrightarrow{\lambda}_{co}$	Primer vector at t_{co}
$\underrightarrow{\lambda}_{oL}$	Last estimated primer vector at t_o
$\underrightarrow{\lambda}_m$	Primer vector at midpoint of trajectory integration step
μ	Gravitational coefficient of spherical Earth model
ρ	"defined by equation"
τ	"defined by equation"
$\Phi(t_i, t_f)$	State transition matrix relating state at t_f to state at t_i (option dependent)
Φ_{nm}	Three-by-three partition of state transaction matrix (option dependent)

REFERENCES

1. Brown, K. R., and Johnson, G. W., "Rapid Computation of Optimal Trajectories", IBM Report 66-220-0002, November 1966.

2. Battin, Richard H., Astronautical Guidance, McGraw-Hill Book Company, New York, 1964, pp. 80-82.

3. Lion, P. M., and Handelsman, M., "The Printer Vector on Fixed-Time Impulsive Trajectories", AIAA Paper 67-54, January 1967.

Appendix IV

DIRECT SEARCH OPTIMIZATION PROGRAM "DSOP"

1. INTRODUCTION

A FORTRAN IV program named DSOP (Direct Search Optimization Program) has been developed for the experimental study of optimization algorithms and their application to mission planning in the QRGT study. DSOP finds a local minimum of any given function of a set of independent variables, subject to a given set of constraints. Explicit constraints (bounds on the independent variables) are handled directly. More general constraints are treated by the penalty-function method.

As the name implies, the optimization algorithms of DSOP are of the "direct search" type; that is, they make no use of analytical derivatives. A test problem (or any problem to which DSOP is applied) is therefore completely defined by equations for evaluating the function to be minimized and the constraint functions (if any). No equations for partial derivatives of these functions are required. This characteristic is highly desirable in an optimizer designed for use on-board a spacecraft, since it greatly reduces the number of routines that must be provided if there are a large number of different optimization problems which may require solution. Also, it makes the optimizer applicable to problems where analytic derivatives would be difficult or expensive to generate, and improves flexibility by simplifying the specification of new problems for optimization and the modification of existing ones to accommodate changes in mission objectives and/or constraints, etc.

DSOP consists of a main program and ten subroutines. This structure facilitates experimentation since the interconnection of the subroutines can easily be changed. The structure of MAIN is described in Paragraph 2.

The two principal search methods used are PMS (Pattern Move Search) and VA04A (Powell's method). PMS is a modification of the direct search technique developed by Hook and Jeeves (Ref. 1). VA04A is described in Powell's 1964 paper (Ref. 2). PMS, VA04A, and the other search methods used are described in Paragraphs 3 and 4 of this Appendix.

Results of this optimization experiments on test functions are presented in Paragraph 5. For the experiments without constraints, Powell's method (VA04A) gave the best performance. Experiments using penalty functions to enforce constraints were successful with PMS, but relatively unsuccessful with VA04A. The reasons for this are not fully understood at present.

This report emphasizes:

a) How to use DSOP

b) Block diagrams and listings for better understanding of DSOP so that future improvements can be easily implemented

c) Results obtained on certain test problems

The following paragraphs discuss the above mentioned items in more detail. The 7094 FORTRAN IV program listings are given in Paragraph 6. A list of references, with emphasis on those used in the work in this report, is given in Paragraph 7. For theoretical background, the reader is referred to these references. Many further references on direct search techniques are given in the referenced papers.

2. MAIN PROGRAM

The primary functions performed in the MAIN program are:

a) Input-Output

b) Selection of Type of Search

c) Pattern Search Logic

d) Penalty Function Cycling

A block diagram of the MAIN program is shown in Figure IV-1. Table IV-1 is a General Symbol Table which explains the most used symbols. Table IV-2 contains the symbols used in VA04A and such key symbols as $\underline{U}$, P etc. The underlined symbols are vectors, each with the dimension N, where N is the number of independent variables.

The first operation performed by MAIN is initialization and card reading. The formats for the input cards can be obtained from the listing in Paragraph 6.

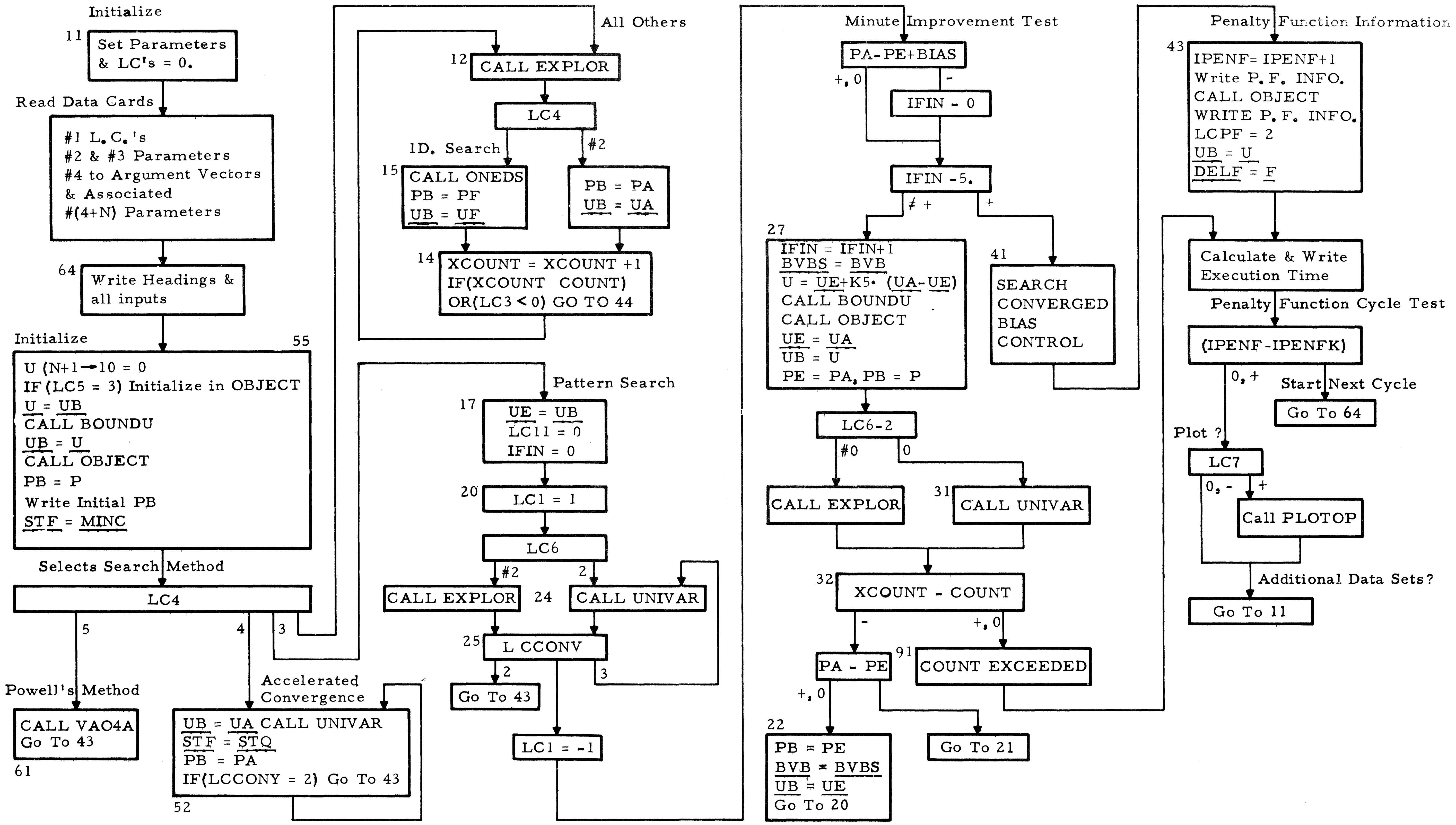


Figure IV -1 Block Diagram of Main Program for DSOP

GENERAL SYMBOL TABLE

Table IV-1

FORTRAN Symbol	Programs Where Used	Definition
BIAS	MAIN	Controls minute improvement exit test.
BVA	UNIVAR, EXPLOR	Input specifying if U components are to be fixed or varied.
BVB	UNIVAR	Records which U components are at bounds.
BVBS	MAIN	Storage vector for BVB.
DELE	OBJECT	Intermediate quantity used in PFV.
IFIN	MAIN	Counter used in minute improvement exit test.
IPENF	MAIN	Penalty function cycle counter.
IPENFK	MAIN	Number of penalty function cycles to be tried.
K1	EXPLOR	STF Parameter. $1/2N \leq K1 \leq 1/N$.
K2	EXPLOR	STE Parameter.
K3	EXPLOR	Success STE Parameter $K3 > 1$.
K4	EXPLOR	Parameter controlling interpolatory step.
K5	MAIN	Parameter controlling Pattern Move step size.
K11	UNIVAR	STE Parameter.
LCCONV	SUCCES, MAIN	Specifies state of convergence for last iteration.
LCPF	VA04A	Controls retaining W for multiple cycles.
LC1	SUCCES, MAIN	Controls exit on failure to reduce P.

Table IV-1. General Symbol Table (continued)

FORTRAN Symbol	Programs Where Used	Definition
LC3	SUCCES, EXPLOR, UNIVAR	Controls exit for converged.
LC4	MAIN	Type of basic search.
LC5	OBJECT	Selects an objective function.
LC6	MAIN	Selects UNIVAR or EMR for Pattern Move.
LC7	MAIN	Selects plotting for Pattern Move.
LC8	UNIVAR	Selects New Block Search.
LC11		Not used.
LC12	UNIVAR	Controls entry to quadratic fit.
LC14	OBJECT	Controls initialization for orbit transfer O. F.
LC15	OBJECT	Controls printout of objective function title.
LC17	OBJECT	Controls initialization for Test Problem #1.
MINC	UNIVAR, EXPLOR, VA04A	Minimum magnitude for stepsize in the U coordinates. (Also see Table 2).
NPT	SUCCES	Counter for points to be plotted.
OCOUNT	OBJECT	Objective function evaluation count.
OFV	OBJECT	Objective function value using penalty functions.
PA, PB	UNIVAR, EXPLOR, MAIN, NEWB	Remembered past values of P.

Table IV-1 General Symbol Table (continued)

FORTRAN Symbol	Programs Where Used	Definition
PC	ONEDS	Remembered past value of P.
PE	MAIN	Remembered value of PA for Pattern Move.
PF	ONEDS	Final iteration value for P.
PFV	OBJECT	Penalty function value.
PT	EXPLOR, UNIVAR, NEWB	Remembered past value of P.
PFCON	OBJECT	Penalty function parameter.
STE	EXPLOR	Vector used in generating "steps" which change U.
STF	EXPLOR, NEWB	Vector used in generating interpolatory step.
STG	NEWB	.5 (UA - UB) calculation for STF.
SUM	EXPLOR	Used in interpolatory step decision.
UA	EXPLOR, ONEDS, MAIN, NEWB	The "advance-point" - the best point found to date during the current exploration.
UB	EXPLOR, ONEDS, MAIN, NEWB	"Base point" (from which) exploratory moves begin.
UE	MAIN	Remembered values of UA in pattern search.
UF	ONEDS	Final iteration values for UA.

Table IV-1 General Symbol Table (continued)

FORTRAN Symbol	Programs Where Used	Definition
UT	NEWB	Used for UB storage.
UMAX	UNIVAR	Input upper bounds for U components.
UMIN	UNIVAR	Input lower bounds for U components.
XCOUNT	MAIN	Counter for number of UNIVAR or EMR iterations.

Table IV - 2

SYMBOL TABLE FOR VA04A

FORTRAN Symbol	Math Symbol[1]	Definition	Statement Sequence Numbers Where Set in VA04A Listing
A		Intermediate qty. used in D & DD	111
AAA		Convergence Control Parameter	187, 192, 200, 206, 209
B		Intermediate qty. used in D & DD	112
COUNT	MAXIT	No. of iterations allowed	Fixed, Input, 237
D	d	L.S. Increment Parameter	43, 60, 73, 108, 141, 172
DA	a	Direction Increment from DB to FA Coordinate	47, 71, 96, 100, 105, 115, 132
DACC		L.S. Parameter for Absolute Accuracy Cutoff	37
DB	b	Midpoint for Quadratic Fits	66, 69, 92, 107, 117, 134
DC	c	Direction Increment from DB to FC Coordinate	90, 100
DD	D	Change of D at current step & 2$^{\underline{cd}}$ derivative estimate	48, 142, 168
DDMAG		Used in DMAG Calculation	5, 84, 195, 227, 235
DDMAX		Ten Times DMAG	80, 83, 86, 103

[1] From Reference 5

Table IV-2 Symbol Table for VA04A (continued)

FORTRAN Symbol	Math Symbol	Definition	Statement Sequence Numbers Where Set in VA04A Listing
DI		Used in Absolute & Relative Accuracy Tests	120, 123
DL		Last Value of D	42, 49, 102
DMAG		Maximum Allowed Step Size	38, 39
DMAX		Used in DMAG Calculation	36
ESCALE		Limits Max. Change of Variables	Fixed Input, 5, 6, 18
FA	f_a	P value at an end point in l.s.	46, 70, 95, 99, 104, 114, 131
FB	f_b	P value at midpoint in l.s.	65, 68, 91, 106, 116, 133
FC	f_c	P value at an end point in l.s.	89, 109
FHOLD		Value of F after N l.s. & before s. in new direction	161
FI		Intermediate min. P setting	121, 124
FKEEP		Value of F at previous convergence, used in complex convergence procedure when ICON = 2.	25, 224, 252
FP		Value of P at beginning of iteration	27, 248, 251
FPREV		Value of P at beginning of l.s.	156, 157

Table IV-2 Symbol Table for VA04A (continued)

FORTRAN Symbol	Math Symbol	Definition	Statement Sequence Numbers Where Set in VA04A Listing
ICON		Controls restart for accuracy test	Fixed Input, 214
IDIRN		Index for s. Direction Components in $\underline{W}$.	34, 146, 183
ILINE		Index for s. Direction	35, 199, 197
IND		A control integer used when ICON = 2	11, 215, 233
INN		A control integer used when ICON = 2	12, 217, 267
IPRINT		A parameter controlling printing	Fixed Input, 150
IS		Controls Search Mode	77, 79, 82, 136, 138, 162, 45
ISGRAD		2 for 1st iteration, 1 thereafter	23, 235
ITERC		Count of iterations	22, 236
ITONE		New direction control	26, 126, 184
IXP		Index for $\underline{W}$ (N + !) components	29, 31, 163, 165, 186, 189, 199, 203, 253, 255
J		Controls integration of new direction	174
JIL	m	Value of ILINE for max. SUM	158, 247, 250, 264
JJ	$N^2 + N$	Index for new direction components	7
JJJ	JJ + N	Index for $\underline{W}$ (N + 2) components	8, 243

Table IV-2 Symbol Table for VA04A (continued)

FORTRAN Symbol	Math Symbol	Definition	Statement Sequence Numbers Where Set in VA04A Listing
K		A parameter for DO loops	9, 19, 50, 53, 178, 185, 193, 218, 220, 256
MINC	$\underline{E}$	E(I) specifies required accuracy of Ith argument	Fixed Input, 16, 205, 206
N	n	Dimensionality of space	Fixed Input
NFCC		Count of Function Evaluations	56, 225
P	$F = f(p)$	P(U) - the function to be minimized	55, 140, 201, 225, 241
SCER		f(ESCALE) used in calculating DACC	6, 37
SUM	Δ	Max [FPREV - P] ILINE 1→N	28, 157
$\underline{U}$	p	N - component argument vector	52, 144, 204, 222, 261
$\underline{W}(o)$ [2]		Holds direction scaling variables	18, 148, 196
$\underline{W}(j)$ [2]	$\xi_1 \cdots \xi_n$	N stored search directions	17, 145, 190
$\underline{W}(N+1)$ [2]	$p_n - p_o$	$\underline{U}$ from iteration or new search direction	32, 166
$\underline{W}(N+2)$ [2]		Terminal point of a previous convergence when ICON = 2	

[2] For notation simplicity write: $W(I, J) = W(N\,J+I)$ where $1 \leq I \leq N$ and $\underline{W}(J) = [W(I, J),\ W(2, J), \ldots W(N, J)]$

As shown in Figure IV-1, the program input LC4 selects the search procedure to be used. Setting LC4 = 5 selects VA04A, LC4 = 4 selects Accelerated Convergence, LC4 = 3 selects PMS, LC4 = 2 selects EXPLOR with ONEDS, and LC4 = 1 selects EXPLOR alone. Additional options can easily be added. All of these search procedures are subroutines except PMS and Accelerated Convergence, which are built into MAIN. The operation of PMS will be explained in the next section. PMS is not a self-contained optimization algorithm; it requires an "exploratory" subroutine which performs a restricted local search about a given point. The two exploratory subroutines presently included in DSOP are EMR (Exploratory Move Routine) and UNIVAR. EMR is modeled after the exploratory subroutine used by Hook and Jeeves (Ref. 1). UNIVAR is similar to part of a published optimization algorithm called BEST UNIVAR (Ref. 3). These subroutines are described in Paragraph 4. "Accelerated Convergence" was an experimental procedure combining a modified UNIVAR (produced by setting program input LC8 positive) with a special subroutine called NEWB. The Accelerated Convergence method, which was an attempt to incorporate part of Powell's method into UNIVAR, produced generally favorable results in experiments with test functions (see Table IV-3 and Paragraph 5) but has been superseded by VA04A.

PMS and UNIVAR incorporate logic for enforcing upper and lower limits on each component of the argument vector. The other search algorithms do not include any provisions for constraints. However, regardless of the search algorithm selected, MAIN provides means for handling general constraints by use of penalty functions. In the penalty-function method, the objective function is modified by adding extra terms which depend on the constraints. This enables a constrained minimization problem to be solved (approximately) by a sequence of unconstrained minimizations. Each cycle of the sequence consists of an unconstrained minimization of the modified objective function, followed by an adjustment of the coefficients in the penalty terms. In the usual form of the penalty function method, these coefficients are scale factors which are increased each cycle. DSOP experiments have been made with a more sophisticated scheme, in which the penalty terms include bias

Table IV-3

TABLE OF THE NUMBER OF OBJECTIVE FUNCTION EVALUATIONS FOR SEVERAL D.S. SCHEMES AND TEST FUNCTIONS

				Pattern Move		Powell's VA04A ESCALE = 1000. ICON = 2	
Test Function (Starting Value)	LC5 Value	Point of Recording	Accel-erated Conver-gence Test	With New Block	With-Out New Block	MINC equals .005	MINC equals .05
Quadratic (0, 1)	1	N.A.(1) C.	31. 212.	74. 305.	111. 152.	16. 34.	5. 13.
CUBE (0, 1)	2	N.A.(1) C.	183. 368.	451. 544.	247. 247.	45. 120.	7. 14.
Colville's T.P. #4 (-3, -1, -3, -1)	6	N.A.(1) C.			1039. 1235.	814. 944.	183. 330.
Colville's T.P. #1 (0, 0, 0, 0, 1) 1 cycle only	7	N.A. C.			555.	3075. (PFCON= 1000.)	333.(2)

N.A. - Near Answer
C. - Cutoff Point
(1) - Under .5% error in each component (.0005 for 0. minimum)
(2) - Objective function value close to correct value but coordinates in error (also ICON = 1).

coefficients that are adjusted each cycle, in a manner designed to give rapid convergence to the constrained minimum.

Figure IV-1shows the logic which handles penalty function cycling. Each cycle, the hopefully improved values of U and F are stored in UB and DELE and the flow is routed to block 64 to start the next cycle. OBJECT is called again in block 43 because the last computed penalty function values may not correspond to the best obtained values of U. The input IPENFK controls the number of cycles. This automatic cycling has only been used on Test Problem #1 but it could easily be generalized for use with most objective functions with implicit constraints.

The mathematical form of the penalty function calculation for Test Problem #1 in OBJECT originally was:

$$P = P_{o.f.} + PFCON \sum_i f^2 (E_i + \Delta E_i)$$

where: $f(x) = \min (o, x)$

$$E_i = \left[\sum_{j=1}^{10} a_{ij} u_j - b_{\underline{i}} \right]$$

$$\Delta E_i = o \text{ for 1st cycle}$$

and the summation on i goes over i = 3, 4, 5, 6, 9 only since it can easily be shown that the others need not be considered. At the end of the convergence cycle, set $E_i = f(E_i)$ using the converged U values and start the next cycle. However, recent tests have indicated that only i = 5 and 6 were active for VA04A trials on T.P. #1, and OBJECT reflects this. Manual cycling was used with the Penalty Function Test function shown in Figure IV-6.

3. PATTERN MOVE SEARCH (PMS)

The basic logic of Pattern Move Search is shown in Figure IV-2, and its detailed logic shown in Figure IV-1. The program makes use of an exploratory subroutine, which operates in two different modes. The exploratory subroutine starts at a given base point UB, and makes exploratory moves, according to built-in rules, in a search for a smaller value of P. The output of the exploratory

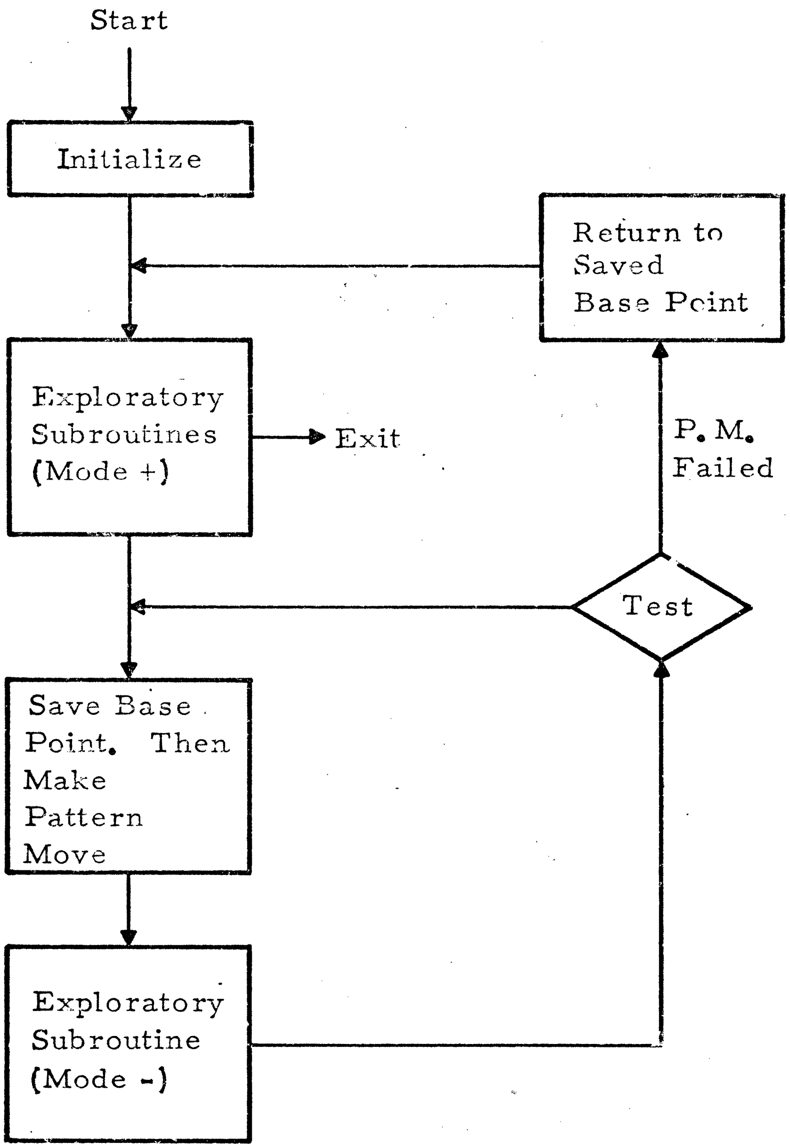

Start
Initialize
Return to Saved Base Point
Exploratory Subroutines (Mode +)
Exit
P. M. Failed
Test
Save Base Point. Then Make Pattern Move
Exploratory Subroutine (Mode -)

Figure IV-2 Basic Logic of PMS

subroutine is an "advance point" UA and function value PA, with PA ≠ PB if the search succeeds, and with PA = PB, UA = UB if it fails. In "Mode Plus", the subroutine keeps trying, varying step sizes and/or other search parameters, until it either succeeds, or concludes that PMS has converged to a solution. In "Mode Minus", the subroutine stops after one run through its exploration sequence, regardless of success or failure.

Pattern Move Search proceeds by generating a sequence of accepted base points, each of which gives a lower function value than the previous one. A "pattern move" generates a tentative new base point by displacing the newest accepted base point by a vector equal to (or proportional to) the difference between the newest and next-newest accepted base points. The tentative new base point is improved by use of the exploratory subroutine (in "Mode Minus") and the result tested to see if it is an improvement. If so, it becomes the newest accepted base point, and the cycle repeats. If it is not an improvement, i.e., if the pattern move (together with its associated local exploration) is a failure, the program returns to the newest accepted base point (UE) and executes the exploratory subroutine in "Mode Plus". If this fails, the search ends. If it succeeds, the output and input of the subroutine become the newest and next-newest accepted base points respectively.

A special merit of Pattern Move Search is that the tentative new base point generated by a pattern move is not tested for success until after an attempt has been made to improve it by a local exploration. This means that Pattern Move Search can follow a curving valley; even if a pattern move misses the valley floor and hence gives a high function value, the subsequent exploration sequence may find the valley again, so the over-all move is a success. Search programs which always reject points that do not show an immediate improvement will sometimes be much less efficient than Pattern Move Search. (Sometimes they may be more efficient; persistence is not always a virtue.)

Most of the logic in Figure IV-1 can be understood from the previous discussion, and comparing with Figure IV-2. FORTRAN reference numbers to the left of the blocks are used in the following discussion. LCCONV is a branching control quantity set by the exploratory subroutine in subroutine

SUCCES, and subsequently tested by PMS to decide whether to continue or to terminate the search. Another possible termination of the search is by the Minute Improvement Test, which counts the number of consecutive iterations that have failed to improve the function value by amounts above a given threshold, named BIAS. A third mode of termination occurs if the number of exploratory moves, XCOUNT, exceeds a given limit, COUNT.

Block 17 is initialization. Block 27 saves the newest accepted base point (labeling it $\underline{UE}$) and performs a pattern move. The coefficient K_5 in Block 27 is currently set at $K_5 = 2$. By choosing K_5 greater than 2, the lengths of a sequence of pattern moves can be made to grow more rapidly, but with increased probability of failure. Block 22 restores the saved base point after failure of the pattern move. The vector $\underline{BVB}$, which remembers which components of the argument vector are at their bounds (and hence requires different treatment in the exploratory subroutine) is saved and restored along with $\underline{UE}$. The operation denoted by BOUND U, in Blocks 55 and 27, is a subroutine that examines each component of U to see if it exceeds the prescribed upper or lower bound. If so, $\underline{U}$(I) is set equal to whichever bound was violated, and $\underline{BVB}$ is changed accordingly.

Pattern Move Search provides a very general framework into which almost any type of optimization algorithm can be fitted as an exploratory subroutine, with minimal changes in PMS itself. For example, if it should be decided to modify the optimizer so it can handle general sets of equality and/or inequality constraints, this could be done by developing a new exploratory subroutine with the desired capabilities. The only changes required in PMS itself would be (a) replacing BOUNDU with a more elaborate operation which enforces, or approximately enforces, all active constraints, by projecting the search point onto constraint surfaces, and (b) saving and restoring information that remembers which constraints are active; this requires an extension of the $\underline{BVB}$ vector.

The PMS program and its associated subroutines were first designed for minimization without constraints, and modified until they accomplished this reliably and moderately efficiently on a set of test problems.

From these experiments, UNIVAR appeared to be a better "valley-follower" than EMR and was therefore selected for further development. PMS and UNIVAR were modified to provide for the enforcement of specified upper and lower bounds for each component of the argument vector. Also, PMS was modified to incorporate an improved stopping rule (the "Minute Improvement Test"). EMR was not modified, but EMR and the original version of UNIVAR are still usable for problems where argument bounds are absent, or do not affect the solution.

4. SUBROUTINES

The ten subroutines and their interrelationships are indicated in Figure IV-3. The arrows indicate which subroutines are called by each individual subroutine. The subroutines are divided into two groups: the primary search techniques, and accessory subroutines which accomplish smaller tasks. The subroutine OBJECT evaluates the objective function, and therefore is used by all the search programs. Block diagrams of the key subroutines are included in Figures IV-4. IV-5 and IV-7.

4.1 VA04A (POWELL'S METHOD)

Powell's method is what is known as a "conjugate direction" method. It proceeds by a sequence of one-dimensional searches. The search directions are chosen in a manner which will find the exact minimum of a quadratic function of n variables with continuous second derivatives, in a finite number of steps. For functions which are not quadratic, but can be approximated by quadratic functions in the neighborhood of the minimum, rapid convergence may be expected. VA04A has no provisions for handling constraints except by penalty functions.

A detailed flow diagram of VA04A, as defined by Powell's unpublished program listing (Ref. 4) is given in Figure IV-4. The principal symbols are listed in Table IV-2. To understand the program logic, which is fairly complex, Figure IV-4 and Table IV-2 should be studied together with Powell's paper (Ref. 2) which explains the principles behind his program. The basic operating cycle of VA04A has been concisely described in a review paper by Fletcher (Ref. 5) who presents it in pseudo - ALGOL notation as:

```
y:    = x;
for i:  = 1 step 1 until N do MIN(i);
```

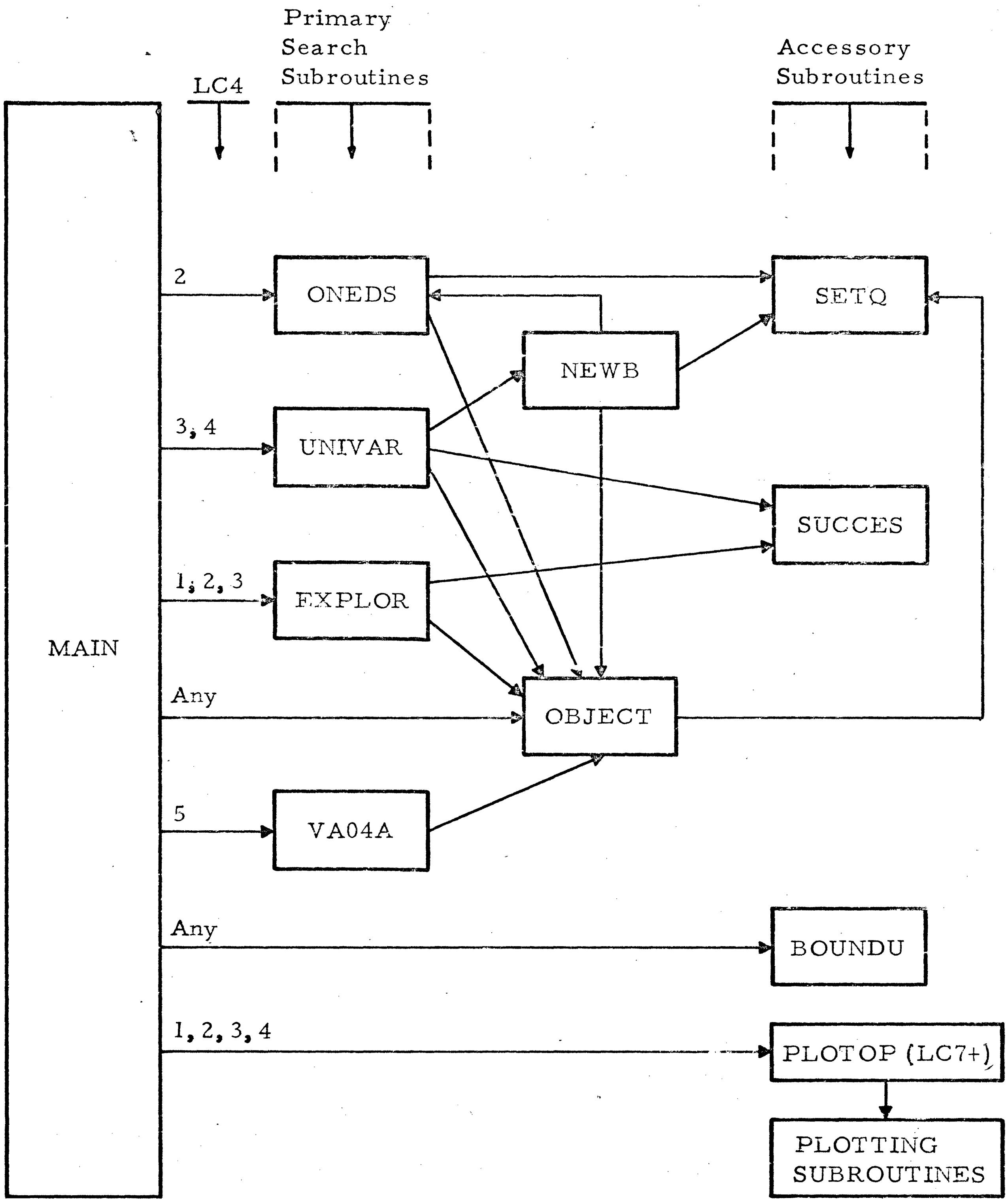


Figure IV-3 Interconnection of Subroutines

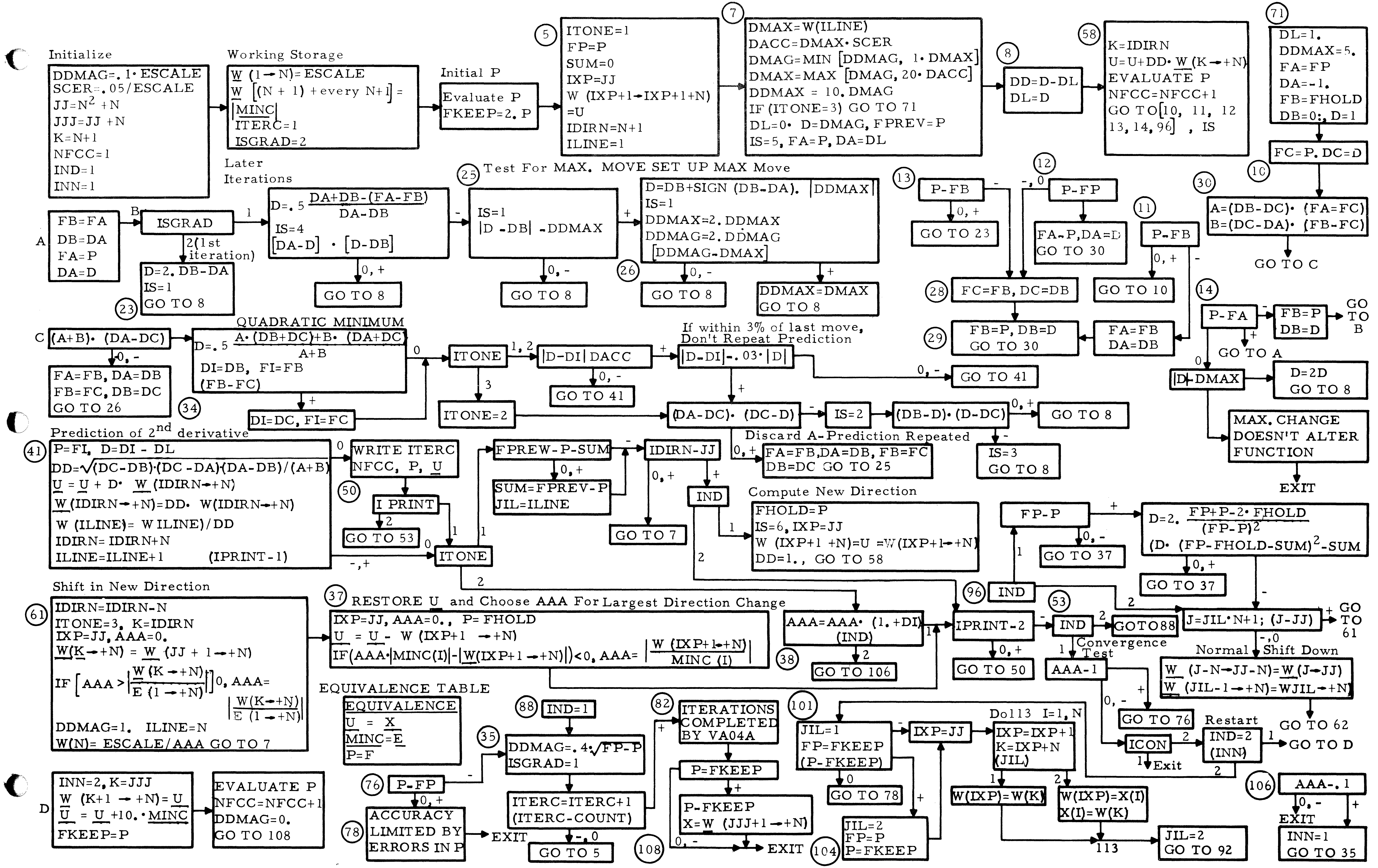


Figure IV-4 Block Diagram of M.J.D. Powell's Direct Search Optimization Subroutine "VA04A"

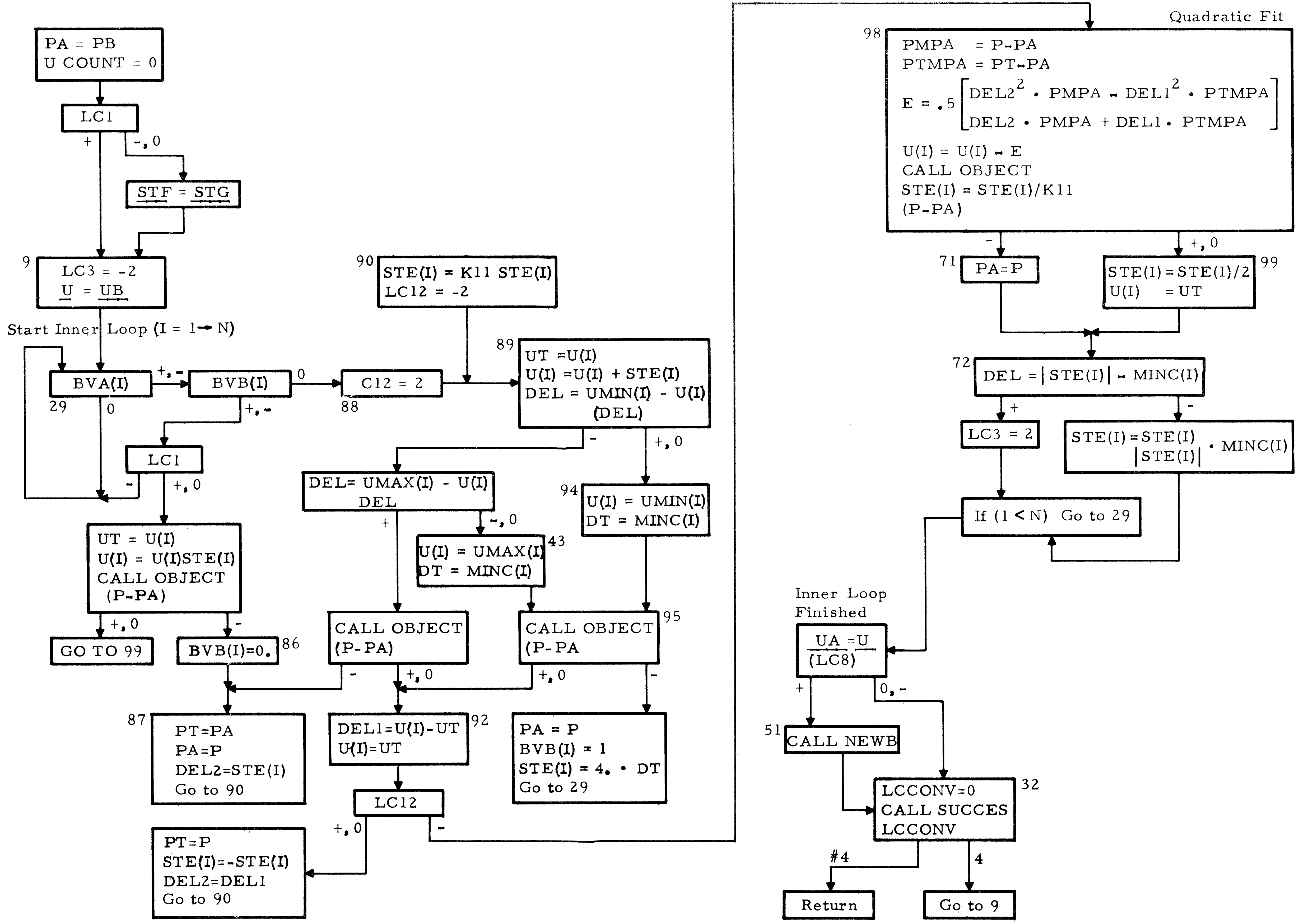


Figure IV-5 Block Diagram of UNIVAR Subroutine

FUNCTION	LC5 SETTING	TEST OBJECTIVE FUNCTION P
Quadratic	1	$u_1^2 + 50. - u_2^2$
Cube	2	$100. (u_2 - u_1^3)^2 + (1. - u_1^2) + 10. - u_3^2$
Orbit Transfer	3	Lengthy Evaluation
Sharp Valley	4	$(1. - u_1^2) + 100. \left\lvert u_2 - u_1^3 \right\rvert$
Penalty Function Test	5	$F(u) + A \bullet G(u) + B \bullet G(u)^2$
Colville's Test Problem #4	6	$100. (u_2 - u_1^2)^2 + (1. - u_1)^2 + 90. (u_4 - u_3^2)^2 +$ $(1. - u_3)^2 + 10.1 \ (u_2 - 1)^2 + \left[(u_4 - 1)^2\right] +$ $19.8 \ (u_2 - 1) (u_4 - 1)$
Coleville's Test Problem #1	7	$\sum_{j=1}^{5} e_j u_j + \sum_{j=1}^{5} \sum_{j=1}^{5} c_{ij} u_i u_j + \sum_{j=1}^{5} d_j u^3_j$ And constraints are present.

Figure IV-6 Available Test Functions in Subroutine "Object"

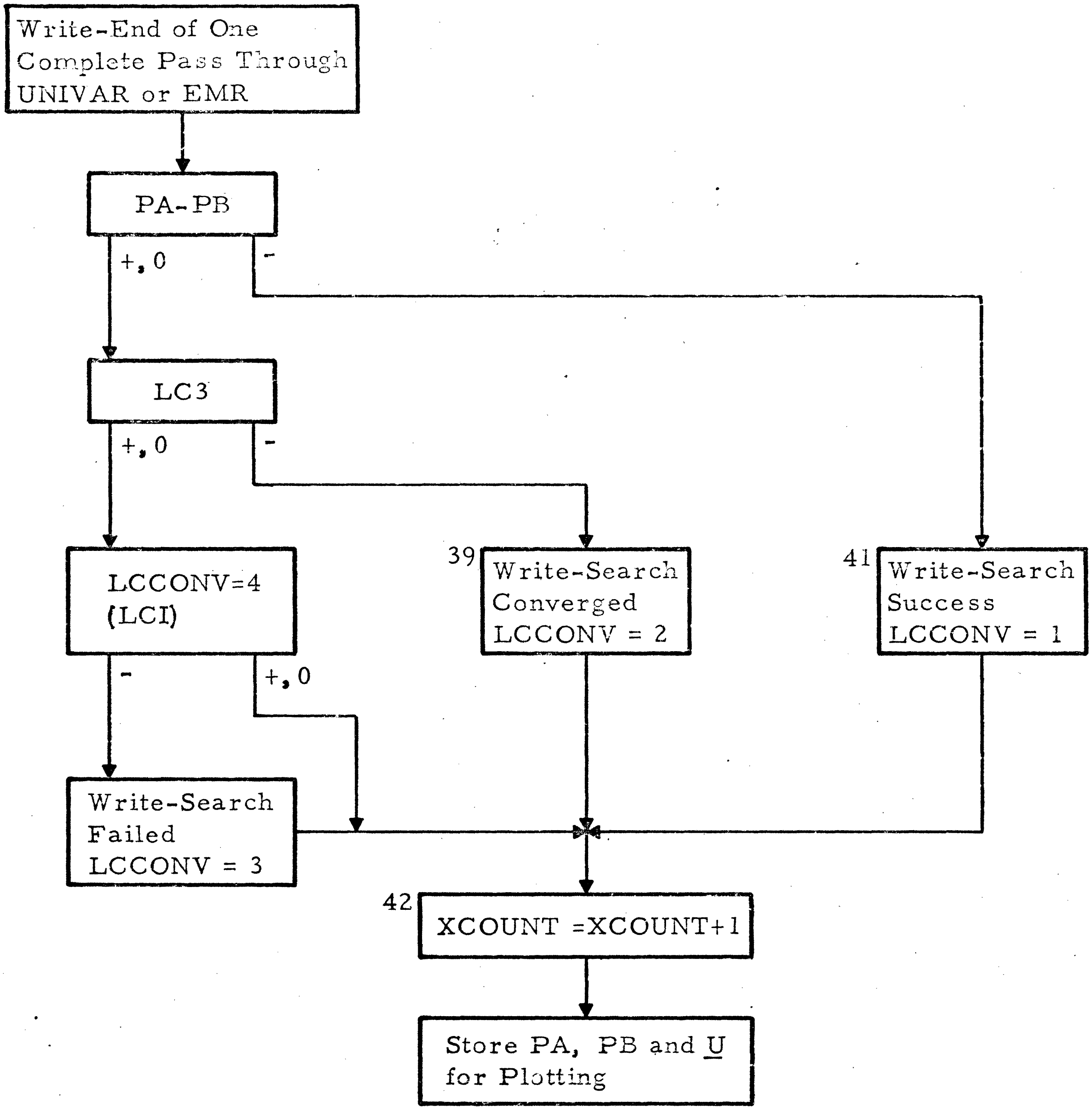

Write-End of One Complete Pass Through UNIVAR or EMR
PA-PB
+, 0
-
LC3
+, 0
-
LCCONV=4 (LCI)
-
+, 0
39
Write-Search Converged LCCONV = 2
41
Write-Search Success LCCONV = 1
Write-Search Failed LCCONV = 3
42
XCOUNT =XCOUNT+1
Store PA, PB and U for Plotting

Figure IV-7 Block Diagram for Subroutine Success

```
for i: = 1 step 1 until N-1 do $p_i$: = $p_{i}$ + 1;
$p_N$: = y = x; MIN(N).
```

where x is an N-component argument vector (equivalent to the U of our notation), y is the value x had at the beginning of the cycle, and MIN(i) is a subroutine which performs a one-dimensional minimizing search parallel to p_i, starting from the best point found previously. The vectors p_i are originally chosen to be parallel to the coordinate directions. The routine performs a linear search parallel to each p_i in sequence. The net resulting change in x defines a new p-vector, and all the p-vectors shift down in the list, the oldest (the previous p_i) being discarded.

Powell's actual procedure is somewhat more complex than this. It is evident that if his set of vectors p_i ever become linearly dependent, his search process becomes trapped on a hyperplane and will never find the solution if it is not on this plane. To avoid this, he does not always discard the oldest vector; another vector, chosen by a rather complex set of rules, is sometimes discarded instead. Also, the newly generated direction is not always accepted.

Powell's program performs N+1 one-dimensional searches per iteration, each search requiring at least two new function evaluations. It will find the exact minimum of a quadratic function in N iterations, requiring a total of $N^2 + N$ linear searches. For non-quadratic functions, the exact answer is generally not found, and more than N iterations will usually be required to obtain the accuracy desired. The convergence is ultimately quadratic. That is, if the function to be minimized behaves like a quadratic function in the neighborhood of the optimum, then if the search point is sufficiently close to the solution and at least N cycles have occurred, each subsequent iteration will double the number of significant figures. The number of linear searches required for minimizing a quadratic function could be halved by a slight change in the program logic, but this change would not double the program's efficiency for minimizing non-quadratic functions; at most, it might save half of the first $N^2 + N$ searches that would otherwise be performed, but subsequent operation would not be improved.

The version of VA04A used in DSOP differs from Powell's original program by two small changes. The first of these changes eliminates a difficulty that would otherwise occur in unusual conditions: The zero output of the branch on FP-P, at Block 96, is now treated like a negative output (GO to 37). This change prevents a division by zero which would otherwise cause the program to fail. The second change was introduced to facilitate the efficient use of VA04A with penalty-function cycling. It provides an option of bypassing Powell's initialization, so search directions and scaling are preserved from the last use of the VA04A routine.

4.2 UNIVAR

The subroutine UNIVAR exists in two versions: an original version which has no provision for bounds on the argument variables, and a revised version that incorporates such bounds. Either version works with PMS, but only the revised version, whose flow diagram is shown in Figure IV-5, is included here.

The original UNIVAR proceeds by making a sequence of one-dimensional searches, each parallel to a coordinate direction, until all directions have been tried except those for which BVA(I) = 0. Each one-dimensional search begins from the best point found by the previous search. The logic for a one-dimensional search is as follows:

Step in the stored direction (and by the stored amount; both are defined by a component of the step vector STE). If this succeeds, increase step size and repeat. If it fails, reverse direction and go the other way. Proceed until the minimum along the search direction has been bracketed, i.e., until a set of three points has been found with the smallest function-value in the middle. This occurs when a success is followed by a failure, or when the first step fails and is immediately followed by a second failure when the reverse direction is tried. Fit a quadratic function to the values found at these three points, and try the point where this quadratic is minimal. The search result is either this new point or the previous middle point, whichever has the least function-value. The step size is reduced to cancel the last increase and is further reduced by an extra factor of one-half if the last trial (from the quadratic fit) is not a success. However, step size is not allowed to go below a prescribed minimum size.

This one-dimensional search procedure is known to be inefficient, since it requires a minimum of three new function evaluations per direction, whereas Powell's one-dimensional search procedure (based on remembering and updating an estimate of the second derivative) reduces the minimum to two. Adopting Powell's rules for one-dimensional searches might reduce the number of function evaluations for each execution of UNIVAR by as much as 33%. The average reduction would be less than this, but still significant. This improvement has not been incorporated because other developments have had higher priority.

The revised UNIVAR is the same as the original UNIVAR except for extra logic to handle upper and lower bounds on the independent variables. The binary-valued vector BVB keeps track of which variables are at their bounds. If BVB(I) = 0, indicating that the Ith coordinate is in the interior of its allowed range, the search along the Ith direction is the same as in the original UNIVAR, unless a tentative step reaches or exceeds a boundary. In this case, a step to the boundary is tried instead. If this step fails, the search along the Ith direction will end at an interior point. If it succeeds, BVB(I) is set to 1, STE(I) is set to call for a small step (4 times the minimum size) away from the boundary, and the routine goes on to the next direction.

If BVB(I) = 1, indicating that the Ith variable is at a boundary, this coordinate is left unaltered unless LCI = +, in which case a step away from the boundary is attempted. If this succeeds, BVB(I) is set to 0 and the search proceeds as from an interior point. If it fails, step size is reduced to half its previous magnitude, or to the minimum magnitude allowed, whichever is greater.

The motivation for restricting attempted moves away from a boundary to cases when LCI = + (i.e., to times when the pattern-move sequence is being started for the first time, or is being restarted from an old base point after failure of a pattern move) is to reduce the number of unsuccessful attempts to move away from the boundary that would otherwise occur if the true final solutions call for one or more variables to be at their bounds. Since search termination on step size is permitted only when LC1 = +, the program cannot exit on step size without having tried moves away from all boundaries. It can "exit on test" if there is a prolonged sequence of pattern moves which are all successful but give very small improvement, but this is highly unlikely; in general, one of the pattern moves will fail, and the next execution of UNIVAR will attempt moves away from the boundaries.

4.3 EXPLOR

This was the first "exploratory" program tried with the Pattern Move Search. Its principles are described in Reference 1. No exercised constraints are allowed. Test runs generally indicated EXPLOR was inferior to UNIVAR (see Paragraph 5 of this Appendix). The EXPLOR listing in Paragraph 6 can be used to follow the discussion below.

Components of the argument vector U̲ for which BVA(I) = 0 are treated as constant constraints; other components of U̲ are varied systematically in a search for a local minimum. For each value of I for which BVA(I) = 1, the routine tentatively changes U(I) in one direction and then (if this fails to reduce the function value) in the opposite direction. The direction tried first is determined by the sign of STE(I), and the magnitude of this quantity determines the step size. If either change succeeds, the value of PA (which is the best value of P found so far) is updated, and the search proceeds from the altered argument vector, by considering the next value of I. If both directions of change along the Ith coordinate axis are unsuccessful, no change is made in U(I), and the routine moves on to the next value of I after saving some auxiliary information.

When all variable components of U(I) have been treated as described above, the routine decides whether to accept the result, or try one additional "interpolatory" step, whose components are given by a vector STF. The value of STF(I) is zero for any value of I for which BVA(I) = 0 or for which either of the steps ± STE(I) succeeded. For the remaining cases (i.e., BVA(I) = 1, and ± STE(I) both unsuccessful), STF(I) is computed by considering the location of the minimum of a quadratic function fitted to the function values found at the three points U(I) and U(I) ± STE(I), with all other components of U fixed. If K_1 were chosen to be 1/2, and no other coordinates were involved, STF(I) would give a move to this minimum. A value of K_1 less than 1/2 is appropriate in multidimensional cases. If P(U) is a quadratic function, and if all coordinate steps have failed, it can be shown that the interpolatory step always succeeds if $K_1 < 1/N$, and is always too short (undershoots the minimum) if $K_1 < 1/2N$. Therefore, K_1 should be between 1/N and 1/2N.

The decision to try the interpolatory step, or not try it, is made by comparing the total function reduction obtained by steps ± STE(I) with the expected further reduction obtainable by the interpolatory step. This expected reduction is proportional to SUM, if interaction between coordinates is ignored. A coefficient K_4 is provided which can be adjusted to bias the decision. If $K_4 \leq 0$, the interpolatory step is never tried unless all the steps ± STE(I) failed. If K_4 is made very large, the interpolatory step will always be tried.

The step vector STE is updated during the routine, to provide a favorable step for next time. If step STE(I) is successful, the new STE(I) is larger ($K_3 > 1$). If STE(I) fails but the reverse step succeeds, STE(I) reverses sign but does not change magnitude. If both steps ± STE(I) were unsuccessful the new STE(I) has its sign determined by comparison of the two unsuccessful trials.

An input option (LC1) is tested in SUCCES to allow the routine to exit on failure to reduce the function, or try again with reduced step sizes until either success is attained, or the step sizes are reduced to the allowed minimums given by MINC.

The performance of EMR could probably be improved significantly by adding a one-dimensional search along the direction defined by STE, i.e., parallel to the "interpolatory move." However, this experiment has not been tried.

4.4 ONEDS

This is simply a one-dimensional search program which is used by New Block and with a search variation with EXPLOR in the MAIN program.

4.5 OBJECT

This program computes a single number which corresponds to the value of the objective function. The inputs are essentially just the independent variables U. Seven different test functions (see Figure IV-6) have been used in the experimental tests of the optimization algorithms. The selection is controlled by the input integer LC5. The two dimensional version of CUBE was taken from Reference 3; Colville's test problems came from Reference 6; the orbit transfer problem came from Reference 7.

Additional functions can be readily added to OBJECT by using new LC7 branches. Also, any available function-evaluation program can be substituted for OBJECT by appropriately renaming variables and adding COMMON statements.

4.6 NEWB

NEWB (New Block) uses some of Powell's basic ideas to accelerate convergence. As is shown in Table IV-3, it has been used both with the separate accelerated convergence test and in UNIVAR with PMS.

4.7 SUCCES

This program is used by UNIVAR and EXPLOR after every iteration to test the condition of the search and set LCCONV accordingly. LCCONV is then tested in the PMS in the MAIN program. Also the iteration counter, XCOUNT, is updated and information for plotting is stored. See Figure IV-7 for the Block Diagram.

4.8 SETQ

SETQ performs some simple necessary settings for UA, UB, PA and PB.

4.9 BOUNDU

This program limits all components of U to their bounds, called UMAX and UMIN. These quantities must be inputted on cards (or by modification through COMMON). BOUNDU is called exclusively from MAIN, initially and in PMS. Similar operations are also performed in UNIVAR.

Table IV-4

A. R. COLVILLE'S FORMAT FOR TEST PROBLEMS # 1 AND # 4

Test Problems & Search Method	MINC	OBJ. Function Value	Prep. Time Guess (Hrs.)	Execute Time (Sec.)	Funct. & Constr. Calcs.	Std.[2] Time Ratio
Test Problem #4 Powell (ICON = 2)	.05[1]	0.	.5	1.3167	330.	.0196
Test Problem #4 Pattern with UNIVAR (ICON = 2)	.005	0.	.5	3.667	1235.	.0545
Test Problem #1 Powell (1 cycle) (ICON = 1)	.05[1]	-32.38	1.	2.450	333.	.0364

[1] MINC = .05 for Powell is equivalent to .005 for Pattern Move since Powell tests for .1 (MINC = E)

[2] 7094 Standard Time = 67.2 sec.

Table IV-5

ACCURACY AND TIMING COMPARISONS BETWEEN UNIVAR AND EXPLOR WITH PMS

Test Problem = CUBE

Starting Conditions	Running Time (sec.) UNIVAR	EMR
Case 1 (0, 0, 0)	2.583	2.317
Case 2 (10, 0, 0)	1.683	2.466

	UNIVAR				EXPLORATORY			
	OCO UNT	P	U(1)	U(2)	OCO UNT	P	U(1)	U(2)
	7	1.	0.	0.	6	0.9999	$0.1056.10^{-4}$	0.
	1	1.	0.	0.	18	0.9996	$0.2082.10^{-3}$	$0.14021.10^{-7}$
	31	0.4727	0.31250	0.03052	30	0.9905	$0.4779.10^{-2}$	$0.10216.10^{-7}$
Case	51	0.4172	0.35408	0.04439	54	0.5849	0.23575	0.01031
1	79	0.1901	0.56310	0.17855	78	0.2758	0.47495	0.10603
	113	0.07818	0.72039	0.37386	112	0.06079	0.75354	0.42716
	147	0.02546	0.84044	0.59365	144	0.00223	0.95286	0.86495
	180	0.00394	0.93726	0.82334	177	$0.606.10^{-4}$	1.0075	1.02294
	220	$0.655.10^{-6}$	0.99919	0.99757	199	$0.570.10^{-4}$	1.0075	1.02281
	8	1.	$.596.10^{-7}$	0.	10	0.57554	0.26401	0.
	23	1.	0.	0.	24	1.4454	0.26401	0.
	35	1.	0.	$0.254.10^{-8}$	34	0.6165	0.26401	0.
	55	0.47266	0.3125	0.030518	55	0.3370	0.42102	0.078811
Case	82	0.28749	0.46381	0.09978	84	0.3008	0.45322	0.088791
2	114	0.12412	0.64769	0.27170	115	0.1194	0.65488	0.279237
	147	0.04666	0.78398	0.48186	149	0.0255	0.84058	0.59484
	182	0.01184	0.89119	0.70781	183	$0.74.10^{-5}$	1.0008	1.0028
	238	$0.655.10^{-6}$	0.99919	0.99757	204	$0.11.10^{-5}$	1.0009	1.0026

4.10 PLOTOP

UA, UB, PA and PB from UNIVAR and EMR can be plotted as functions of XCOUNT by setting LC7+. The necessary storage of quantities is done in SUCCES.

5. COMPARATIVE TABLES

Four of the test functions shown in Figure IV-6 were used with various search methods to determine the number of objective functional evaluations required since this factor is important for problems with the complicated objective functions expected to be encountered in the QRGTS. Observations from the results shown in Table IV-3 include:

1) Powell's method was superior in almost all unconstrained cases, as was expected from a review of References 2 and 8.
2) Accelerated Convergence approached the minimum faster than PMS but PMS terminated sooner.
3) PMS with New Block was generally worse than without it.
4) Powell's rate of convergence is definitely a function of the MINC and ESCALE settings. It appears that MINC in addition to ESCALE should be set to reasonably high values.
5) PMS was generally better than VA04A on Colville's Test Problem #1 (a problem with constraints) since VA04A failed for MINC = 0.5 and ICON = 2 and had an excessive number of evaluations for MINC = .005.

Table IV-4 is included so Standard Time Ratios of several of the runs in Table IV - 3 can be compared with the results in Colville's Nonlinear Programming Study summary. All three rank in the top half of his listings with .0196 ranking 5th out of 24 methods. The Standard Time Ratio is an index developed by Colville (Ref. 6) to compare optimizer efficiencies independently of computer speeds.

Table IV-5 is comparisons between UNIVAR and EMR when used with PMS on CUBE. These results, along with another run not shown, were the determining factors in the decision to further develop UNIVAR rather than

EMR. For CUBE, EMR has lower OCOUNTS but has a U(2) error of 2% in Case 1, and its running time is longer on the average.

6. FORTRAN IV LISTINGS

```
    B
        MAINOP      -  EFN   SOURCE STATEMENT  -  IFN(S)  -

 C BOB COFER 3/22/67
 C MAIN PROGRAM FOR DSOP
       DIMENSION UE(10),BVBS(20)
       COMMON N,UA(10),UB(10),PA,PB,U(10),P,PF,UF(10),LC11
     C        ,BVA(10),K1,K2,K3,K4,COUNT,STE(10),MINC,LC1,LC3
     C          ,LCCONV,XCOUNT,E,LC17,LC8,STF(20),STG(20),PFCON
       COMMON /FUNCT/ LC5,LC14,OCOUNT,BIAS,NX,LC15
       COMMON /UNIV/ K11,UCOUNT,UMIN(20),UMAX(20),BVB(20),ILIMI,ILIM
       COMMON /VAO/ ESCALE,IPRINT,ICON,LCPF
       COMMON /PLOT/ US(20,500),NPT,PAS(500),PBS(500)
       COMMON /PENF/ EX(10),DELE(10),F(10),OFV,PFV
       REAL MINC(10),K1,K2,K3,K4,K11,NX(10),K5
     1 FORMAT (24I3)
     2 FORMAT (9E8.5)
     3 FORMAT (//(5E20.8))
     4 FORMAT (1H ///32HO LIST OF INPUT PARAMETER VALUES)
     5 FORMAT (1H ///17HO TIME(SECONDS) =,1E20.8)
     6 FORMAT (1H1//43HO DIRECT SEARCH OPTIMIZATION PROGRAM (DSOP))
     7 FORMAT (1H //25HO LOGICAL CHOICE SETTINGS)
     8 FORMAT (1H /34HO LC1 (NEGATIVE IS FAILURE EXIT) =,I1)
    81 FORMAT      (42HO LC4 (TYPE OF BASIC SEARCH - PATTERN=3) =,I1)
    82 FORMAT     (39HO LC5 (SELECTS AN OBJECTIVE FUNCTION) =,I1)
    83 FORMAT      (45HO LC6 (SELECTS UNIVAR(2) OR EMR WHEN LC4=3) =,I1)
    84 FORMAT (1H /111HO   N                  UB                 MINC
     C   STE                BVA                UMIN               UMAX         )
    85 FORMAT (7E16.8)
    86 FORMAT (1H //7H  K1  =E15.8,7H  K2  =E15.8,7H   K3 =E15.8,7H  K4
     C =E15.8,7H  K5  =E15.8/1X,8H COUNT = E15.8,7H BIAS =E15.8,7H  K11
     C=E15.8,9H ESCALE =E15.8,8H PFCON =E15.8)
    87 FORMAT (1H /26HO THE INITIAL VALUE OF P =,E15.8)
    88 FORMAT (1H ///41HO BEGINNING OF DIRECT SEARCH CALCULATIONS)
    89 FORMAT (33HO SEARCH CONVERGED - BIAS CONTROL)
    91 FORMAT (36HO SEARCH TERMINATED - COUNT EXCEEDED)
    92 FORMAT      (31HO LC7 (POSITIVE FOR PLOTTING) =,I1)
    93 FORMAT      (32HO LC8 (POSITIVE FOR NEW BLOCK) =,I1)
    94 FORMAT      (33HO IPRINT (POWELL PRINT CONTROL) =,I1)
    95 FORMAT      (33HO ICON (POWELL RESTART CONTROL) =,I1)
    96 FORMAT      (46HO IPENFK (NUMBER OF PENALTY FUNCTION CYCLES) =,I2)
    11 RTIM = CLOCKF(QQQ)                                                      2
       XCOUNT = 0.
       LCCONV = 0
       UCOUNT = 0
       OCOUNT = 0.
       LC11 = 0
       LC14 = 0
       LC15 = 2
       LC17 = 0
       ILIM = 0
       NPT = 0
       IPENF = 0
       LCPF = 0
       READ (5,1)  N,LC1,LC4,LC5,LC6,LC7,LC8,IPRINT,ICON,IPENFK               11
       READ (5,2) K1,K2,K3,K4,K5 ,COUNT,BIAS,K11,ESCALE,PFCON                 22
       DO 99 I = 1,N
    99 READ(5,2) NX(I),UB(I),MINC(I),STE(I),BVA(I),UMIN(I),UMAX(I)             26
```

```
     B
        MAINOP      -  EFN   SOURCE STATEMENT  -  IFN(S)  -

          WRITE (6,6)                                                    35
          WRITE (6,4)                                                    36
          WRITE (6,7)                                                    37
          WRITE (6,8) LC1                                                38
          WRITE (6,81) LC4                                               39
          WRITE (6,82) LC5                                               40
          WRITE (6,83) LC6                                               41
          WRITE (6,92) LC7                                               42
          WRITE (6,93) LC8                                               43
          WRITE (6,94) IPRINT                                            44
          WRITE (6,95) ICON                                              45
          WRITE (6,96) IPENFK                                            46
          WRITE (6,86) K1,K2,K3,K4,K5 ,COUNT,BIAS,K11,ESCALE,PFCON       47
   64     WRITE (6,84)                                                   48
          WRITE (6,85) (NX(I),UB(I),MINC(I),STE(I),BVA(I),UMIN(I),UMAX(I),
        C I = 1,N)                                                       49
          NC = N + 1
          DO 55 I = NC,10
   55     U(I) = 0.
C INITIALIZE PB
          IF (LC5 .EQ. 3) CALL OBJECT                                    71
          DO 10 I = 1,N
   10     U(I) = UB(I)
          CALL BOUNDU                                                    81
          DO 9 I = 1,N
    9     UB(I) = U(I)
          CALL OBJECT                                                    91
          PB = P
          WRITE (6,88)                                                   93
          WRITE (6,87)    PB                                             94
   50     DO 51 I = 1,N
   51     STF(I) = MINC(I)
          IF (LC4 .EQ. 3) GO TO 17
          IF (LC4 .EQ. 4) GO TO 52
          IF (LC4 .EQ. 5) GO TO 61
   12     CALL EXPLOR                                                    114
          IF (LC4 .EO.2) GO TO 15

          PB = PA
          DO 13 I = 1,N
   13     UB(I) = UA(I)
   14     XCOUNT = XCOUNT + 1.
          IF ((XCOUNT .GE. COUNT) .OR. (LC3 .LT. 0))GO TO 44
          GO TO 12
C ONE DIMENSIONAL SEARCH
   15     CALL ONEDS                                                     136
          PB = PF
          DO 16 I = 1,N
   16     UB(I) = UF(I)
          GO TO 14
C PATTERN SEARCH
   17     DO 18 I = 1,N
   18     UE(I) = UB(I)
          LC 11 = 0
          IFIN = 0
   20     LC1 = +1
```

```
      B
          MAINOP      -  EFN   SOURCE STATEMENT  -  IFN(S)  -
        IF (LC6 .EQ. 2) GO TO 24
        CALL EXPLOR                                                         162
        GO TO 25
     24 CALL UNIVAR                                                         165
     25 IF (LCCONV .EQ. 2) GO TO 43
        IF (LCCONV .EQ. 3) GO TO 24
C EXIT TEST FOR SUCCESSIVE TRIALS WITH MINUTE IMPROVEMENT
        LC1 = -1
     21 IF ((PA-PE+BIAS) .LT. 0.) IFIN = 0
        IF (IFIN .GE. 5) GO TO 41
        IFIN = IFIN + 1
        DO 27 I = 1,N
     27 BVBS(I) = BVB(I)
        DO 19 I = 1,N
     19 U(I) = UE(I) + K5*(UA(I) - UE(I))
        CALL BOUNDU                                                         198
        CALL OBJECT                                                         200
        DO 23 I = 1,N
        UE(I) = UA(I)
     23 UB(I) = U(I)
        PE = PA
        PB = P
        IF (LC6 .EQ. 2) GO TO 31
        CALL EXPLOR                                                         215
        GO TO 32
     31 CALL UNIVAR                                                         218
     32 IF (XCOUNT .GE. COUNT) GO TO 42
        IF (PA .LT. PE) GO TO 21
        PB = PE
        DO 22 I = 1,N
        BVB(I) = BVBS(I)
     22 UB(I) = UE(I)
        GO TO 20
     41 WRITE (6,89)                                                        236
        GO TO 43
     42 WRITE (6,91)                                                        238
        GO TO 44
C TEST FOR PENALTY FUNCTION ITERATION
     43 IPENF = IPENF + 1
        WRITE (6,3 ) EX,DELE,F,OFV,PFV                                      241
        CALL OBJECT                                                         245
        WRITE (6,3 ) EX,DELE,F,OFV,PFV                                      246
        LCPF = 2
        DO 62 I = 1,10
        UB(I) = U(I)
     62 DELE(I) = F(I)
     44 RTIM2 = CLOCKF(QQQ)                                                 262
        TIME = (RTIM2 - RTIM) * .6
        WRITE (6,5)  TIME                                                   263
        IF ((IPENF - IPENFK) .LT. 0) GO TO 64
        IF (LC7 .GT. 0) CALL PLOTOP                                         268
        GO TO 11
C EXPERIMENT WITH ACCELERATED CONVERGENCE
     52 CALL UNIVAR                                                         272
        DO 53 I = 1,N
        UB(I) = UA(I)
      B
         MAINOP     -  EFN   SOURCE STATEMENT  -  IFN(S)  -
      IF (LC6 .EQ. 2) GO TO 24
      CALL EXPLOR                                                         162
      GO TO 25
   24 CALL UNIVAR                                                         165
   25 IF (LCCONV .EQ. 2) GO TO 43
      IF (LCCONV .EQ. 3) GO TO 24
C EXIT TEST FOR SUCCESSIVE TRIALS WITH MINUTE IMPROVEMENT
      LC1 = -1
   21 IF ((PA-PE+BIAS) .LT. 0.) IFIN = 0
      IF (IFIN .GE. 5) GO TO 41
      IFIN = IFIN + 1
      DO 27 I = 1,N
   27 BVBS(I) = BVB(I)
      DO 19 I = 1,N
   19 U(I) = UE(I) + K5*(UA(I) - UE(I))
      CALL BOUNDU                                                         198
      CALL OBJECT                                                         200
      DO 23 I = 1,N
      UE(I) = UA(I)
   23 UB(I) = U(I)
      PE = PA
      PB = P
      IF (LC6 .EQ. 2) GO TO 31
      CALL EXPLOR                                                         215
      GO TO 32
   31 CALL UNIVAR                                                         218
   32 IF (XCOUNT .GE. COUNT) GO TO 42
      IF (PA .LT. PE) GO TO 21
      PB = PE
      DO 22 I = 1,N
      BVB(I) = BVBS(I)
   22 UB(I) = UE(I)
      GO TO 20
   41 WRITE (6,89)                                                        236
      GO TO 43
   42 WRITE (6,91)                                                        238
      GO TO 44
C TEST FOR PENALTY FUNCTION ITERATION
   43 IPENF = IPENF + 1
      WRITE (6,3 ) EX,DELE,F,OFV,PFV                                      241
      CALL OBJECT                                                         245
      WRITE (6,3 ) EX,DELE,F,OFV,PFV                                      246
      LCPF = 2
      DO 62 I = 1,10
      UB(I) = U(I)
   62 DELE(I) = F(I)
   44 RTIM2 = CLOCKF(QQQ)                                                 262
      TIME = (RTIM2 - RTIM) * .6
      WRITE (6,5)  TIME                                                   263
      IF ((IPENF - IPENFK) .LT. 0) GO TO 64
      IF (LC7 .GT. 0) CALL PLOTOP                                         268
      GO TO 11
C EXPERIMENT WITH ACCELERATED CONVERGENCE
   52 CALL UNIVAR                                                         272
      DO 53 I = 1,N
      UB(I) = UA(I)
      B
          MAINOP      -  EFN   SOURCE STATEMENT  -  IFN(S)  -

   53  STF(I) = STG(I)
       PB = PA
       IF (LCCONV .EQ.2) GO TO 43
       GO TO 52
C POWELL SUBROUTINE
   61  CALL VA04A
       GO TO 43
       END
```

```
      B
          NEWB1       -  EFN   SOURCE STATEMENT  -  IFN(S)  -

C MODIFIED POWELL APPROACH   H. ROBBINS, R. COFER
       SUBROUTINE NEWB
       DIMENSION UT(20)
       COMMON N,UA(10),UB(10),PA,PB,U(10),P,PF,UF(10),LC11
     C        ,BVA(10),K1,K2,K3,K4,COUNT,STE(10),MINC,LC1,LC3
     C          ,LCCONV,XCOUNT,E,LC17,LC8,STF(20),STG(20)
           COMMON /UNIV/ K11,UCOUNT,UMIN(20),UMAX(20),BVB(20),ILIMI,ILIM
       REAL K11,MINC(10),K1,K2,K3,K4
       PT = PB
       DO 11 I=1,N
       UT(I) = UB(I)
   11  U(I) = UA(I) + STF(I)
       CALL OBJECT
       IF (P .GE. PA) GO TO 10
       CALL SETQ
       GO TO 9
   10  DO 13 I=1,N
   13  UB(I) = U(I)
       PB = P
    9  CALL ONEDS
       PB = PT
       PA = PF
       DO 14 I=1,N
       UB(I) = UT(I)
   14  UA(I) = UF(I)
       CALL ONEDS
       PA = PF
       DO 15 I=1,N
       UA(I) = UF(I)
   15  STG(I) = .5*(UA(I) - UB(I))
       RETURN
       END
      B
        VA04        -  EFN   SOURCE STATEMENT  -  IFN(S)  -

C BOB COFER 3/22/67
C POWELL PROGRAM CONVERTED TO FORTRAN 4 WITH MINOR CHANGES FOR P.F. CYCLES
      SUBROUTINE VA04A
      DIMENSION W(40 ),X(10),E(10)
      EQUIVALFNCE (X(1),U(1)),(E(1),MINC(1)),(F,P)
      COMMON N,UA(10),UB(10),PA,PB,U(10),P,PF,UF(10),LC11
     C       ,BVA(10),K1,K2,K3,K4,COUNT,STE(10),MINC,LC1,LC3
      COMMON /VA0/ ESCALE,IPRINT,ICON,LCPF
      REAL MINC(10),K1,K2,K3,K4,K11,NX(10),K5
  511 FORMAT (5E16.8)
      IF (LCPF .EQ. 2) GO TO 211
      MAXIT = COUNT
      DDMAG=0.1*ESCALE
      SCER=0.05/ESCALE                                                  VA04A006
      JJ=N*N+N                                                          VA04A007
      JJJ=JJ+N                                                          VA04A008
      K=N+1                                                             VA04A009
      NFCC=1                                                            VA04A010
      IND=1                                                             VA04A011
      INN=1                                                             VA04A012
      DO 1 I=1,N                                                        VA04A013
      DO 2 J=1,N                                                        VA04A014
      W(K) = 0.
      IF(I-J)4,3,4                                                      VA04A016
    3 W(K)=ABS (E(I))                                                   VA04A017
      W(I)=ESCALE                                                       VA04A018
    4 K=K+1                                                             VA04A019
    2 CONTINUF                                                          VA04A020
    1 CONTINUF                                                          VA04A021
      ITERC=1                                                           VA04A022
      ISGRAD=2                                                          VA04A023
      GO TO 212
  211 IND = 2
  212 FKEEP=ABS (F)+ABS (F)                                             VA04A025
    5 ITONE=1                                                           VA04A026
      FP=F                                                              VA04A027
      SUM=0.
      IXP=JJ                                                            VA04A029
      DO 6 I=1,N                                                        VA04A030
      IXP=IXP+1                                                         VA04A031
      W(IXP)=X(I)                                                       VA04A032
    6 CONTINUF                                                          VA04A033
      IDIRN=N+1
      ILINE=1                                                           VA04A035
    7 DMAX=W(ILINE)
      DACC=DMAX*SCER                                                    VA04A037
      DMAG=AMIN1 (DDMAG,0.1*DMAX)
      DMAG=AMAX1(DMAG,20.*DACC)
      DDMAX=10.*DMAG                                                    VA04A040
      GO TO (70,70,71),ITONE                                            VA04A041
   70 DL=0.                                                             VA04A042
      D=DMAG                                                            VA04A043
      FPREV=F                                                           VA04A044
      IS=5                                                              VA04A045
      FA=F                                                              VA04A046
      B
         VA04          -  EFN   SOURCE STATEMENT  -  IFN(S)  -

        DA=DL                                                           VA04A047
    8   DD=D-DL                                                         VA04A048
        DL=D                                                            VA04A049
   58   K=IDIRN                                                         VA04A050
        DO 9 I=1,N                                                      VA04A051
        X(I)=X(I)+DD*W(K)                                               VA04A052
        K=K+1                                                           VA04A053
    9   CONTINUE                                                        VA04A054
        CALL OBJECT                                                     VA04A055
        NFCC=NFCC+1                                                     VA04A056
        GO TO (10,11,12,13,14,96),IS                                    VA04A057
   14   IF(F-FA)15,16,24                                                VA04A058
   16   IF (ABS(D) - DMAX) 17,17,18
   17   D=D+D                                                           VA04A060
        GO TO 8                                                         VA04A061
   18   WRITE(6,19)                                                     VA04A062
   19   FORMAT(5X,44HVA04A MAXIMUM CHANGE DOES NOT ALTER FUNCTION)      VA04A063
        GO TO 20                                                        VA04A064
   15   FB=F                                                            VA04A065
        DB=D                                                            VA04A066
        GO TO 21                                                        VA04A067
   24   FB=FA                                                           VA04A068
        DB=DA                                                           VA04A069
        FA=F                                                            VA04A070
        DA=D                                                            VA04A071
   21   GO TO (83,23),ISGRAD                                            VA04A072
   23   D=DB+DB-DA                                                      VA04A073
        IS=1                                                            VA04A074
        GO TO 8                                                         VA04A075
   83   D=0.5*(DA+DB-(FA-FB)/(DA-DB))                                   VA04A076
        IS=4                                                            VA04A077
        IF((DA-D)*(D-DB))25,8,8
   25   IS=1                                                            VA04A079
        IF(ABS (D-DB)-DDMAX)8,8,26                                      VA04A080
   26   D=DB+SIGN (DDMAX,DB-DA)                                         VA04A081
        IS=1                                                            VA04A082
        DDMAX=DDMAX+DDMAX                                               VA04A083
        DDMAG=DDMAG+DDMAG                                               VA04A084
        IF(DDMAX-DMAX)8,8,27                                            VA04A085
   27   DDMAX=DMAX                                                      VA04A086
        GO TO 8                                                         VA04A087
   13   IF(F-FA)28,23,23                                                VA04A088
   28   FC=FB                                                           VA04A089
        DC=DB                                                           VA04A090
   29   FB=F                                                            VA04A091
        DB=D                                                            VA04A092
        GO TO 30                                                        VA04A093
   12   IF(F-FB)28,28,31
   31   FA=F
        DA=D                                                            VA04A096
        GO TO 30                                                        VA04A097
   11   IF(F-FB)32,10,10
   32   FA=FB
        DA=DB                                                           VA04A100
        GO TO 29                                                        VA04A101
   71   DL=1.
```

```
       B
          VA04          -  EFN   SOURCE STATEMENT  -  IFN(S)  -

       DDMAX=5.                                                        VA04A103
       FA=FP                                                           VA04A104
       DA=-1.                                                          VA04A105
       FB=FHOLD                                                        VA04A106
       DB=0.                                                           VA04A107
       D=1.                                                            VA04A108
  10   FC=F
       DC=D                                                            VA04A110
  30   A=(DB-DC)*(FA-FC)
       B=(DC-DA)*(FB-FC)                                               VA04A112
       IF((A+B)*(DA-DC))33,33,34                                       VA04A113
  33   FA=FB
       DA=DB                                                           VA04A115
       FB=FC                                                           VA04A116
       DB=DC                                                           VA04A117
       GO TO 26                                                        VA04A118
  34   D=0.5*(A*(DB+DC)+B*(DA+DC))/(A+B)
       DI=DB                                                           VA04A120
       FI=FB                                                           VA04A121
       IF(FB-FC)44,44,43                                               VA04A122
  43   DI=DC
       FI=FC                                                           VA04A124
  44   GO TO (86,86,85),ITONE
  85   ITONE=2
       GO TO 45                                                        VA04A127
  86   IF (ABS (D-DI)-DACC) 41,41,93
  93   IF (ABS (D-DI)-0.03*ABS (D)) 41,41,45
  45   IF ((DA-DC)*(DC-D)) 47,46,46
  46   FA=FB
       DA=DB                                                           VA04A132
       FB=FC                                                           VA04A133
       DB=DC                                                           VA04A134
       GO TO 25                                                        VA04A135
  47   IS=2
       IF ((DB-D)*(D-DC)) 48,8,8                                       VA04A137
  48   IS=3
       GO TO 8                                                         VA04A139
  41   F=FI
       D=DI-DL                                                         VA04A141
       DD=SQRT ((DC-DB)*(DC-DA)*(DA-DB)/(A+B))                         VA04A142
       DO 49 I=1,N                                                     VA04A143
       X(I)=X(I)+D*W(IDIRN)                                            VA04A144
       W(IDIRN)=DD*W(IDIRN)                                            VA04A145
       IDIRN=IDIRN+1                                                   VA04A146
  49   CONTINUE
       W(ILINE)=W(ILINE)/DD                                            VA04A148
       ILINE=ILINE+1                                                   VA04A149
       IF(IPRINT-1)51,50,51                                            VA04A150
  50   WRITE(6,52)ITERC,NFCC,F,(X(I),I=1,N)
  52   FORMAT (/1X,9HITERATION,I5,I15,16H FUNCTION VALUES,
     1 10X,3HF =,E21.14/(5E24.14))
       GO TO(51,53),IPRINT                                             VA04A154
  51   GO TO (55,38),ITONE
  55   IF (FPREV-F-SUM) 94,95,95
  95   SUM=FPREV-F
       JIL=ILINE                                                       VA04A158
```

```
      8
          VA04         -  EFN    SOURCE STATEMENT  -  IFN(S)  -
  94  IF (IDIRN-JJ) 7,7,84
  84  GO TO (92,72),IND
  92  FHOLD=F
      IS=6                                                          VA04A162
      IXP=JJ                                                        VA04A163
      DO 59 I=1,N                                                   VA04A164
      IXP=IXP+1                                                     VA04A165
      W(IXP)=X(I)-W(IXP)                                            VA04A166
  59  CONTINUE
      DD=1.                                                         VA04A168
      GO TO 58                                                      VA04A169
  96  GO TO (112,87),IND
 112  IF (FP-F) 37,37,91
  91  D=2.*(FP+F-2.*FHOLD)/(FP-F)**2
      IF (D*(FP-FHOLD-SUM)**2-SUM) 87,37,37                         VA04A173
  87  J=JIL*N+1
      IF (J-JJ) 60,60,61                                            VA04A175
  60  DO 62 I=J,JJ
      K=I-N                                                         VA04A177
      W(K)=W(I)                                                     VA04A178
  62  CONTINUE
      DO 97 I=JIL,N                                                 VA04A180
      W(I-1)=W(I)                                                   VA04A181
  97  CONTINUE
  61  IDIRN=IDIRN-N
      ITONE=3                                                       VA04A184
      K=IDIRN                                                       VA04A185
      IXP=JJ                                                        VA04A186
      AAA=0.                                                        VA04A187
      DO 65 I=1,N                                                   VA04A188
      IXP=IXP+1                                                     VA04A189
      W(K)=W(IXP)                                                   VA04A190
      IF (AAA-ABS (W(K)/E(I))) 66,67,67                             VA04A191
  66  AAA=ABS (W(K)/E(I))
  67  K=K+1
  65  CONTINUE
      DDMAG=1.                                                      VA04A195
      W(N)=ESCALE/AAA                                               VA04A196
      ILINE=N                                                       VA04A197
      GO TO 7                                                       VA04A198
  37  IXP=JJ
      AAA=0.                                                        VA04A200
      F=FHOLD                                                       VA04A201
      DO 99 I=1,N                                                   VA04A202
      IXP=IXP+1                                                     VA04A203
      X(I)=X(I)-W(IXP)                                              VA04A204
      IF (AAA*ABS (E(I))-ABS (W(IXP))) 98,99,99                     VA04A205
  98  AAA=ABS (W(IXP)/E(I))
  99  CONTINUE
      GO TO 72                                                      VA04A208
  38  AAA=AAA*(1.+DI)
      GO TO (72,106),IND                                            VA04A210
  72  IF (IPRINT-2) 53,50,50
  53  GO TO (109,88),IND
 109  IF (AAA-0.1) 89,89,76
  89  GO TO (20,116),ICON
```

```
       B
           VA04        -  EFN   SOURCE STATEMENT  -  IFN(S)  -

  116  IND=2
       GO TO (100,101),INN                                            VA04A216
  100  INN=2
       K=JJJ                                                          VA04A218
       DO 102 I=1,N                                                   VA04A219
       K=K+1                                                          VA04A220
       W(K)=X(I)                                                      VA04A221
       X(I)=X(I)+10.*E(I)                                             VA04A222
  102  CONTINUE
       FKEEP=F                                                        VA04A224
       CALL OBJECT                                                    VA04A225
       NFCC=NFCC+1                                                    VA04A226
       DDMAG=0.                                                       VA04A227
       GO TO 108                                                      VA04A228
   76  IF (F-FP) 35,78,78
   78  WRITE(6,80)
   80  FORMAT (5X,37HVA04A ACCURACY LIMITED BY ERRORS IN F)
       GO TO 20                                                       VA04A232
   88  IND=1
   35  DDMAG=0.4*SQRT (FP-F)
       ISGRAD=1                                                       VA04A235
  108  ITERC=ITERC+1
       IF (ITERC-MAXIT) 5,5,81                                        VA04A237
   81  WRITE (6,82) MAXIT
   82  FORMAT(I5,30H ITERATIONS COMPLETED BY VA04A)
       IF (F-FKEEP) 20,20,110                                         VA04A240
  110  F=FKEEP
       DO 111 I=1,N                                                   VA04A242
       JJJ=JJJ+1                                                      VA04A243
       X(I)=W(JJJ)                                                    VA04A244
  111  CONTINUE
       GO TO 20                                                       VA04A246
  101  JIL=1
       FP=FKEEP                                                       VA04A248
       IF (F-FKEEP) 105,78,104                                        VA04A249
  104  JIL=2
       FP=F                                                           VA04A251
       F=FKEEP                                                        VA04A252
  105  IXP=JJ
       DO 113 I=1,N                                                   VA04A254
       IXP=IXP+1                                                      VA04A255
       K=IXP+N                                                        VA04A256
       GO TO (114,115),JIL                                            VA04A257
  114  W(IXP)=W(K)
       GO TO 113                                                      VA04A259
  115  W(IXP)=X(I)
       X(I)=W(K)                                                      VA04A261
  113  CONTINUE
       JIL=2                                                          VA04A263
       GO TO 92                                                       VA04A264
  106  IF (AAA-0.1) 20,20,107
   20  RETURN
  107  INN=1
       GO TO 35                                                       VA04A268
       END
```

```
C H. ROBBINS, R. COFER 3/22/67
      SUBROUTINE UNIVAR
C BEST UNIVAR DIRECT SEARCH PROGRAM - WITH PROVISIONS FOR MIN-MAX CONSTRAINTS
      COMMON N.UA(10),UB(10),PA,PB,U(10),P,PF,UF(10),LC11
     C        .BVA(10).K1.K2.K3.K4.COUNT,STE(10).MINC.LC1.LC3
     .C           .LCCONV,XCOUNT,E,LC17,LC8,STF(20),STG(20)
      COMMON /UNIV/ K11,UCOUNT,UMIN(20),UMAX(20),BVB(20).ILIMI.ILIM
      REAL K11.MINC(10).K1,K2,K3.K4
    4 FORMAT (//(5E18.8))
      PA = PB
      UCOUNT = 0
      IF (LC1 .GE. 0) GO TO 9
      DO 8 I = 1.N
    8 STF(I) = STG(I)
    9 LC3 = -2
      DO 11 I = 1,N
   11 U(I) = US(I)
C START INNER LOOP
      DO 29 I = 1,N
      IF (BVA(I) .EQ. 0.)GO TO 29
      IF (BVD(I) .EQ. 0.) GO TO 88
      IF (LC1 .LT. 0) GO TO 29
      UT = U(I)
      U(I) = U(I) + STE(I)
      CALL OBJECT
      IF (P .LT. PA) GO TO 86
      GO TO 99
   86 BVB(I) = 0
   87 PT = PA
      PA = P
      DEL2 = STE(I)
      GO TO 90
   88 LC12 = 2
      GO TO 89
   90 STE(I) = K11*STE(I)
      LC12 = -2
   89 UT = U(I)
      U(I) = U(I) + STE(I)
      DEL = UMIN(I) - U(I)
      IF (DEL .GE. 0.)GO TO 94
      DEL = UMAX(I) - U(I)
      IF (DEL  .LE. 0.)GO TO 93
      CALL OBJECT
      IF((P - PA) .LT. 0.) GO TO 87
   92 DEL1 = U(I) - UT
      U(I) = UT
      IF (LC12 .LT. 0) GO TO 98
      PT = P
      STE(I) = -STE(I)
      DEL2 = -DEL1
      GO TO 90
   93 U(I) = UMAX(I)
      CT = - MINC(I)
      GO TO 95
   94 U(I) = UMIN(I)
```

```
          B
            UNIVA        -   EFN    SOURCE STATEMENT    -   IFN(S)   -
          DT = MINC(I)
      95  CALL OBJECT
          IF ((P-PA) .GE. 0.)  GO TO 92
          PA = P
          BVB(I) = 1
          STE(I) = DT*4.
          GO TO 29
      98  PMPA = P - PA
          PTMPA = PT - PA
          E =(.5*((DEL2**2)*PMPA -(DEL1**2)*PTMPA))/(DEL2*PMPA +DEL1*PTMPA)
          U(I) = U(I) - E
          CALL OBJECT
          STE(I) = STE(I)/ K11
          IF ((P-PA) .LT. 0.) GO TO 71
      99  STE(I) = STE(I)/2.
          U(I) = UT
          GO TO 72
      71  PA = P
      72  DEL = ABS(STE(I))- MINC(I)
          IF (DEL .GT. 0.) LC3 = 2
          IF (DEL .LT. 0.) STE(I) = SIGN(MINC(I),STE(I))
      29  CONTINUE
C END OF INNER LOOP
          DO 31 I = 1,N
      31  UA(I) = U(I)
          IF (LC8 .GT. 0) GO TO 51
      32  LCCONV = 0
          CALL SUCCES
          IF (LCCONV .EQ. 4) GO TO 9
          RETURN
C PROVISION FOR NEW BLOCK - ACCELERATED CONVERGENCE
      51  CALL NEWB
          GO TO 32
          END
```

```
         B
             EXPL        -  EFN    SOURCE STATEMENT   -   IFN(S)  -

C EXPLORATORY MOVE ROUTINE - H. ROBBINS, R. COFER
        SUBROUTINE EXPLOR
        DIMENSION STF(10)
        COMMON N,UA(10),UB(10),PA,PB,U(10),P,PF,UF(10),LC11
      C        ,BVA(10),K1,K2,K3,K4,COUNT,STE(10),MINC,LC1,LC3
      C          ,LCCONV,XCOUNT
        REAL MINC(10),K1,K2,K3,K4
     4  FORMAT (//(5E18.8))
    11  LC3 = -1
        SUM = 0.
        PA = PB
        DO 12 I = 1,N
        U(I) = UB(I)
    12  STF(I) = 0.
C START OF FIGURE 1
        DO 29 I = 1,N
        IF (BVA(I) .EG. 0.) GO TO 29
        U(I) = U(I) + STE(I)
        CALL OBJECT
        IF (P - PA) 13,14,14
    13  STE(I) = K3*STE(I)
    15  PA = P
        LC3 = +1
        GO TO 29
    14  PT = P
        U(I) = U(I) - 2.*STE(I)
        CALL OBJECT
        IF (P - PA) 16,18,18
    16  STE(I) = -STE(I)
        GO TO 15
    18  U(I) = U(I) + STE(I)
        PMPT = P - PT
        PPPT = P + PT
        PPPTPA = PPPT - 2.*PA
        STF(I) = K1* PMPT *STE(I)/ PPPTPA
        QQ = SIGN(1.,PMPT)
        IF (PMPT .EQ. 0.)  QQ = 1.
        STE(I) = K2*STE(I)*QQ
        SUM = SUM + (PMPT**2/PPPTPA)
        DELTA    = ABS(STE(I)) - MINC(I)
        IF (DELTA) 21,21,19
    19  LC3 = +1
        GO TO 29
    21  STE(I) = MINC(I)*SIGN(1.,STE(I))
    29  CONTINUE
C END OF INNER LOOP
        WRITE (6,4) U,STE,STF,P,PA,PT,SUM
C END OF FIGURE 1
        DELTA = PA - PB + K4*SUM
        IF (DELTA) 31,33,33
    31  DO 32 I = 1,N
    32  UA(I) = U(I)
        LCCONV = 1
        GO TO 41
    33  DO 34 I = 1,N
```

```
      B
          EXPL         -  EFN   SOURCE STATEMENT , -  IFN(S)  -
34   U(I) = U(I) + STF(I)
     CALL OBJECT
     IF (P - PA) 35.37.37
35   PA = P
     DO 36 I = 1.N
36   UA(I) = U(I)
     LCCONV = 1
     GO TO 41
37   DO 38 I = 1.N
38   UA(I) = U(I) - STF(I)
     LCCONV = 0
41   CALL SUCCES
     IF (LCCONV .NE. 4) RETURN
43   DO 44 I = 1.N
44   STE(I) = .5*STE(I)
     GO TO 11
     END

C BOB COFER 3/22/67
C WRITES FINAL CONDITIONS AND STORES PLOTTING INFORMATION
         SUBROUTINE SUCCES
         COMMON N,UA(10),UB(10).PA.PB.U(10).P.PF.UF(10).LC11
      C        .BVA(10).K1.K2.K3.K4.COUNT.STE(10).MINC.LC1.LC3
      C           .LCCONV,XCOUNT.E
         COMMON /FUNCT/ LC5.LC14.OCOUNT.BIAS.NX.LC15
         COMMON /PLOT/ US(20.500).NPT.PAS(500).PBS(500)
         REAL K11.MINC(10).K1.K2.K3.K4.NX(10)
     1   FORMAT (15H0 SEARCH FAILED)
     2   FORMAT (16H0 SEARCH SUCCESS)
     3   FORMAT (40H0 SEARCH CONVERGED ' PA=PB AND LC3 MINUS)
     4   FORMAT (//(5E18.8))
     5   FORMAT(1H ///48H0 END OF ONE COMPLETE PASS THROUGH UNIVAR OR EMR)
     6   FORMAT      (7H    P   =E15.8.7H   PA   =E15.8.7H    PB =   E15.8.9H XCO
      CUNT =E15.8.9H OCOUNT =E15.8)
     7   FORMAT      (58H0     N                    U                   UB
      C       STE/(4E16.8))
         WRITE (6.5)
         IF (PA .LT. PB) GO TO 41
         IF (LC3 .LT. 0) GO TO 39
         LCCONV = 4
         IF (LC1 .GE. 0) GO TO 42
         WRITE (6.1)
         LCCONV = 3
         GO TO 42
    39   WRITE (6.3)
         LCCONV - 2
         GO TO 42
    41   WRITE (6.2)
         LCCONV = 1
    42   XCOUNT = XCOUNT + 1.
C STORE FOR PLOTTING
         NPT = NPT + 1
         DO 43 I = 1.N
    43   US(I.NPT) = U(I)
         PAS(NPT) = PA
         PBS(NPT) = PB
         WRITE (6.6)P.PA.PB,XCOUNT,OCOUNT
         WRITE (6.7) (NX(I).U(I).UB(I).STE(I). I = 1.N)
         RETURN
         END
```

```
       B
           ONED          -  EFN    SOURCE STATEMENT  -  IFN(S)  -

C  ONE DIMENSIONAL SEARCH PROGRAM  H. ROBBINS, R. COFER
       SUBROUTINE ONEDS
     1 FORMAT (//(5E20.8))
       COMMON N,UA(10),UB(10),PA,PB,U(10),P,PF,UF(10)
       REAL K
    10 DO 11 I = 1,N
    11 U(I) = UA(I) + 2.*(UA(I) - UB(I))
       CALL OBJECT
       IF (P .GE. PA) GO TO 9
       CALL SETQ
       GO TO 10
     9 PC = P
       DO 12 I = 1,N
    12 U(I) = UA(I) + (UA(I) - UB(I))
       CALL OBJECT
       IF (P .GE. PA) PC = P
       IF (P .LT. PA) CALL SETQ
       K = .5*(PB-PC)/(PB+PC-2.*PA)
       DO 13 I = 1,N
    13 U(I) = UA(I) + K*(UA(I) - UB(I))
       CALL OBJECT
       IF (P - PA) 16,14,14
    14 PF = PA
       DO 15 I = 1,N
    15 UF(I) = UA(I)
       GO TO 18
    16 PF = P
       DO 17 I = 1,N
    17 UF(I) = U(I)
    18 WRITE (6,1) UA,UB,U,UF,PA,PB,P,PF,K
       RETURN
       END
```

```
       B                                                                      12/31/64
           SETQ1         -  EFN    SOURCE STATEMENT  -  IFN(S)  -

C BOB COFER 3/22/67
C PERFORMS REPETITIVE SETTINGS OF A AND B QUANTITIES
       SUBROUTINE SETQ
       COMMON N,UA(10),UB(10),PA,PB,U(10),P,PF,UF(10)
       PB = PA
       PA = P
       DO 12 I = 1,N
       UB(I) = UA(I)
    12 UA(I) = U(I)
       RETURN
       END
```

```
C BOB COFER 3/22/67
C LC5 SELECTS ONE OF EIGHT TEST FUNCTIONS
      SUBROUTINE OBJECT
      DIMENSION EU1(5),EU2(5),R2MR1(5),R2(5),R1(5),V1(5),V2(5),V(5)
     C        ,R1CR2(5)
      DIMENSION W1(5),W2(5),E1(5),E2(5),U1(5),U2(5)
      DIMENSION AA(10,5),BB(10),CC(5,5),DD(5),EE(5)
      COMMON N,UA(10),UB(10),PA,PB,U(10),P,PF,UF(10),LC11
     C      ,BVA(10),K1,K2,K3,K4,CCUNT,STE(10),MINC,LC1,LC3
     C        ,LCCONV,XCOUNT,E,LC17,LC8,STF(20),STG(20),PFCON
      COMMON /FUNCT/ LC5,LC14,OCOUNT,BIAS,NX,LC15
      COMMON /PENF/ EX(10),DELE(10),F(10),OFV,PFV
      REAL MU,I1(5),I2(5),IN,MF,MINC(10),NX(10)
    1 FORMAT (9E8.5)
    2 FORMAT (1H /41HO THE OBJECTIVE FUNCTION = ORBIT TRANSFER)
    3 FORMAT (1H /85HO THE OBJECTIVE FUNCTION = CUBE = 100.*(U(2)-U(1)*
     C*3)**2 + (1.-U(1))**2 + 10.*U(3)**2)
    4 FORMAT (1H /63HO THE OBJECTIVE FUNCTION =(1.-U(1))**2 + ABS(U(2)-
     CU(1)**3)*100.)
    7 FORMAT (1H /49HO THE OBJECTIVE FUNCTION =F(U) +A*G(U) +B*G(U)**2)
    8 FORMAT (1H /48HO THE OBJECTIVE FUNCTION = U(1)**2 + 50.*U(2)**2)
    9 FORMAT (//(5E18.8))
   99 FORMAT (5E16.8)
C PRINT OBJECTIVE FUNCTION NAME
      IF (LC15 .EQ. 0) GO TO 10
      LC15 = 0
      IF (LC5 .EQ. 2) WRITE (6,3)
      IF (LC5 .EQ. 4) WRITE (6,4)
      IF (LC5 .EQ. 5) WRITE (6,7)
      IF (LC5 .EQ. 1) WRITE (6,8)
   10 OCOUNT = OCOUNT + 1.
      IF (LC5 .EQ. 2) GO TO 6
      IF (LC5 .EQ. 3) GO TO 21
      IF (LC5 .EQ. 4) GO TO 42
      IF (LC5 .EQ. 5) GO TO 51
      IF (LC5 .EQ. 6) GO TO 61
       IF (LC5 .EQ. 7) GO TO 71
      P = U(1)**2 + 50.*U(2)**2
      GO TO 5
    6 P = 100.*(U(2) - U(1)**3)**2 + (1. - U(1))**2 + 10.*U(3)**2
    5 IF ((P .GE. PA) .OR. (LC11 .EQ. 0)) GO TO 12
      CALL SETQ
      GO TO 13
   12 LC11 = 0
   13 RETURN
   21 IF (LC14 .EQ. 2) GO TO 30
C INITIALIZATION FOR ORBIT TRANSFER OBJECTIVE FUNCTION
      WRITE (6,2)
      LC14 = 2
      READ (5,1) WW1,EE1,P1,WW2,EE2,P2,IN
      WRITE (6,9) WW1,EE1,P1,WW2,EE2,P2,IN
      MU = 14.076539E15
      MF = 5280.
      FM = 1./5280.
      DR = .0174532925
     B
        OBJT         -  EFN   SOURCE STATEMENT  -  IFN(S)  -
     RD = 57.2957795
     WW1 = DR*WW1
     WW2 = DR*WW2
     P1 = MF*P1
     P2 = MF*P2
     PAR3 = SQRT(MU/P1)
     PAR4 = SQRT(MU/P2)
     IN = DR*IN
     CI = COS(IN)
     SI = SIN(IN)
     SWW1 = SIN(WW1)
     DO 41 I = 1,2
     STE(I) = DR*STE(I)
     UB(I) = DR*UB(I)
  41 MINC(I) = MINC(I)*DR
     STE(3) = MF*STE(3)
     UB(3) = MF*UB(3)
     MINC(3) = MF*MINC(3)
     W1(1) = 0.
     W1(2) = -SI
     W1(3) = CI
     W2(1) = 0.
     W2(2) = 0.
     W2(3) = 1.
     E1(1) = EE1*COS(WW1)
     E1(2) = SWW1*CI*EE1
     E1(3) = SWW1*SI*EE1
     E2(1) = EE2*COS(WW2)
     E2(2) = EE2*SIN(WW2)
     E2(3) = 0.
     GO TO 35
C START OF REPETITIVE SECTION
  30 SU1 = SIN(U(1))
     U1(1) = COS(U(1))
     U1(2) = SU1*CI
     U1(3) = SU1*SI
     U2(1) = COS(U(2))
     U2(2) = SIN(U(2))
     U2(3) = 0.
     VAR1 = P1/(1. + EE1*COS(U(1) - WW1))
     VAR2 = P2/(1. + EE2*COS(U(2) - WW2))
     VAR3 = SQRT(MU*U(3))
     VAR4 = SQRT(MU/U(3))
     DO 31 I = 1,3
     R1(I) = VAR1*U1(I)
     R2(I) = VAR2*U2(I)
     EU1(I) = E1(I) + U1(I)
     EU2(I) = E2(I) + U2(I)
  31 R2MR1(I) = R2(I) - R1(I)
     V1(1) = PAR3*(W1(2)*EU1(3) - W1(3)*EU1(2))
     V1(2) =-PAR3*(W1(1)*EU1(3) - W1(3)*EU1(1))
     V1(3) = PAR3*(W1(1)*EU1(2) - W1(2)*EU 1(1))
     V2(1) = PAR4*(W2(2)*EU2(3) - W2(3)*EU2(2))
     V2(2) =-PAR4*(W2(1)*EU2(3) - W2(3)*EU2(1))
     V2(3) = PAR4*(W2(1)*EU2(2) - W2(2)*EU2(1))
    R1CR2(1)=R1(2)*R2(3) - R1(3)*R2(2)
```

```
        B
            OBJT        -  EFN   SOURCE STATEMENT  -  IFN(S)  -
       R1CR2(2)=-R1(1)*R2(3) + R1(3)*R2(1)
        R1CR2(3) = R1(1)*R2(2) - R1(2)*R2(1)
        R1CR2M = SQRT(R1CR2(1)**2 + R1CR2(2)**2 + R1CR2(3)**2)
        DO 32 I = 1,3
   32   V(I) = VAR3*R2MR1(I)/R1CR2M
        DT =ARCOS(U1(1)*U2(1) + U1(2)*U2(2) + U1(3)*U2(3))
        Z = VAR4*TAN(DT/2.)
        DO 33 I = 1,3
        I1(I) = V(I) + Z*U1(I) - V1(I)
   33   I2(I) = V2(I) - (V(I) - Z*U2(I))
        P = SQRT(I1(1)**2 + I1(2)**2 + I1(3)**2)+ SQRT(I2(1)**2 + I2(2)**
      C 2 + I2(3)**2)
   35   RETURN
C TEST FUNCTION
   42   P =       (1. - U(1))**2 + ABS(U(2) - U(1)**3)*100.
        LC11 = 0
        RETURN
C PENALTY FUNCTION
   51   F = U(1) + U(2)
        A = .705968
        B = 50.
        G = U(1)**2 + U(2)**2 -1.
        P = F + A*G + B*G**2
        RETURN
   61   P = 100.*(U(2)-U(1)**2)**2 + (1.-U(1))**2 + 90.*(U(4)-U(3)**2)**2
      C + (1.-U(3))**2 + 10.1*((U(2)-1.)**2 + (U(4)-1.)**2) + 19.8*(U(2)-
      C1.)*(U(4)-1.)
        RETURN
C TEST PROBLEM NO. 1
   71   IF (LC17 .EQ. 2) GO TO 74
         LC17 = 2
        DO 72 I = 1,10
   72   READ (5,1)  (AA(I,J),J=1,5),BB(I)
        DO 73 I = 1,5
   73   READ (5,1) (CC(I,J),J=1,5)
        READ (5,1) DD
        READ (5,1) EE
        READ (5,1)DELE(3),DELE(4),DELE(5),DELE(6),DELE(9)
        WRITE (6,99) AA,BB,CC,DD,EE
C COMPUTE OBJECTIVE FUNCTION VALUE
   74   SCYY = 0.
        DO 77 J = 1,5
        SCY = 0.
        DO 76 I = 1,5
   76   SCY = CC(I,J)*U(I) + SCY
   77   SCYY = SCY*U(J) + SCYY
        EPDY = 0
        DO 78 J = 1,5
   78   EPDY = U(J)*(EE(J) + DD(J)*U(J)**2) + EPDY
        OFV = SCYY + EPDY
C PENALTY FUNCTION CALCULATION
        DO 81 I = 5,6
       EX(I) = 0.
        DO 79 J = 1,5
   79   EX(I) = AA(I,J)*U(J) +EX(I)
   81   F(I) = EX(I) - BB(I) + DELE(I)
```

```
B
    OBJT          -  EFN   SOURCE STATEMENT   -   IFN(S)   -

F(3) = 0.
F(4) = 0.
F(9) = 0.
PFV =PFCON*(F(3)**2 + F(4)**2 + F(5)**2 + F(6)**2 + F(9)**2)
P = OFV + PFV
RETURN
END
```

```
       B                                                                12/01/64
           BOUND        -  EFN   SOURCE STATEMENT   -   IFN(S)   -

C BOB COFER 3/22/67
C LIMITS EACH COMPONENT OF U TO ITS EXPLICIT CONSTRAINTS
      SUBROUTINE BOUNDU
      COMMON N,UA(10),UB(10),PA,PB,U(10),P,PF,UF(10),LC11
    C         ,BVA(10),K1,K2,K3,K4,COUNT,STE(10),MINC,LC1,LC3
      COMMON /UNIV/ K11,UCOUNT,UMIN(20),UMAX(20),BVB(20),ILIMI,ILIM
      DO 11 I = 1,N
   11 BVB(I) = 1.
      DO 48 I = 1,N
      DEL = UMIN(I) - U(I)
      IF (DEL) 43,46,46
   43 DEL = UMAX(I) - U(I)
      IF (DEL) 45,45,44
   44 BVB(I) =0.
      GO TO 48
   45 U(I) = UMAX(I)
      STE(I) = -ABS(STE(I))
      GO TO 48
   46 U(I) = UMIN(I)
      STE(I) = ABS(STE(I))
   48 CONTINUE
      RETURN
      END
```

```
        B
            PLOTO       -  EFN   SOURCE STATEMENT  -  IFN(S)  -

C BOB COFER 3/22/67
C PLOTS PA.PB. AND U COMPONENTS VS. UNIVAR PASSES FCR PATTERN SEARCH
      SUBROUTINE PLOTOP
      DIMENSION ULABEL(10),UPS(500).XAXIS(500)
      COMMON N,UA(10),UB(10),PA,PB,U(10).P,PF,UF(10).LC11
     C       .BVA(10).K1.K2.K3.K4.COUNT.STE(10).MINC.LC1.LC3
      COMMON /PLOT/ US(20,500),NPT,PAS(500),PBS(500)
      REAL IN
      DATA (ULABEL(I),I=1,10) /6HU(1)...6HU(2)...6HU(3)...6HU(4)...6HUC
     C5)...6HU(6)...6HU(7)...6HU(8)...6HU(9)...6HU(10)./
C SCALE X AXIS FOR ALL PLOTS
      IN = 0.
      DO 10 I = 1,NPT
      IN = IN + 1.
   10 XAXIS(I) = IN
    5 FORMAT (5E20.8)
      WRITE (6,5) XAXIS,PAS,PBS
      CALL PSCLE (XAXIS,NPT,10.,XMIN,XSCL)
C PLOT PA AND PB VS. COMPLETE PASSES THROUGH UNIVAR
C SET INITIAL PA AND PB = 0. TO AVOID EXTREMELY LARGE PB VALUES
      DO 12 I = 1,3
      PAS(I) = 0.
   12 PBS(I) = 0.
      CALL PSCLE (PAS,NPT,8.,PAMIN,PASCL)
      CALL PSCLE (PBS,NPT,8.,PBMIN,PBSCL)
      CALL PAXIS (0.,0.,23H COMPLETED EXPLORATORY MOVES,28,10.,0.,XMIN,
     CXSCL)
      CALL PAXIS (0.,0.,10H PA AND PB,10,10.,90.,PBMIN,PBSCL)
      CALL PLINE (XAXIS,PAS,NPT,1)
      CALL PLINE (XAXIS,PBS,NPT,1)
      CALL PLEND (14.,0)
C PLOT U COORDINATES VS. COMPLETE PASSES THROUGH UNIVAR
      DO 15 I = 1,N
      DO 11 J = 1,NPT
   11 UPS(J) = US(I,J)
      CALL PSCLE (UPS,NPT,8.,UMIN,USCL)
      CALL PAXIS (0.,0.,28H COMPLETED EXPLORATORY MOVES,28,10.,0.,XMIN,
     CXSCL)
      CALL PAXIS(0.,0.,ULABEL(I),6,8.,90.,UMIN,USCL)
      CALL PLINE (XAXIS,UPS,NPT,1)
   15 CALL PLEND (14.,0)
      CALL PLEND (1.,1)
      RETURN
      END
```

REFERENCE

1. Hooke,.R., and Jeeves, T., Direct Search Solution of Numerical and Statistical Problems, J. Assoc. Computing Mach. 8, 212-229, 1961.

2. Powell, M.J.D., An Efficient Way of Finding the Minimum of a Function of Several Variables Without Computing Derivatives, Computer Journal 7, 155-162, 1964.

3. Leon, A., A Comparison Among Eight Known Optimizing Procedures, in Recent Advances in Optimization Techniques, John Wiley, 1966.

4. Powell, M.J.D., Subroutine VA04A, Unpublished Notes, 1964, 8 pp.

5. Fletcher, R., Function Minimization Without Evaluating Derivatives a Review Computer Journal 8, 33-41 1965.

6. Colville, A.R. Non Linear Programming Study, Unpublished Notes, 1966, 35 pp.

7. McCue, G.A., Optimum Two-Impulse Orbital Transfer and Rendezvous between Inclined Eliptical Orbits, AIAA Journal, Vol. I, No. 8, 1963, pp 1865 - 1872.

Appendix V (Unclassified)

NONLINEAR PROGRAMMING: A REVIEW OF THE STATE-OF-THE-ART

1. INTRODUCTION

In recent years, there has been an increasingly widespread recognition that many important practical problems can be formulated as problems of constrained minimization or maximization. That is, many problems of design, control, resource allocation, etc., can be formulated as follows: find the vector argument $\underline{x}$ which minimizes (or maximizes) a given scalar function $f(\underline{x})$, subject to a given set of equality and/or inequality constraints. The function f is called the objective function. It is also known as the cost function, the performance index, the payoff function, etc.

If the function f is linear, and the constraints are linear equalities and/or inequalities, this is the general linear programming problem, for which theory and practice are highly developed. If the oojective function or the constraints, or both, are nonlinear, it is the general nonlinear programming problem, also known under various other names: function minimization, constrained minimization, mathematical programming, optimization, parameter optimization, design optimization, etc. These names have differing connotations and associations. "Nonlinear programming" is the term favored by mathematicians. "Mathematical programming" is a general expression which includes linear, nonlinear, and integer programming, and is principally used in the field of operations research. The other terms are mostly used by engineers. The terms "function minimization" and "parameter optimization" definitely imply that there are only a finite number of variables, so the minimization/optimization problem is a problem in the ordinary calculus, as distinguished from the calculus of variations. The other names are ambiguous in this respect; they may be used in a broad sense which includes both finite-dimensional and infinite-dimensional cases, but are more often used in a restricted sense which excludes the latter alternative.

The development of numerical methods for solution of nonlinear programming problems is still in a relatively primitive state, but is a very active area of research. Unfortunately, most of the research work has been strongly biased toward one or the other of two distinct approaches which have not, as yet, been satisfactorily combined. One of these approaches views nonlinear programming as a generalization of linear programming. In linear programming the constraints are everything; without constraints there would be no problem. Therefore, this approach leads to emphasis on handling medium to large numbers of constraints, by modifications of the highly successful algorithms for linear programming. The basic assumption made is that the objective and constraint functions are approximately linear over regions large enough to permit reasonable-size steps toward the solution. The other approach starts with consideration of difficult problems in unconstrained minimization (e. g., problems where the objective function has very narrow, curving valleys) and adds provisions for constraints as an afterthought, if at all. An important aspect of the research work based on this approach has been the development of methods with second-order (or at least, higher than first-order) convergence in the neighborhood of the solution. This is not only desirable in itself; it is almost a necessity in cases where the matrix of second partial derivatives of the objective function has a wide range of eigenvalues.

The constraint-oriented approach has been the basis of some highly successful, large scale optimization programs, such as POP (Ref. 1), COP (Ref. 2) and NDS (Ref. 3). These programs have been made highly versatile and efficient by sophisticated programming techniques, which may make them competitive even in applications where the other approach is more appropriate in principle. They are potentially inefficient in cases where very narrow valleys occur, but in practical problems of design optimization, process control, etc., such cases do not occur very frequently if reasonable care is used in problem formulation. Usually, the objective function and constraints are quite simple and well-behaved, so constraint-oriented optimization algorithms can be highly efficient. However, important exceptions are known to exist.

By combining the principles of both approaches, it should be possible to develop algorithms which handle general sets of constraints efficiently and have accelerated convergence even for unfavorable (but smooth) forms of the objective function, or the constraints, or both. Such a combination has not yet been demonstrated in a completely satisfactory form, but numerous investigators are working toward this goal, and significant progress has already been made. Major advances may therefore be expected during the next few years.

The review briefly describes the nonlinear programming problem and some of the principal techniques that have been proposed for its solution. Most of the discussion is restricted to the finite-dimensional case, but optimization in function space (i. e., the optimal control problem) is considered in one section. A brief Bibliography is included. It is limited to books, review articles, and technical papers which describe significant technical advances.

2. FORMULATIONS OF THE PROBLEM

The general nonlinear programming problem in n dimensions is to find an n-component argument vector $\underline{x}$ which is feasible (i. e., satisfies a given set of constraints) and minimizes the objective function $f(\underline{x})$ subject to these constraints. Two extreme special cases are those with no constraints (unconstrained minimization) and those where the constraints completely determine $\underline{x}$, so the nonlinear programming problem reduces to solving a set of nonlinear equations. There are many ways of expressing the constraints. One way is given by the relations

$$g_i(\underline{x}) = 0 \qquad 0 < i \le m_1 \tag{1a}$$

$$g_i(\underline{x}) \ge 0 \qquad m_1 < i \le m \tag{1b}$$

which give m_1 equality constraints and $(m-m_1)$ inequality constraints. This form will be used in most of the discussions below. Other forms may be better for computational use. Inequality constraints often occur in pairs, giving a limited range for some variable, and this can be exploited to reduce the number of separate relations. For example, in the optimization algorithm POP (Ref. 1) the constraints are stated in the form

$$\underline{a} \leq \underline{x} \leq \underline{b} \qquad (2a)$$

$$\underline{c} \leq \underline{y} \leq \underline{d} \qquad (2b)$$

where $\underline{a}$, $\underline{b}$, $\underline{c}$, $\underline{d}$ are constant vectors, and $\underline{y} = \underline{y}(x)$ is a vector function of $\underline{x}$, not necessarily of the same dimensionality. Another form, which appears different but is equivalent to either of the above formulations, is

$$\underline{a} \leq \underline{x} \leq \underline{b} \qquad (3a)$$

$$g_i(\underline{x}) = 0 \qquad 0 < i \leq m_2 \qquad (3b)$$

This formulation is used in Abadie's Reduced Gradient method (Ref. 4).

3. NECESSARY CONDITIONS FOR OPTIMALITY

If $f(\underline{x})$ possesses continuous first partial derivatives, a necessary condition for an unconstrained local minimum at $\underline{x}_o$ is

$$\text{grad } f(\underline{x}_o) = 0 \qquad (4)$$

The corresponding necessary conditions for constrained cases are the Kuhn-Tucker conditions (Ref. 5) which involve the function

$$F(\underline{x}, \underline{\lambda}) = f(\underline{x}) + \sum_{1}^{m} \lambda_i g_i(\underline{x}) \qquad (5)$$

The Kuhn-Tucker conditions are that the constraint relations given by Eq (1) hold, that the gradient of $F(\underline{x}, \underline{\lambda})$ with respect to $\underline{x}$ vanishes, and that for values of i corresponding to inequality constraints ($m_1 < i \leq m$) the multipliers λ_i satisfy

$$\lambda_i \leq 0 \qquad \text{if } g_i(\underline{x}_o) = 0 \qquad (6)$$

$$\lambda_i = 0 \qquad \text{if } g_i(\underline{x}_o) > 0$$

There are also second-order necessary conditions, if the relevant partial derivatives exist. For unconstrained minimization, the second-order necessary condition is that M(x), the matrix of second partial derivatives of f(x), be positive semidefinite at $\underline{x}_o$:

$$M(\underline{x}_o) \geq 0 \qquad (7)$$

The second-order necessary condition for constrained cases is less familiar. It is given by Hadley (Ref. 6, page 101-102) in the following form: Let M_F $(\underline{x}, \underline{\lambda})$ denote the matrix of second partials of $F(\underline{x}, \underline{\lambda})$ with respect to components of $\underline{x}$, and define $G(\underline{x})$ to be the m x n matrix whose ith now is the gradient of $g_i(\underline{x})$ if the ith constraint is active (holds as an equality) and is a row of zeros if the ith constraint is inactive (holds as a strict inequality). The latter alternative is possible only for inequality constraints. The necessary condition is that $\underline{d}^T M_F(\underline{x}_o, \underline{\lambda})\underline{d}$ be positive semidefinite for every vector $\underline{d}$ satisfying $G(\underline{x}_o)\underline{d} = 0$. An equivalent condition is that some m x m matrix N must exist such that

$$M_F^*(\underline{x}_o, \underline{\lambda}) = M_F(\underline{x}_o, \underline{\lambda}) + G^T(\underline{x}_o)\, NG(\underline{x}_o) \qquad (8)$$

is positive semidefinite. Without loss of generality, N may be assumed to be some multiple of the unit matrix. This necessary condition is related to the obvious fact that if $\underline{x}_o$ gives a constrained local minimum of $f(\underline{x})$, it must do the same for

$$f^*(\underline{x}) = f(\underline{x}) + \sum_{a.c.} g_i(\underline{x})\, N_{ij}\, g_j(\underline{x}) \qquad (9)$$

where N is positive semidefinite but otherwise arbitrary, and the summation is over constraints active at $\underline{x}_o$.

A sufficient condition for a strong local minimum is that the first-order conditions hold, and that M (in the unconstrained case) or M_F^* (in the constrained case) be positive definite, rather than semidefinite only.

4. THE NEWTON-RAPHSON METHOD

The Newton-Raphson method of solving a nonlinear programming problem is based on solving the first-order necessary conditions as a set of simultaneous equations and/or inequalities. For unconstrained cases, a Taylor expansion of $\underline{h}(\underline{x}) = \text{grad } f(\underline{x})$ gives

$$\underline{h}(\underline{x} + \Delta\underline{x}) \approx \underline{h}(x) + M(\underline{x})\, \Delta\underline{x} \qquad (10)$$

which suggests the Newton-Raphson iteration

$$\text{new } \underline{x} = \underline{x} - M^{-1}(\underline{x})\, \underline{h}(\underline{x}) \qquad (11)$$

This iteration converges rapidly in some neighborhood of the minimum. Unfortunately, it converges just as readily to stationary points which are not minima, and often fails to converge at all. Powell (Ref. 7) has noted that convergence can be improved by regarding the Newton-Raphson step as indicating only a direction of change of $\underline{x}$, and letting the magnitude of the step be determined by a one-dimensional search for the minimum of $f(\underline{x})$ along a line. However, he found cases in which this also fails, even if $f(\underline{x})$ has no stationary points other than the minimum.

If $M(\underline{x})$ is positive definite (as it must be, in the neighborhood of a strong local minimum) a sufficiently short move in the direction given by the Newton-Raphson correction necessarily decreases $f(\underline{x})$. The convergence difficulty arises if $M(\underline{x})$ has negative eigenvalues for some values of $\underline{x}$, enabling $f(\underline{x})$ to be increased or left unchanged. An obvious remedy is to modify the inverse matrix in Eq (11), forcing it to become positive definite if not already so. An elegant procedure for doing this was invented by Davidon (Ref. 8). Davidon's method will be discussed in Section 8 of this Appendix.

For constrained cases, the procedure is to assume that certain constraints are active and others are inactive, ignore the inactive constraints, and treat all active constraints as equality constraints. The assumed set of active constraints is justified (or corrected) after the fact. A Taylor expansion of the first-order conditions gives

$$0 = \begin{bmatrix} \underline{h}^* \\ \\ \underline{g} \end{bmatrix} + \begin{bmatrix} M_F^*(\underline{x}, \underline{\lambda}) & G^T(x) \\ \\ G(x) & 0 \end{bmatrix} \begin{bmatrix} \Delta \underline{x} \\ \\ \Delta \underline{\lambda} \end{bmatrix} \qquad (12)$$

where $\underline{g}$ is a column vector of the constraint functions $g_i(\underline{x})$, and

$$\underline{h}^*(\underline{x}, \underline{\lambda}) = \underline{h}(x) + G^T(\underline{x}) \left[\underline{\lambda} + 2N\underline{g}(\underline{x})\right] \qquad (13)$$

is the gradient, with respect to $\underline{x}$, of

$$F^*(\underline{x}, \underline{\lambda}) = f^*(x) + \underline{\lambda}^T \underline{g}(\underline{x}) \qquad (14)$$

The matrix N is to be chosen (if possible) to make M_F^* positive definite. Eq (12) suggests a Newton-Raphson iteration analogous to Eq (11). If this iteration converges, it converges to a constrained stationary point, which is not necessarily a constrained minimum unless M_F^* can be made positive semi-definite by some choice of N. Convergence to non-minimal stationary points can presumably be prevented, as in the unconstrained case, by modification of the matrix M_F^* to force positive-definiteness. However, no analog of Davidon's method, applicable to constrained cases, appears to have been published so far.

5. CLASSIFICATION OF METHODS

In spite of its attractive feature of rapid convergence near the solution, the Newton-Raphson method is generally unsatisfactory because of (a) need for second derivatives which may be difficult and/or expensive to compute, (b) convergence uncertainties, including the possibility of false convergence, and (c) in constrained cases, the need to guess which constraints are active, with tedious recomputations if the initial guess is wrong. Therefore, a large variety of other methods have been developed. These may be classified according to the way in which they acquire derivative information (analytical derivatives, numerical derivatives obtained by finite differences, or no explicit computation of derivatives) and according to their speed of convergence near the solution (first order, second order). As with most classification systems, these classifications are not entirely clear-cut. Some algorithms, for example, use analytic first derivatives but compute second derivatives by finite differences. Also, many of the algorithms loosely described as second order have not been proven to have true second-order convergence, although this convergence is faster than first order. The two classifications are related, since any method with superlinear convergence must make use of second derivatives or equivalent information. For applications where a large variety of different problems must be solved, algorithms which use no derivatives, or compute derivatives by finite differences, are generally to be preferred over algorithms which require analytic derivatives, since the latter require much more input information for specification of a new problem. Explicit computation of derivatives by finite differences, as compared to implicit use of derivative information from changes of function value, has the advantage that the differencing technique can be changed in the course of the computation, to provide increased accuracy when needed; the algorithm can use simple differences at first, and then go over to symmetric differences.

For applications where essentially the same problem is to be solved many times, with slightly different parameter values each time (as in on-line optimal control) superlinear convergence is highly desirable not only for economy of computation, but because it gives a more smoothly varying output.

6. HEURISTIC DIRECT-SEARCH METHODS

The two best-known methods which make no use of derivative information, and are purely heuristic in motivation, are Rosenbrock's method (Reference 9) and the "Direct Search" or "Pattern Move Search" method of Hooke and Jeeves (Reference 10). Numerous variants of both of these methods exist. Neither method, in its original form, included provisions for constraints, but ad hoc modifications for constrained problems have been proposed by several authors.

In Rosenbrock's method, the program remembers a set of orthogonal search directions and uses them in sequence. The relative success of the different directions, as measured by net displacements of the argument vector, is then used to generate a new set of search directions. The procedure is intended to make the search directions tend to align themselves with principal axes of the quasi-ellipsoids which are iso-value contours of the objective function near a minimum. This behavior has never been proved mathematically, but is supported by experimental evidence. Convergence is first-order, but usually reasonably rapid.

The basic idea of Pattern Move Search is that what succeeds once, should be tried again. Starting from an initial "base point", an exploratory subroutine generates another point with smaller objective function. This constitutes a new base point. The vector difference between the two most recent base points defines a "pattern move" which is applied to the newest base point to generate an "advance point" from which the exploratory subroutine is used again. If the overall result of the pattern move and subsequent exploration is an improvement, it defines a new base point and the process repeats. Otherwise, the procedure restarts from the most recent base point, with an exploratory sequence. Pattern Move Search is a general framework, into which almost any other algorithm can be fitted as exploratory subroutine. With an appropriate choice for this subroutine, it has been found to be an efficient "valley-follower", and gives reasonably rapid (but first-order) convergence in most cases. It can handle constraints if the exploratory subroutine has this capability, and if the main program is modified to enforce feasibility initially and after each pattern move.

7. GRADIENT METHODS

For unconstrained cases, the basic equation of the gradient, or steepest descent, approach to function minimization is

$$\text{new } \underline{x} = \underline{x} - k\, H\, \underline{h}\,(\underline{x}) \qquad (15)$$

where k is a positive scalar multiplier which may be held constant or iteratively adjusted, and the positive-definite matrix H is the "inverse metric" i.e., the matrix whose inverse defines vector magnitudes by the relation

$$|\Delta \underline{x}|^2 \triangleq \Delta \underline{x}^T H^{-1} \Delta \underline{x} \qquad (16)$$

The use of a factor like H in Equation (15) is unavoidable since the vectors $\underline{x}$ and $\underline{h}$ do not transform alike under coordinate transformations, and generally do not even have the same dimensionality. The convergence properties of the gradient method can be studied by considering the special case of a quadratic function with its minimum at $\underline{x} = 0$. This function is of the form

$$f(\underline{x}) = 0.5\, \underline{x}^T M_o \underline{x} \tag{17}$$

so its gradient is

$$\underline{h}(\underline{x}) = M_o \underline{x} \tag{18}$$

If the vectors $\underline{v}_i$ and scalars α_i are the eigenvectors and eigenvalues, respectively, of HM_o, $\underline{x}$ has an expansion of the form

$$\underline{x} = \sum_i c_i \underline{v}_i \tag{19}$$

with scalar coefficients c_i. The eigenvectors have the orthogonality property

$$\underline{v}_i^T H^{-1} \underline{v}_j = 0 = \underline{v}_i^T M_o \underline{v}_j \qquad \text{if } i \neq j \tag{20}$$

so

$$|\underline{x}|^2 = \sum_i |c_i|^2 \tag{21}$$

Expressing both sides of Equation (15) in terms of eigenvectors gives

$$\text{new } c_i = (1 - k\alpha_i)\, c_i \tag{22}$$

If all the α_i are nearly equal, i.e., if HM_o approximates a multiple of the unit matrix, it is possible to choose k so that

$$|1 - kx_i| \ll 1 \tag{23}$$

for all i, and the gradient method gives rapid convergence. In the more usual case where the α_i differ greatly from each other (in particular, if the magnitudes of the largest and smallest eigenvalues have a large ratio) this is not possible. every choice of k either makes some c_i increase in magnitude, or leaves other c_i almost unchanged, so a k chosen small enough for stability gives very sluggish convergence. Dynamic variation of k from iteration to iteration alleviates this situation only slightly. Ultimate convergence of $f(\underline{x})$ to a local minimum can be assured (if H is positive definite) by controlling k by rules that force monotone decrease of $f(\underline{x})$, but computational efficiency depends on choosing the metric H^{-1} to be similar to M_o, the matrix of second partial derivatives at the solution-point. Davidon's method, described in Section 8 of this Appendix, is an automatic procedure for iteratively adjusting H to make it approach $M^{-1}(\underline{x})$ if possible, but remain positive definite always.

For constrained cases, the gradient method is replaced by the "projected gradient" method of Rosen (Refs 11, 12) or the "reduced gradient" method of Wolfe (Ref 4). In the projected gradient method, the specified $\Delta\underline{x}$ at each iteration is the sum of two displacements: a displacement which attempts to null all deviations from the active constraints, plus another displacement parallel to the projection of the gradient vector unto the constraint surfaces. In the reduced gradient method, the active constraints are used to eliminate some components of $\underline{x}$ by determining them as functions of the remaining components, and the unconstrained gradient method is then applied in a smaller-dimensional space. In both these techniques, efficient means are provided for treating changes in the set of active constraints. However for both methods, efficiency depends on an appropriate choice of metric. No automatic procedure for iteratively adjusting the metric (analogous to Davidon's method) has yet been proposed for constrained cases, although Kelley's method (Ref 13) accomplishes essentially the same result.

Three gradient methods which warrant special discussion have already been mentioned in the Introduction. One of these is the NDS program of Beskind (Ref 3). This program is based on the projected-gradient method. It employs a logarithmic scaling which gives a reasonable metric in most practical cases. Also, computation is economized by a method of obtaining and updating approximate gradient information for the objective and constraint functions, as a byproduct of function evaluations performed in the natural course of the search for the optimum. The other two methods are POP and COP (Refs 1 and 2) which are slightly different versions of the same algorithm, designed for different computers. The algorithm is a sophisticated form of the Simplex (LP) method, applied by local linearization. A number of ingenious techniques are employed to minimize computation.

8. SECOND-ORDER METHODS FOR UNCONSTRAINED MINIMIZATION

There are three principal methods of unconstrained minimization which are commonly referred to as second-order methods: Davidon's method (Ref 8), the Fletcher-Reeves conjugate gradient method (Ref 14), and Powell's conjugate directions method (Ref 15). Strict mathematical proofs of second-order convergence for these methods have not been given, but their convergence is known to be super-linear (faster than first order).

In Davidon's method, which has already been referred to, each change of $\underline{x}$ and h $(\underline{x})$ is followed by an adjustment of the H matrix, of the form

$$\text{new } H = d_1 (H\Delta\underline{h})(H\Delta\underline{h})^T + d_2 (\Delta\underline{x})(\Delta\underline{x})^T \tag{24}$$

where d_1 and d_2 are computed coefficients. Since Davidon's paper is not easily accessible, his method is most widely known in a slightly modified form given by Fletcher and Powell (Ref 16). However, some users of the modified method have reported numerical difficulties (loss of positive definiteness, due to computational errors) which apparently do not occur with Davidon's original method.

Both the original and the Fletcher-Powell versions of Davidon's method require analytical evaluation of the gradient vector $\underline{h}(\underline{x})$. A form of Davidon's method using finite differences to determine approximately $\underline{h}(\underline{x})$ from changes of $f(\underline{x})$ has recently been reported by Stewart (Ref 17). This constitutes a significant advance.

The Fletcher-Reeves method of conjugate gradients uses analytic evaluation of the gradient, but could presumably be modified to use finite differences instead. Its basic cycle involves 3 vectors: $\underline{x}$, $\underline{h}$, and $\underline{s}$. Initially, $\underline{s} = -\underline{h}$. The cycle steps are

1. Choose k to minimize $f(\underline{x} + k\underline{s})$
2. New $\underline{x} = \underline{x} + k\underline{s}$
3. Evaluate new $\underline{h} = \underline{h}$ (new $\underline{x}$)
4. $b = |\text{new } \underline{h}|^2 / |\underline{h}|^2$

5. New $\underline{s} = b\underline{s}$ - (new $\underline{h}$)
6. Go to 1

For a quadratic function, this procedure gives the exact minimum in n cycles. For non-quadratic functions, it gives rapid convergence, but requires periodic restart every n cycles. A modified version not requiring restart is probably possible.

In Powell's method of conjugate directions, the algorithm remembers a set of n search directions, and performs a one-dimensional minimizing search in each of these directions in sequence. The net resulting change of $\underline{x}$ defines a new direction, which (usually) replaces one of the stored directions. No analytical derivatives are used, but equivalent information is obtained from changes of function value. The method finds the exact minimum of a quadratic function in n cycles, and has rapid (superlinear) convergence for non-quadratic functions. However, there are reports of decreased efficiency when n is large (above 15 or so). None of these second-order methods have, as yet, been generalized to deal with constraints, except by the use of penalty functions.

9. PENALTY-FUNCTION METHODS FOR CONSTRAINTS

A convenient and flexible, but approximate, method of enforcing constraints in function-minimization problems is by the use of penalty functions. In one version of the penalty-function method, the objective function f(x) is replaced by the modified objective function

$$f_p(\underline{x}) = f(\underline{x}) + \sum K_i \, d_i^2(g_i(\underline{x})) \qquad (25)$$

where the coefficients $K_i > o$ are chosen constants, $g_i(x)$ is the i^{th} constraint function, and

$$d_i(y) = \begin{Bmatrix} y \\ \min(o, y) \end{Bmatrix} \begin{matrix} \text{if the ith constraint} \\ \text{is of the form} \end{matrix} \begin{Bmatrix} g_i = o \\ g_i \geq o \end{Bmatrix} \qquad (26)$$

Eq (25) could be generalized by using a quadratic form in the quantities d_i instead of a weighted sum of squares, but no particular advantage would be gained by doing so. The quantities d_i^2 in Eq (25) are called penalty functions. The larger K_i is chosen, the more accurately the

i^{th} constraint is enforced. Therefore, an approximate solution found with a moderate value of K_i can be improved by trying again with a larger value of K_i. The displacement from the true solution will be approximately proportional to $1/K_i$. Therefore, constraints which must be satisfied with high accuracy require large values of K_i. This is a potential source of difficulty, since large values of K_i make the function $f_p(x)$ have "valleys" with at least one steep wall. The valleys are curved if the constraint surfaces are curved, and not all algorithms for unconstrained minimization can follow such valleys efficiently. Algorithms with quadratic convergence (such as Powell's or Davidon's) become highly desirable, though other methods (e.g., Rosenbrock's or Pattern Move) may also be effective. An automatic method for adjusting the coefficients K_i has been given by Kelley (Ref 13) who uses penalty functions in conjunction with Davidon's method. As a byproduct, Kelley's ingenious method generates an excellent metric for use with the projected-gradient method in a "refinement phase" which follows his rough preliminary minimization.

In the SUMT algorithm (Ref 18) equality constraints are treated as described above, but a slightly different approach is used for inequality constraints. The penalty function is nonzero even when the constraint is satisfied, and varies inversely with distance from the constraint surface. On the violation side, the penalty is infinite: moves across this surface are rejected as absolute failures. To improve accuracy, the coefficient of the penalty function is decreased, rather than increased as for quadratic penalties. The advantage of the SUMT method is that each approximate solution is feasible, so far as inequality constraints are concerned.

It is instructive to compare the penalty-function method with the Lagrange multiplier method of function minimization. In the latter method, we form

$$F(\underline{x}, \underline{\lambda}) = f(\underline{x}) + \Sigma_i \lambda_i g_i(\underline{x}) \tag{27}$$

and seek values of the multipliers λ_i, such that the solution of the resulting unconstrained problem satisfies the constraints. We also require

$\lambda_i = 0$ for all inequality constraints which are inactive (strongly satisfied)

$\lambda_i \leq 0$ for all inequality constraints of the form $g_i(x) \geq 0$

A necessary condition for $\underline{x}_o$ to give the unconstrained minimum of $F(\underline{x},\underline{\lambda})$ is

$$\nabla F(\underline{x}_o,\underline{\lambda}) = \underline{h}(\underline{x}_o) + \sum_i \lambda_i \nabla g_i(\underline{x}_o) \qquad (28)$$

Compare this with the result of differentiating Equation (25), which is

$$\nabla f_p(\underline{x}_o) = \underline{h}(\underline{x}_o) + 2\sum_i \left[K_i d_i(g_i(\underline{x}_o))\right] \quad \nabla g_i(\underline{x}_o) \qquad (29)$$

so the two are alike if we identify

$$2K_i d_i(g_i(\underline{x}_o)) = \lambda_i \qquad (30)$$

This suggests that if K_i is increased, g_i $(\underline{x}_o)$ and d_i $(g_i$ $(x_o))$ will decrease in magnitude, and the product will approach a limit which is one-half the Lagrange multiplier. Therefore, the penalty method may be regarded as a servo for adjusting λ_i to make it enforce the i^{th} constraint, and $2K_i$ as the feedback gain for this servo. The servo has a standoff error inversely proportional to the gain.

10. COMPARATIVE STUDIES OF NLP ALGORITHMS

Relatively little has been published about extensive, objective comparisons of different algorithms for nonlinear programming. Numerous authors have reported comparisons between their proposed new methods and one or more previously existing methods, with invariably favorable conclusions. A number of reasonably objective comparisons of selected methods have been reported (see review articles in the Bibliography). An extensive survey has been conducted by Colville (Ref 19) by sending a questionnaire and set of problems to possessors of optimization algorithms, with requests for

(a) additional test problems

(b) description of algorithm

(c) performance data for the test problems (success or not, preparation time, running time, number of evaluations, etc.)

Several dozen replies were received. To correct for differences in computer speed, each reported running time was normalized by dividing by the time required for execution, on the same computer, of a standard computation defined by Colville. However, even the normalized times are difficult to interpret, since no standard was given for the degree of accuracy to be sought. More meaningful, perhaps, is the fact that for most of the methods, success was not reported for all problems. However, this may reflect limited effort rather than limited capability. Since all but one of Colville's problems involve constraints, algorithms designed purely for unconstrained minimization rated poorly in his comparisons (and rightly so).

11. NONLINEAR PROGRAMMING FOR TRAJECTORY OPTIMIZATION

There are two distinct ways of applying nonlinear programming to problems of optimal control of dynamic systems (trajectory optimization). The first way, which is called the "indirect" method for historical reasons, is to make use of necessary conditions for optimality derived from the calculus of variations (or its modern counterpart, optimal control theory) to reduce the candidate trajectories to an n-parameter family of extremals defined by given sets of differential equations. The objective function and the constraint functions then become ordinary functions of n parameters, and the algorithms for finite-dimensional NLP are applicable. Each evaluation of the objective function or the constraints requires integration of a set of differential equations.

The second way, which is called the "direct" method, is to use NLP in function space. The n-component argument vector x is replaced by a control history $\underline{u}$ (t), regarded as a vector $\underline{u}$ in function-space, and the objective function becomes a functional of $\underline{u}$. The constraints may involve instantaneous values of $\underline{u}$(t), or functionals of $\underline{u}$, or both. Many of the NLP algorithms for n dimensions have analogs in function space. For example, the function-space analog of the n-dimensional Newton-Raphson method has been popularized by Bellmannas his "quasilinearization" method, and by

McGill and Roberts as the "Generalized Newton-Raphson" method (Ref 20). The "Steepest ascent" methods of Bryson and Kelley (Refs 21, 22) are analogous to the projected gradient method in n variables. As might be expected, these methods are critically dependent on the choice of metric. The convergence properties can be analyzed by considering the case where the objective function is a quadratic functional of $\underline{u}$, written symbolically as

$$f = 0.5\underline{u}^T M \underline{u} \tag{31}$$

where M is a linear operator in function space. The inverse metric is defined by some other linear operator H, which must have an easily computed inverse. Then

$$|\underline{u}|^2 \triangleq \underline{u}^T H^{-1} \underline{u} \tag{32}$$

The convergence analysis sketched in Section 7 can be carried out in function space by considering the eigenvalue problem

$$\alpha H^{-1} \underline{v} = M\underline{v}$$

There will generally be an infinite number of eigenvalues. If their magnitudes vary over a wide (or infinite) range, convergence will be poor (or impossible). Under fairly general conditions, an especially favorable choice of the metric is

$$\underline{u}^T H^{-1} \underline{u} \triangleq \int_{t_u}^{t_f} \underline{u}^T(t) W(t) \underline{u}(t) dt \tag{33}$$

where W(t) is the matrix of partial derivatives of the variational Hamiltonian with respect to control variables. With this choice of metric, it can be shown that the eigenvalues approach a finite limit, so all but a finite number of them will be within a small range. This greatly improves the convergence properties, and is the theoretical justification of the "Min-H" method of Kelley (Ref 22) and Gottleib (Ref 23). In favorable cases, the "Min-H" method can give excellent convergence. Gottleib has generalized the "Min-H" method to cases which cannot be handled by the metric given by Eq. (33). His generalization involves certain matrices which must be chosen arbitrarily, and critically affect the convergence. No general rules are available for selecting these matrices, but Gottleib has been able to find good choices for his example cases.

A function-space analog of the Fletcher-Reeves conjugate gradient method has been proposed by Mitter and Lasdon (Ref. 24). The second order convergence of the conjugate gradient method in n dimensions does not carry over into function space, but a major improvement over ordinary gradient methods can be expected. Theory indicates that this method, modified by use of the "Min-H" metric, should be capable of convergence rates not greatly inferior to a second order method.

12. SELECTED BIBLIOGRAPHY ON NONLINEAR PROGRAMMING

12.1 Books on NLP and Optimization

A. Balakvishnan and C. Neustadt (Eds), "Computing Methods in Optimization Problems", Academic Press, 1964

R. Bellmann (Ed), "Mathematical Optimization Techniques", Univ. of California Press, 1963

G. Hadley, "Nonlinear and Dynamic Programming", Addison-Wesley, 1964

G. Leitmann (Ed), "Optimization Techniques with Applications to Aerospace Systems", Academic Press, 1962

T. Vogl and A Lavi (Eds), "Recent Advances in Optimization Techniques", Wiley, 1965

D.J. Wilde, "Optimum Seeking Methods", Prentice-Hall, 1964

G. Zoutendizk, "Methods of Feasible Directions", Elsevier, 1960

12.2 Review Articles

J. Breakwell, "The Optimization of Trajectories", J. SIAM 7, 215-247 (1959)

A. Colville, "Nonlinear Programming Study" (Preliminary only; replies to a questionnaire circulated with test problems), IBM New York Scientific Center, 590 Madison Avenue, New York

W. Dorn, "Nonlinear Programming - A Survey", Manage. Sci. 9 171-208 (1963)

T. Edelbaum, "Theory of Maxima and Minima", Chapter of "Optimization Techniques with Applications to Aerospace Systems", G. Leitmann, (Ed), Academic Press, 1962

P. Fleischer, "Optimization Techniques", Chapter of "System Analysis by Digital Computer", F. Kou (Ed), Wiley, 1966

R. Fletcher, "Function Minimization Without Evaluating Derivatives - A Review", Brit. Comp. J. 8, 33-41 (1965)

H. Kelley, "Method of Gradients" Chapter of "Optimization Techniques with Applications to Aerospace Systems", G. Leitmann (Ed), Academic Press, 1962

A. Leon, "A Comparison Among Eight Known Optimizing Techniques" Chapter of "Recent Advances in Optimization Techniques" T. Vogl and A. Lavi (Eds), Wiley, 1965

J. Lewallen and B. Tapley, "Analysis and Comparison of Several Numerical Optimization Methods" (for trajectories), AIAA Paper, 58-67, January 1967

B. Paiwonsky, "Optimal Control: A Review of Theory and Practice", AIAA Journal 3, 1985-2006

M. Powell, "Minimization of Functions of Several Variables" Chapter of unpublished book; no further information

H. Spang, "A Review of Minimization Techniques for Nonlinear Functions", SIAM Review 4, 343-365 (1962)

C. Wood, "Review of Design Optimization Techniques", IEEE Trans. Systems Sc. and Cybemetics 1, 14-20 (1965)

P. Wolfe, "Recent Developments in Nonlinear Programming", RAND Corp. Papers, P-2063 and P-2333-1 (1961), R-401PR (1962)

12.3 Significant Papers on Nonlinear Programming Methods (for N Variables)

(No author) "1800 Control Optimization Program (COP) Application Descriptions", IBM Report H20-0208-1

J. Abadie, "Generalization of Wolfe's Reduced-Gradient Method for the Case of Nonlinear Constraints", Proc. IFORS Congress, Boston, September 1966 HR7 262/0-JA-JC MSA

J. Barnes, "An Algorithm for Solving Nonlinear Equations Based on the Secant Method", Brit. Comp. J 8, 66-72 (1965)

B. Beskind, "An N-Dimensional Search and Optimization Procedure", Aerospace Corp. Report TDR-269 (4550-20)-2, December 1963

C. Broyden, "A Class of Methods for Solving Nonlinear Simultaneous Equations", Math. Comp. 18, 577-593 (1965)

C. Carroll, "The Created Response Surface Technique of Optimizing Nonlinear, Restrained Systems", Oper. Res. 9, 169-184 (1961)

A. Colville, "Process Optimization Program (POP) for the IBM 7040/40 or 7090/94" Share General Program Library, H9IBM007

C. Davidon, "Variable Metric Method for Minimization", Argonne National Lab., ANL-5990, November 1959, 21pp

A. Fiacco and G. McCormick, "Computational Algorithm for the Sequential Unconstrained Minimization Technique (SUMT) for Nonlinear Programming", Manag. Sci. 10, 601-617 (1964)

A. Fiacco and G. McCormick, "Extensions of SUMT for Nonlinear Programming: Equality Constraints and Extrapolation", Manage. Sci. 12, 816-820 (1966)

R. Fletcher and M. Powell, "A Rapidly Convergent Descent Method for Minimization", Brit. Comp. J 6, 163-168 (1963)

R. Fletcher and C. Reeves, "Function Minimization by Conjugate Gradients", Brit. Comp. J 7, 149-154 (1964)

M. Hestenes, "The Conjugate-Gradient Method for Solving Linear Systems", Proc. of Symp. in Appl. Math, McGraw-Hill, Vol. VI, 83-102 (1956)

R. Hooke and T. Jeeves, "Direct Search Solution of Numerical and Statistical Problems", J. Assoc. Comput. Mach. 8, 212-229 (1961)

H. Kelley, et. al., "An Accelerated Gradient Method for Parameter Optimization with Nonlinear Constraints", presented at AAS Space Flight Mechanics Specialist Conference, July 1966

H. Kuhn and A. Tucker, "Nonlinear Programming", Proc. Second Berkeley Symposium on Math, Stat. and Probability, 481-492 (1951)

M. Powell, "A Method for Minimizing a Sum of Squares of Nonlinear Functions Without Calculating Derivatives", Brit. Comp. J. 7, 303-307

J. Rosen, "The Gradient Projection Method for Nonlinear Programming, Part I: Linear Constraints", J. SIAM 8, p 181-217

J. Rosen, "The Gradient Projection Method for Nonlinear Programming, Part II: Nonlinear Constraints", SIAM J. 9, 514-532 (1961)

H. Rosenbrock, "An Automatic Method of Finding the Greatest or Least Value of a Function", Brit. Comp. J. 3, 175-184 (1960)

G. Stewart III, "A Modification of Davidon's Minimization Method to Accept Difference Approximations of Derivatives", J. ACM 14, 72-83 (1967)

12.4 Significant Papers on Trajectory Optimization Methods

J. Breakwell, J. Speyer, A. Bryson "Optimization and Control of Nonlinear Systems Using the Secind Variation", SIAM J. on Control, Ser. A 1, 193-223 (1963)

K. Brown and G. Johnson, "Real-Time Optimal Guidance", IBM Federal Systems Division Report #66-220-001, August 1966

A. Bryson and W. Denham, "A Steepest-Ascent Method for Solving Optimum Programming Problems", J. Appl. Mech., Trans. ASME, Ser. E, 29 247-257

S. Conte, "The Numerical Solution of Linear Boundary Value Problems", SIAM Review 8, 309-321 (1966)

R. Gottleib, "Rapid Convergence to Optimum Solutions Using a Min-H Strategy", AIAA Journal 5, 322-329 (1967)

M. Handelsman, "Optimal Free-Space Fixed-Thrust Trajectories Using Impulsive Trajectories as Starting Iteratives", AIAA J. 4, 1077-1082 (1966)

H. Ingram, "The Automation of Optimization Problems", AAS Preprint 66-116, July 1966

H. Kelley, "Gradient Theory of Optimal Flight Paths", J. ARS 50, 947-954 (1960)

H. Kelley, "Method of Gradients". Chapter in "Optimization Techniques with Applications to Aerospace Systems", G. Leitmann, (Ed), Academic Press, 1962

H. Kelley, R. Kopp, H. Moyer, "A Trajectory Optimization Technique Based On the Teory of the Second Variation", AIAA Paper 63-415 (1963)

G. Leitmann, "The Optimization of Rocket Trajectories - A Survey", Progress in Astronautical Sciences (S. Singer, Ed) N. Holland, 1961

P. Lion and M. Handelsman, "The Primer Vector on Fixed-Time Impulsive Trajectories", AIAA Paper 67-54, January 1967

G. McCue, "Satellite Orbit Transfer Studies (Final Report)", North American Aviation, Space and Information Systems Division, STD 66-1224, August 1966

G. McCue, "Quasilinearization Determination of Optimum Finite-Thrust Orbital Transfers", North American Aviation, Inc., SID 66-1278 (1966)

R. McGill and P. Kenneth, "Solution of Variational Problems By Means of a Generalized Newton-Raphson Method", AIAA J. 2, 1761-1766 (1964)

S. Mitter, L. Lasdon, A. Waren, "The Method of Conjugate Gradients for Optimal Control Problems", IEEE Journal, June 1966, 904-905

S. Pines, "Optimal Guidance for Ascent Trajectories Into Circular Orbits", AIAA Paper 67-56, January 1967

H. Robbins, "An Analytical Study of the Impulsive Approximation", AIAA J. 4, 1417-1423 (1966)

R. Wingrove, "A Method of Trajectory Optimization by Fast-Time Repetitive Computations", NASA Ames Res. Ctr., NASA-TND-3404, April 1966

REFERENCES

1. Colville, A., "Process Optimization Program (POP) for the IBM 7040/40 or 7090/94" Share General Program Library, H91BM007

2. (No Author) "1800 Control Optimization Program (COP) Application Descriptions", IBM Report H20-0208-1

3. Beskind, B., "An N-Dimensional Search and Optimization Procedure", Aerospace Corp. Report TDR-269 (4550-20)-2, December 1963

4. Abadie, J., "Generalization of Wolfe's Reduced-Gradient Method for the Case of Nonlinear Constraints", Proc. IFORS Congress, Boston, September 1966, HR7 262/0 - JA - JC MSA

5. Kuhn, H. and Tucker, A., "Nonlinear Programming", Proc. Second Berkeley Symposium on Math Stat. and Probability, 481-492 (1951)

6. Hadley, G., "Nonlinear and Dynamic Programming", Addison-Wesley, 1964

7. Powell, M., "Minimization of Functions of Several Variables", Chapter of unpublished book, no further information

8. Davidon, C., "Variable Metric Method for Minimization", Argonne National Lab., ANL-5990, November 1959

9. Rosenbrock, H., "An Automatic Method of Finding the Greatest or Least Value of a Function", Brit. Computer J. 3, 175 - 184 (1960)

10. Hooke R., and Jeeves, T., "Direct Search Solution of Numerical and Statistical Problems", J. Assoc. Comput. Mach. 8, 212-219 (1961)

11. Rosen, J., "The Gradient Projection Method for Nonlinear Programming, Part I : Linear Constraints", J. SIAM 8, pp 181-217

12. Rosen, J., "The Gradient Projection Method for Nonlinear Programming, Part II : Nonlinear Constraints", SIAM Journal 9, 514 - 532 (1961)

13. Kelley, H., et. al., "An Accelerated Gradient Method for Parameter Optimization with Nonlinear Constraints", presented at AAS Space Flight Mechanics Specialist Conference, July 1966

14. Fletcher, R., and Reeves, C., "Function Minimization by Conjugate Gradients", Brit. Computer J. 7, 149 - 154 (1964)

15. Powell, M., "An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives", Brit. Computer J. 7, 155 - 162 (1964)

16. Fletcher, R., and Powell, M., "A Rapidly Convergent Descent Method for Minimization", Brit. Computer J. 6, 163 - 168 (1963)

17. Stewart, G., III, "A Modification of Davidon's Minimization Method to Accept Difference Approximations of Derivatives", J. ACM 14, 72 - 78 (1967)

18. Fiacco, A., and McCormick, G., "Computational Algorithm for the Sequential Unconstrained Minimization Technique (SUMT) for Nonlinear Programming", Manag. Sci. 10, 601 - 617 (1964)

19. Colville, A., "Nonlinear Programming Study" (Preliminary only: replies to a questionnaire circulated with test problems) IBM New York Scientific Center, 590 Madison Avenue, New York

20. McGill, R., and Kenneth, P., "Solution of Variational Problems By Means of a Generalized Newton-Raphson Method", AIAA J. 2, 1761 - 1766 (1964)

21. Bryson, A., and Denham, W., "A Steepest-Ascent Method for Solving Optimum Programming Problems", J. Appl. Mech., Trans. ASME, Ser. E, 29 247 - 57

22. Kelley, H., "Method of Gradients" Chapter in "Optimization Techniques with Applications to Aerospace Systems", G. Leitmann, Ed., Academic Press 1962

23. Gottleib, R., "Rapid Convergence to Optimum Solutions Using a Min-H Strategy", AIAA Journal 5, 322 - 329 (1967)

24. Mitter, S., Lasdon, L., Waren, A., "The Method of Conjugate Gradients for Optimal Control Problems", IEEE Journal, June 1966, 904 - 905